

Figure 23.7: Illustration of attention in the music transformer. Different colored lines correspond to the 6 attention heads. Line thickness corresponds to attention weights. From Figure 8 of [Hua+18a]. Used with kind permission of Anna Huang.

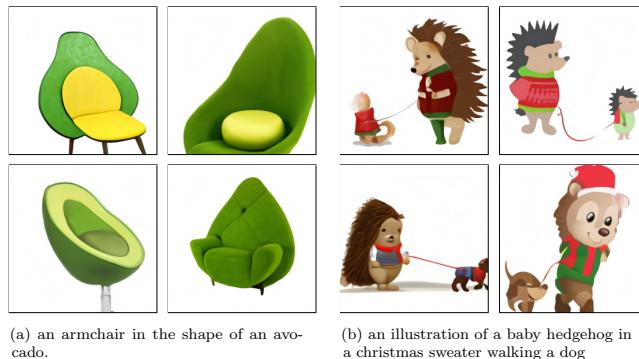


Figure 23.8: Some images generated by the DALL-E model in response to a text prompt. (a) “An armchair in the shape of an avocado”. (b) “An illustration of a baby hedgehog in a christmas sweater walking a dog”. From <https://openai.com/blog/dall-e>. Used with kind permission of Aditya Ramesh.

straightforward, and most of the effort went into data collection (they scrape the web for 250 million image-text pairs) and scaling up the training (they fit a model with 12 billion parameters). Here we just focus on the algorithmic methods.

The basic idea is to transform an image  $\mathbf{x}$  into a sequence of discrete tokens  $\mathbf{z}$  using a discrete VAE model (Section 22.6.5), which defines a model of the form  $p(\mathbf{x}, \mathbf{z})$ . We then fit a transformer to the concatenation of the image tokens  $\mathbf{z}$  and text tokens  $\mathbf{y}$  to get a model of the form  $p(\mathbf{z}, \mathbf{y})$ .

To sample an image  $\mathbf{x}$  given a text prompt  $\mathbf{y}$ , we sample a latent code  $\mathbf{z} \sim p(\mathbf{z}|\mathbf{y})$ , and then we feed  $\mathbf{z}$  into the VAE decoder to get  $\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})$ . Multiple images are generated for each prompt, and these are then ranked according to a pre-trained critic, which gives them scores depending on how well the generated image matches the input text:  $s_n = \text{critic}(\mathbf{x}_n, \mathbf{y}_n)$ . The critic they used was the contrastive CLIP model (see Section 34.1). This discriminative reranking significantly improves the results.

Some sample results are shown in Figure 23.8, and more can be found online at <https://openai.com/blog/dall-e/>. The image on the right of Figure 23.8 is particularly interesting, since the prompt — “An illustration of a baby hedgehog in a christmas sweater walking a dog” — arguably

1 requires that the model solve the “**variable binding problem**”. This refers to the fact that the  
2 sentence implies the hedgehog should be wearing the sweater and not the dog. We see that the model  
3 sometimes interprets this correctly, but not always: sometimes it draws both animals with Christmas  
4 sweaters. In addition, sometimes it draws a hedgehog walking a smaller hedgehog. The quality of the  
5 results can also be sensitive to the form of the prompt. Thus although impressive, this technology is  
6 clearly not yet reliable.  
7

8 Since being released in January 2021, many alternatives to DALL-E have been proposed. For  
9 example, Google released **XMC-GAN** [Zha+21], which uses a GAN (Chapter 27) instead of a  
10 VQ-transformer for the image generation. And in December 2021, OpenAI released **GLIDE** [Nic+21],  
11 which uses a diffusion model (Chapter 26) instead of the VQ-transformer for the image generation.

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

# 24 Normalizing Flows

This chapter was written by George Papamakarios and Balaji Lakshminarayanan.

## 24.1 Introduction

In this chapter we discuss **normalizing flows**, a class of flexible density models that can be easily sampled from and whose exact likelihood function is efficient to compute. Such models can be used for many tasks, such as density modeling, inference and generative modeling. We introduce the key principles of normalizing flows and refer to recent surveys by Papamakarios et al. [Pap+19] and Kobyzev, Prince, and Brubaker [KPB19] for readers interested in learning more. See also <https://github.com/janosh/awesome-normalizing-flows> for a list of papers and software packages.

### 24.1.1 Preliminaries

Normalizing flows create complex probability distributions  $p(\mathbf{x})$  by passing random variables  $\mathbf{u} \in \mathbb{R}^D$ , drawn from a simple **base distribution**  $p(\mathbf{u})$  through a nonlinear but *invertible* transformation  $\mathbf{f} : \mathbb{R}^D \rightarrow \mathbb{R}^D$ . That is,  $p(\mathbf{x})$  is defined by the following process:

$$\mathbf{x} = \mathbf{f}(\mathbf{u}) \quad \text{where} \quad \mathbf{u} \sim p(\mathbf{u}). \tag{24.1}$$

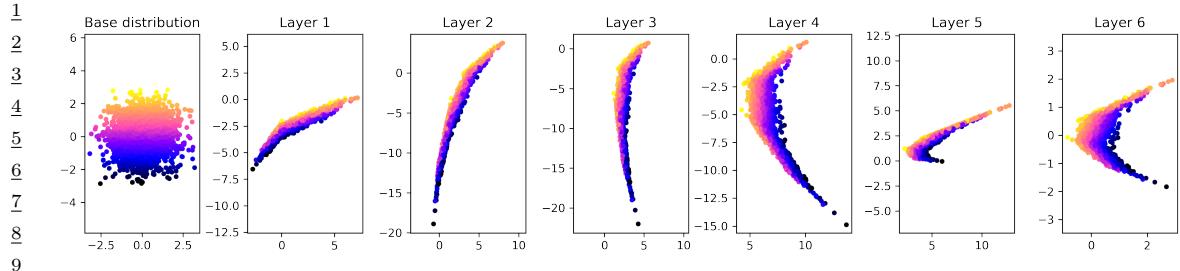
The base distribution is typically chosen to be simple, for example standard Gaussian or uniform, so that we can easily sample from it and compute the density  $p(\mathbf{u})$ . A flexible enough transformation  $\mathbf{f}$  can induce a complex distribution on the transformed variable  $\mathbf{x}$  even if the base distribution is simple, as illustrated in Figure 24.1.

Sampling from  $p(\mathbf{x})$  is straightforward: we first sample  $\mathbf{u}$  from  $p(\mathbf{u})$  and then compute  $\mathbf{x} = \mathbf{f}(\mathbf{u})$ . To compute the density  $p(\mathbf{x})$ , we rely on the fact that  $\mathbf{f}$  is invertible. Let  $\mathbf{g}(\mathbf{x}) = \mathbf{f}^{-1}(\mathbf{x}) = \mathbf{u}$  be the inverse mapping, which “**normalizes**” the data distribution by mapping it back to the base distribution (which is often a Normal distribution). Using the change-of-variables formula for random variables from Equation (2.264), we have

$$p_x(\mathbf{x}) = p_u(\mathbf{g}(\mathbf{x})) |\det \mathbf{J}(\mathbf{g})(\mathbf{x})| = p_u(\mathbf{u}) |\det \mathbf{J}(\mathbf{f})(\mathbf{u})|^{-1}, \tag{24.2}$$

where  $\mathbf{J}(\mathbf{f})(\mathbf{u}) = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}|_{\mathbf{u}}$  is the Jacobian matrix of  $\mathbf{f}$  evaluated at  $\mathbf{u}$ . Taking logs of both sides of Equation (24.2), we get

$$\log p_x(\mathbf{x}) = \log p_u(\mathbf{u}) - \log |\det \mathbf{J}(\mathbf{f})(\mathbf{u})|. \tag{24.3}$$



10 *Figure 24.1: Mapping a 2d standard Gaussian to a more complex distribution using an invertible MLP.*  
11 Points are color-coded by their initial position along the y-axis. Adapted from [Jan18]. Generated by  
12 `flow_2d_mlp.ipynb`.

13

14

15 As discussed above,  $p(\mathbf{u})$  is typically easy to evaluate. So, if one can use flexible invertible transfor-  
16 mations  $\mathbf{f}$  whose Jacobian determinant  $\det \mathbf{J}(\mathbf{f})(\mathbf{u})$  can be computed efficiently, then one can construct  
17 complex densities  $p(\mathbf{x})$  that allow exact sampling and efficient exact likelihood computation. This is  
18 in contrast to latent variable models, which require methods like variational inference to lower-bound  
19 the likelihood.

20 One might wonder how flexible are the densities  $p(\mathbf{x})$  obtained by transforming random variables  
21 sampled from simple  $p(\mathbf{u})$ . It turns out that we can use this method to approximate any smooth  
22 distribution. To see this, consider the scenario where the base distribution  $p(\mathbf{u})$  is a one-dimensional  
23 uniform distribution. Recall that inverse transform sampling (Section 11.3.1) samples random  
24 variables from a uniform distribution and transforms them using the inverse Cumulative Distribution  
25 Function (CDF) to generate samples from the desired density. We can use this method to sample  
26 from any one-dimensional density as long as the transformation  $\mathbf{f}$  is powerful enough to model the  
27 inverse CDF (which is a reasonable assumption for well-behaved densities whose CDF is invertible  
28 and differentiable). We can further extend this argument to multiple dimensions by first expressing  
29 the density  $p(\mathbf{x})$  as a product of one-dimensional conditionals using the chain rule of probability,  
30 and then applying inverse transform sampling to each one-dimensional conditional. The result is a  
31 normalizing flow that transforms a product of uniform distributions into any desired distribution  
32  $p(\mathbf{x})$ . We refer to [Pap+19] for a more detailed proof.

33 How do we define flexible invertible mappings whose Jacobian determinant is easy to compute?  
34 We discuss this topic in detail in Section 24.2, but in summary, there are two main ways. The first  
35 approach is to define a set of simple transformations that are invertible by design, and whose Jacobian  
36 determinant is easy to compute; for instance, if the Jacobian is a triangular matrix, its determinant  
37 can be computed efficiently. The second approach is to exploit the fact that a composition of invertible  
38 functions is also invertible, and the overall Jacobian determinant is just the product of the individual  
39 Jacobian determinants. More precisely, if  $\mathbf{f} = \mathbf{f}_N \circ \dots \circ \mathbf{f}_1$  where each  $\mathbf{f}_i$  is invertible, then  $\mathbf{f}$  is also  
40 invertible, with inverse  $\mathbf{g} = \mathbf{g}_1 \circ \dots \circ \mathbf{g}_N$  and log Jacobian determinant given by

41

$$\log |\det \mathbf{J}(\mathbf{g})(\mathbf{x})| = \sum_{i=1}^N \log |\det \mathbf{J}(\mathbf{g}_i)(\mathbf{u}_i)| \quad (24.4)$$

42

43 where  $\mathbf{u}_i = \mathbf{f}_i \circ \dots \circ \mathbf{f}_1(\mathbf{u})$  is the  $i$ 'th intermediate output of the flow. This allows us to create  
44 complex flows from simple components, just as graphical models allow us to create complex joint  
45

1 distributions from simpler conditional distributions.

2 Finally, a note on terminology. An invertible transformation is also known as a **bijection**. A  
3 bijection that is differentiable and has a differentiable inverse is known as a **diffeomorphism**. The  
4 transformation  $\mathbf{f}$  of a flow model is a diffeomorphism, although in the rest of this chapter we will refer  
5 to it as a “bijection” for simplicity, leaving the differentiability implicit. The density  $p_x(\mathbf{x})$  of a flow  
6 model is also known as the **pushforward** of the base distribution  $p_u(\mathbf{u})$  through the transformation  
7  $\mathbf{f}$ , and is sometimes denoted as  $p_x = \mathbf{f}_* p_u$ . Finally, in mathematics the term “flow” refers to any  
8 family of diffeomorphisms  $\mathbf{f}_t$  indexed by a real number  $t$  such that  $t = 0$  indexes the identity function,  
9 and  $t_1 + t_2$  indexes  $\mathbf{f}_{t_2} \circ \mathbf{f}_{t_1}$  (in physics,  $t$  often represents time). In machine learning we use the term  
10 “flow” by analogy to the above meaning, to highlight the fact that we can create flexible invertible  
11 transformations by composing simpler ones; in this sense, the index  $t$  is analogous to the number  $i$  of  
12 transformations in  $\mathbf{f}_i \circ \dots \circ \mathbf{f}_1$ .

### 14 24.1.2 Example

15 In this section, we consider a simple 2d example of a normalizing flow, where  $p(\mathbf{u})$  is a 2d standard  
16 Gaussian, and  $\mathbf{f}_\theta$  is an MLP. Each layer of the MLP corresponds to the following nonlinear  
17 transformation:

$$\underline{20} \quad \mathbf{z}_{l+1} = \varphi(\mathbf{A}_l \mathbf{z}_l + \mathbf{b}_l), \quad (24.5)$$

22 where  $\varphi$  is a nonlinear activation function applied elementwise and  $\mathbf{A}_l$  is an invertible square matrix.  
23 We cannot use a ReLU activation function, since it is not invertible, so instead we will use a  
24 parameterized ReLU function, which is like a leaky ReLU with a learnable slope  $\lambda_j > 0$  for each  
25 dimension:

$$\underline{27} \quad \varphi(\mathbf{x})_j = x_j \mathbb{I}(x_j \geq 0) + \lambda_j x_j \mathbb{I}(x_j < 0). \quad (24.6)$$

29 We now discuss how to compute the log Jacobian determinant of layer  $l$ . Let  $\mathbf{x} = \mathbf{z}_l$  be its input,  
30  $\mathbf{a} = \mathbf{A}\mathbf{x} + \mathbf{b} = \mathbf{f}(\mathbf{x})$  be its linear transformation (we drop the  $l$  subscript from the parameters for  
31 simplicity), and  $\mathbf{y} = \varphi(\mathbf{a})$  be its output. From Equation (24.4), we have

$$\underline{32} \quad \log |\det \mathbf{J}(\varphi \circ f)| = \log |\det \mathbf{J}(\varphi)| + \log |\det \mathbf{J}(f)|. \quad (24.7)$$

34 Since  $\varphi$  is applied elementwise, we have

$$\underline{36} \quad \mathbf{J}(\varphi) = \text{diag} \left( \frac{\partial y_i}{\partial a_i} \right), \quad (24.8)$$

38 where

$$\underline{40} \quad \frac{\partial y_i}{\partial a_i} = \mathbb{I}(a_i \geq 0) + \lambda_i \mathbb{I}(a_i < 0). \quad (24.9)$$

42 For the second term in Equation (24.7), we have

$$\underline{44} \quad \mathbf{J}(f)_{ij} = \frac{\partial a_i}{\partial x_j} = \frac{\partial (\mathbf{A}_{i,:}^\top \mathbf{x} + b_i)}{\partial x_j} = A_{ij}. \quad (24.10)$$

1 Hence  $\mathbf{J}(f) = \mathbf{A}$ . In general, computing the determinant of  $\mathbf{A}$  takes cubic time in the number of  
2 dimensions. In Section 24.2, we will discuss transformations which support linear time computation  
3 of their Jacobian determinants.  
4

5 We can compose multiple such layers to define a flexible density  $p_{\theta}(\mathbf{x})$ . This is illustrated in  
6 Figure 24.1, where we map a 2d Gaussian to a more complex 2d distribution using a 6-layer MLP.  
7 Note that we needed to use a deep model even for this simple 2d example because each layer is  
8 limited in its expressive power. We consider more complex invertible transformations in Section 24.2.  
9

### 10 24.1.3 How to train a flow model

11 There are two common applications of normalizing flows. The first one is density estimation of  
12 observed data, which is achieved by fitting  $p_{\theta}(\mathbf{x})$  to the data and using it as an estimate of the  
13 data density, potentially followed by generating new data from  $p_{\theta}(\mathbf{x})$ . The second one is variational  
14 inference, which involves sampling from and evaluating a variational posterior  $q_{\theta}(\mathbf{z}|\mathbf{x})$  parameterized  
15 by the flow model. As we will see below, these applications optimize different objectives and impose  
16 different computational constraints on the flow model.  
17

#### 18 24.1.3.1 Density estimation

19 Density estimation requires maximizing the likelihood function in Equation (24.2). This requires that  
20 we can efficiently evaluate the inverse flow  $\mathbf{u} = f^{-1}(\mathbf{x})$  and its Jacobian determinant  $\det \mathbf{J}(f^{-1})(\mathbf{x})$   
21 for any given  $\mathbf{x}$ . After optimizing the model, we can optionally use it to generate new data. To  
22 sample new points, we require that the forward mapping  $f$  be tractable.  
23

#### 24 24.1.3.2 Variational inference

25 Normalizing flows are commonly used for variational inference to parametrize the approximate  
26 posterior distribution in latent variable models, as discussed in Section 10.4.3. Consider a latent  
27 variable model with continuous latent variables  $\mathbf{z}$  and observable variables  $\mathbf{x}$ . For simplicity, we  
28 consider the model parameters to be fixed as we are interested in approximating the true posterior  
29  $p^*(\mathbf{z}|\mathbf{x})$  with a normalizing flow  $q_{\theta}(\mathbf{z}|\mathbf{x})$ .<sup>1</sup> As discussed in Section 10.1.2, the variational parameters  
30 are trained by maximizing the evidence lower bound (ELBO), given by  
31

$$\begin{aligned} \text{33} \quad L(\boldsymbol{\theta}) &= \mathbb{E}_{q_{\theta}(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q_{\theta}(\mathbf{z}|\mathbf{x})] \end{aligned} \quad (24.11)$$

34 When viewing the ELBO as a function of  $\boldsymbol{\theta}$ , it can be simplified as follows (note we drop the  
35 dependency on  $\mathbf{x}$  for simplicity):  
36

$$\begin{aligned} \text{37} \quad L(\boldsymbol{\theta}) &= \mathbb{E}_{q_{\theta}(\mathbf{z})} [\ell_{\boldsymbol{\theta}}(\mathbf{z})]. \end{aligned} \quad (24.12)$$

38 Let  $q_{\theta}(\mathbf{z})$  denote a normalizing flow with base distribution  $q(\mathbf{u})$  and transformation  $\mathbf{z} = f_{\boldsymbol{\theta}}(\mathbf{u})$ . Then  
39 the reparametrization trick (Section 6.6.4) allows us to optimize the parameters using stochastic  
40 gradients. To achieve this, we first write the expectation with respect to the base distribution:  
41

$$\begin{aligned} \text{42} \quad L(\boldsymbol{\theta}) &= \mathbb{E}_{q_{\theta}(\mathbf{z})} [\ell_{\boldsymbol{\theta}}(\mathbf{z})] = \mathbb{E}_{q(\mathbf{u})} [\ell_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(\mathbf{u}))]. \end{aligned} \quad (24.13)$$

43 1. We denote the parameters of the variational posterior by  $\boldsymbol{\theta}$  here, which should not be confused with the model  
44 parameters which are also typically denoted by  $\boldsymbol{\theta}$  elsewhere.

Then, since the base distribution does not depend on  $\theta$ , we can obtain stochastic gradients as follows:

$$\nabla_{\theta} L(\theta) = \mathbb{E}_{q(\mathbf{u})} [\nabla_{\theta} \ell_{\theta}(f_{\theta}(\mathbf{u}))] \approx \frac{1}{N} \sum_{n=1}^N \nabla_{\theta} \ell_{\theta}(f_{\theta}(\mathbf{u}_n)), \quad (24.14)$$

where  $\{\mathbf{u}_n\}_{n=1}^N$  are samples from  $q(\mathbf{u})$ .

As we can see, in order to optimize this objective, we need to be able to efficiently sample from  $q_{\theta}(\mathbf{z}|\mathbf{x})$  and evaluate the probability density of these samples during optimization. (See Section 24.2.4.3 for details on how to do this.) This is contrast to the MLE approach in Section 24.1.3.1, which requires that we be able to compute efficiently the density of arbitrary training datapoints, but it does not require samples during optimization.

## 24.2 Constructing Flows

In this section, we discuss how to compute various kinds of flows that are invertible by design and have efficiently computable Jacobian determinants.

### 24.2.1 Affine flows

A simple choice is to use an affine transformation  $\mathbf{x} = \mathbf{f}(\mathbf{u}) = \mathbf{A}\mathbf{u} + \mathbf{b}$ . This is a bijection if and only if  $\mathbf{A}$  is an invertible square matrix. The Jacobian determinant of  $\mathbf{f}$  is  $\det \mathbf{A}$ , and its inverse is  $\mathbf{u} = \mathbf{f}^{-1}(\mathbf{x}) = \mathbf{A}^{-1}(\mathbf{x} - \mathbf{b})$ . A flow consisting of affine bijections is called an **affine flow**, or a **linear flow** if we ignore  $\mathbf{b}$ .

On their own, affine flows are limited in their expressive power. For example, suppose the base distribution is Gaussian,  $p(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ . Then the pushforward distribution after an affine bijection is still Gaussian,  $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T)$ . However, affine bijections are useful building blocks when composed with the non-affine bijections we discuss later, as they encourage “mixing” of dimensions through the flow.

For practical reasons, we need to ensure the Jacobian determinant and the inverse of the flow are fast to compute. In general, computing  $\det \mathbf{A}$  and  $\mathbf{A}^{-1}$  explicitly takes  $O(D^3)$  time. To reduce the cost, we can add structure to  $\mathbf{A}$ . If  $\mathbf{A}$  is diagonal, the cost becomes  $O(D)$ . If  $\mathbf{A}$  is triangular, the Jacobian determinant is the product of the diagonal elements, so takes  $O(D)$  time; inverting the flow requires solving the triangular system  $\mathbf{A}\mathbf{u} = \mathbf{x} - \mathbf{b}$ , which can be done with backsubstitution in  $O(D^2)$  time.

The result of a triangular transformation depends on the ordering of the dimensions. To reduce sensitivity to this, and to encourage “mixing” of dimensions, we can multiply  $\mathbf{A}$  with a permutation matrix, which has an absolute determinant of 1. We often use a permutation that reverses the indices at each layer or that randomly shuffles them. However, usually the permutation at each layer is fixed rather than learned.

For spatially structured data (such as images), we can define  $\mathbf{A}$  to be a convolution matrix. For example, GLOW [KD18b] uses  $1 \times 1$  convolution; this is equivalent to pointwise linear transformation across feature dimensions, but regular convolution across spatial dimensions. Two more general methods for modeling  $d \times d$  convolutions are presented in [HBW19], one based on stacking autoregressive convolutions, and the other on carrying out the convolution in the Fourier domain.

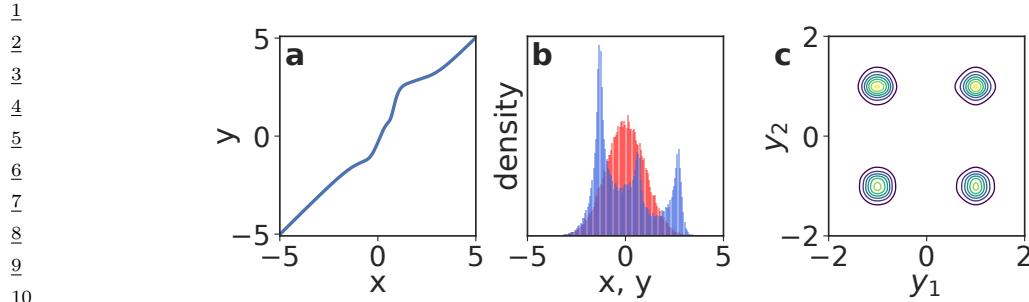


Figure 24.2: Non-linear squared flow (NLSq). Left: an invertible mapping consisting of 4 NLSq layers. Middle: red is the base distribution (Gaussian), blue is the distribution induced by the mapping on the left. Right: density of a 5-layer autoregressive flow using NLSq transformations and a Gaussian base density, trained on a mixture of 4 Gaussians. From Figure 5 of [ZR19b]. Used with kind permission of Zachary Ziegler.

16

17

### 24.2.2 Elementwise flows

19

Let  $h : \mathbb{R} \rightarrow \mathbb{R}$  be a scalar-valued bijection. We can create a vector-valued bijection  $\mathbf{f} : \mathbb{R}^D \rightarrow \mathbb{R}^D$  by applying  $h$  elementwise, that is,  $\mathbf{f}(\mathbf{u}) = (h(u_1), \dots, h(u_D))$ . The function  $\mathbf{f}$  is invertible, and its Jacobian determinant is given by  $\prod_{i=1}^D \frac{dh}{du_i}$ . A flow composed of such bijections is known as an **elementwise flow**.

On their own, elementwise flows are limited, since they do not model dependencies between the elements. However, they are useful building blocks for more complex flows, such as coupling flows (Section 24.2.3) and autoregressive flows (Section 24.2.4), as we will see later. In this section, we discuss techniques for constructing scalar-valued bijections  $h : \mathbb{R} \rightarrow \mathbb{R}$  for use in elementwise flows.

28

#### 24.2.2.1 Affine scalar bijection

30

An **affine scalar bijection** has the form  $h(u; \boldsymbol{\theta}) = au + b$ , where  $\boldsymbol{\theta} = (a, b) \in \mathbb{R}^2$ . (This is a scalar version of an affine flow.) Its derivative  $\frac{dh}{du}$  is equal to  $a$ . It is invertible if and only if  $a \neq 0$ . In practice, we often parameterize  $a$  to be positive, for example by making it the exponential or the softplus of an unconstrained parameter. When  $a = 1$ ,  $h(u; \boldsymbol{\theta}) = u + b$  is often called an **additive scalar bijection**.

36

#### 24.2.2.2 Higher-order perturbations

38

The affine scalar bijection is simple to use, but limited. We can make it more flexible by adding higher-order perturbations, under the constraint that invertibility is preserved. For example, Ziegler and Rush [ZR19b] propose the following, which they term **non-linear squared flow**:

$$h(u; \boldsymbol{\theta}) = au + b + \frac{c}{1 + (du + e)^2}, \quad (24.15)$$

44

where  $\boldsymbol{\theta} = (a, b, c, d, e) \in \mathbb{R}^5$ . When  $c = 0$ , this reduces to the affine case. When  $c \neq 0$ , it adds an inverse-quadratic perturbation, which can induce multimodality as shown in Figure 24.2. Under the

47

1 constraints  $a > \frac{9}{8\sqrt{3}}cd$  and  $d > 0$  the function becomes invertible, and its inverse can be computed  
2 analytically by solving a quadratic polynomial.

### 5 24.2.2.3 Combinations of strictly monotonic scalar functions

6 A strictly monotonic scalar function is one that is always increasing (has positive derivative everywhere)  
7 or always decreasing (has negative derivative everywhere). Such functions are invertible. Many  
8 activation functions, such as the logistic sigmoid  $\sigma(u) = 1/(1 + \exp(-u))$ , are strictly monotonic.  
9

10 Using such activation functions as a starting point, we can build more flexible monotonic functions  
11 via **conical combination** (linear combination with positive coefficients) and function composition.  
12 Suppose  $h_1, \dots, h_K$  are strictly increasing; then the following are also strictly increasing:

- 13 •  $a_1 h_1 + \dots + a_K h_K + b$  with  $a_k > 0$  (conical combination with a bias),
- 14 •  $h_1 \circ \dots \circ h_K$  (function composition).

16 By repeating the above two constructions, we can build arbitrarily complex increasing functions. For  
17 example, a composition of conical combinations of logistic sigmoids is just an MLP where all weights  
18 are positive [Hua+18b].

19 The derivative of such a scalar bijection can be computed by repeatedly applying the chain rule,  
20 and in practice can be done with automatic differentiation. However, the inverse is not typically  
21 computable in closed form. In practice we can compute the inverse using bisection search, since the  
22 function is monotonic.

### 24 24.2.2.4 Scalar bijections from integration

26 A simple way to ensure a scalar function is strictly monotonic is to constrain its derivative to be  
27 positive. Let  $h' = \frac{dh}{du}$  be this derivative. Wehenkel and Louppe [WL19] directly parameterize  $h'$  with  
28 a neural network whose output is made positive via an ELU activation function shifted up by 1.  
29 They then integrate the derivative numerically to get the bijection:

$$\text{30} \quad h(u) = \int_0^u h'(t) dt + b, \quad (24.16)$$

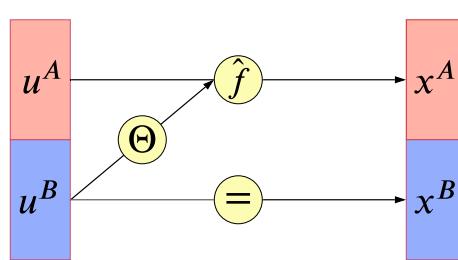
33 where  $b$  is a bias. They call this approach **unconstrained monotonic neural networks**.

34 The above integral is generally not computable in closed form. It can be, however, if  $h'$  is  
35 constrained appropriately. For example, Jaini, Selby, and Yu [JSY19] take  $h'$  to be a sum of  $K$   
36 squared polynomials of degree  $L$ :

$$\text{38} \quad h'(u) = \sum_{k=1}^K \left( \sum_{\ell=0}^L a_{k\ell} u^\ell \right)^2. \quad (24.17)$$

41 This makes  $h'$  a non-negative polynomial of degree  $2L$ . The integral is analytically tractable, and  
42 makes  $h$  an increasing polynomial of degree  $2L + 1$ . For  $L = 0$ ,  $h'$  is constant, so  $h$  reduces to an  
43 affine scalar bijection.

44 In these approaches, the derivative of the bijection can just be read off. However, the inverse is not  
45 analytically computable in general. In practice, we can use bisection search to compute the inverse  
46 numerically.



*Figure 24.3: Illustration of a coupling layer  $\mathbf{x} = f(\mathbf{u})$ . A bijection, with parameters determined by  $\mathbf{u}^B$ , is applied to  $\mathbf{u}^A$  to generate  $\mathbf{x}^A$ ; meanwhile  $\mathbf{x}^B = \mathbf{u}^B$  is passed through unchanged, so the mapping can be inverted. From Figure 3 of [KPB19]. Used with kind permission of Ivan Kobyzhev.*

### 24.2.2.5 Splines

Another way to construct monotonic scalar functions is using **splines**. These are piecewise-polynomial or piecewise-rational functions, parameterized in terms of  $K + 1$  **knots**  $(u_k, x_k)$  through which the spline passes. That is, we set  $h(u_k) = x_k$ , and define  $h$  on the interval  $(u_{k-1}, u_k)$  by interpolating from  $x_{k-1}$  to  $x_k$  with a polynomial or rational function (ratio of two polynomials). By increasing the number of knots we can create arbitrarily flexible monotonic functions.

Different ways to interpolate between knots give different types of spline. A simple choice is to interpolate linearly [Mül+19a], however this makes the derivative discontinuous at the knots. Interpolating with quadratic polynomials [Mül+19a] gives enough flexibility to make the derivative continuous. Interpolating with cubic polynomials [Dur+19], ratios of linear polynomials [DEL20], or ratios of quadratic polynomials [DBP19] allows the derivatives at the knots to be arbitrary parameters.

The spline is strictly increasing if we take  $u_{k-1} < u_k$ ,  $x_{k-1} < x_k$ , and make sure the interpolation between knots is itself increasing. Depending on the flexibility on the interpolating function, more than one interpolations may exist; in practice we chose one that is guaranteed to be always increasing (see references above for details).

An advantage of splines is that they can be inverted analytically if the interpolating functions only contain low-degree polynomials. In this case, we compute  $u = h^{-1}(x)$  as follows: first, we use binary search to locate the interval  $(x_{k-1}, x_k)$  in which  $x$  lies; then, we analytically solve the resulting low-degree polynomial for  $u$ .

### 24.2.3 Coupling flows

In this section we describe coupling flows, which allow us to model dependencies between dimensions using arbitrary non-linear functions (such as deep neural networks). Consider a partition of the input  $\mathbf{u} \in \mathbb{R}^D$  into two subspaces,  $(\mathbf{u}^A, \mathbf{u}^B) \in \mathbb{R}^d \times \mathbb{R}^{D-d}$ , where  $d$  is an integer between 1 and  $D - 1$ . Assume a bijection  $\hat{f}(\cdot; \boldsymbol{\theta}) : \mathbb{R}^d \rightarrow \mathbb{R}^d$  parameterized by  $\boldsymbol{\theta}$  and acting on the subspace  $\mathbb{R}^d$ . We define

the function  $\mathbf{f} : \mathbb{R}^D \rightarrow \mathbb{R}^D$  given by  $\mathbf{x} = \mathbf{f}(\mathbf{u})$  as follows:

$$\mathbf{x}^A = \hat{\mathbf{f}}(\mathbf{u}^A; \Theta(\mathbf{u}^B)) \quad (24.18)$$

$$\mathbf{x}^B = \mathbf{u}^B. \quad (24.19)$$

See Figure 24.3 for an illustration. The function  $\mathbf{f}$  is called a **coupling layer** [DKB15; DSDB17], because it “couples”  $\mathbf{u}^A$  and  $\mathbf{u}^B$  together through  $\hat{\mathbf{f}}$  and  $\Theta$ . We refer to flows consisting of coupling layers as **coupling flows**.

The parameters of  $\hat{\mathbf{f}}$  are computed by  $\boldsymbol{\theta} = \Theta(\mathbf{u}^B)$ , where  $\Theta$  is an *arbitrary* function called the **conditioner**. Unlike affine flows, which mix dimensions linearly, and elementwise flows, which do not mix dimensions at all, coupling flows can mix dimensions with a flexible non-linear conditioner  $\Theta$ . In practice we often implement  $\Theta$  as a deep neural network; any architecture can be used, including MLPs, CNNs, ResNets, etc.

The coupling layer  $\mathbf{f}$  is *invertible*, and its inverse is given by  $\mathbf{u} = \mathbf{f}^{-1}(\mathbf{x})$ , where

$$\mathbf{u}^A = \hat{\mathbf{f}}^{-1}(\mathbf{x}^A; \Theta(\mathbf{x}^B)) \quad (24.20)$$

$$\mathbf{u}^B = \mathbf{x}^B. \quad (24.21)$$

That is,  $\mathbf{f}^{-1}$  is given by simply replacing  $\hat{\mathbf{f}}$  with  $\hat{\mathbf{f}}^{-1}$ . Because  $\mathbf{x}^B$  does not depend on  $\mathbf{u}^A$ , the Jacobian of  $\mathbf{f}$  is block triangular:

$$\mathbf{J}(\mathbf{f}) = \begin{pmatrix} \partial \mathbf{x}^A / \partial \mathbf{u}^A & \partial \mathbf{x}^A / \partial \mathbf{u}^B \\ \partial \mathbf{x}^B / \partial \mathbf{u}^A & \partial \mathbf{x}^B / \partial \mathbf{u}^B \end{pmatrix} = \begin{pmatrix} \mathbf{J}(\hat{\mathbf{f}}) & \partial \mathbf{x}^A / \partial \mathbf{u}^B \\ \mathbf{0} & \mathbf{I} \end{pmatrix}. \quad (24.22)$$

Thus,  $\det \mathbf{J}(\mathbf{f})$  is equal to  $\det \mathbf{J}(\hat{\mathbf{f}})$ .

We often define  $\hat{\mathbf{f}}$  to be an elementwise bijection, so that  $\hat{\mathbf{f}}^{-1}$  and  $\det \mathbf{J}(\hat{\mathbf{f}})$  are easy to compute. That is, we define:

$$\hat{\mathbf{f}}(\mathbf{u}^A; \boldsymbol{\theta}) = (h(u_1^A; \boldsymbol{\theta}_1), \dots, h(u_d^A; \boldsymbol{\theta}_d)), \quad (24.23)$$

where  $h(\cdot; \boldsymbol{\theta}_i)$  is a scalar bijection parameterized by  $\boldsymbol{\theta}_i$ . Any of the scalar bijections described in Section 24.2.2 can be used here. For example,  $h(\cdot; \boldsymbol{\theta}_i)$  can be an affine bijection with  $\boldsymbol{\theta}_i$  its scale and shift parameters (Section 24.2.2.1); or it can be a monotonic MLP with  $\boldsymbol{\theta}_i$  its weights and biases (Section 24.2.2.3); or it can be a monotonic spline with  $\boldsymbol{\theta}_i$  its knot coordinates (Section 24.2.2.5).

There are many ways to define the partition of  $\mathbf{u}$  into  $(\mathbf{u}^A, \mathbf{u}^B)$ . A simple way is just to partition  $\mathbf{u}$  into two halves. We can also exploit spatial structure in the partitioning. For example, if  $\mathbf{u}$  is an image, we can partition its pixels using a “checkerboard” pattern, where pixels in “black squares” are in  $\mathbf{u}^A$  and pixels in “white squares” are in  $\mathbf{u}^B$  [DSDB17]. Since only part of the input is transformed by each coupling layer, in practice we typically employ different partitions along a coupling flow, to ensure all variables get transformed and are given the opportunity to interact.

Finally, if  $\hat{\mathbf{f}}$  is an elementwise bijection, we can implement arbitrary partitions easily using a binary mask  $\mathbf{b}$  as follows:

$$\mathbf{x} = \mathbf{b} \odot \mathbf{u} + (1 - \mathbf{b}) \odot \hat{\mathbf{f}}(\mathbf{u}; \Theta(\mathbf{b} \odot \mathbf{u})), \quad (24.24)$$

where  $\odot$  denotes elementwise multiplication. A value of 0 in  $\mathbf{b}$  indicates that the corresponding element in  $\mathbf{u}$  is transformed (belongs to  $\mathbf{u}^A$ ); a value of 1 indicates that it remains unchanged (belongs to  $\mathbf{u}^B$ ).

1 **24.2.4 Autoregressive flows**

3 In this section we discuss **autoregressive flows**, which are flows composed of autoregressive bijections.  
4 Like coupling flows, autoregressive flows allow us to model dependencies between variables with  
5 arbitrary non-linear functions, such as deep neural networks.

6 Suppose the input  $\mathbf{u}$  contains  $D$  scalar elements, that is,  $\mathbf{u} = (u_1, \dots, u_D) \in \mathbb{R}^D$ . We define an  
7 **autoregressive bijection**  $\mathbf{f} : \mathbb{R}^D \rightarrow \mathbb{R}^D$ , its output denoted by  $\mathbf{x} = (x_1, \dots, x_D) \in \mathbb{R}^D$ , as follows:  
8

9

$$\underline{10} \quad x_i = h(u_i; \Theta_i(\mathbf{x}_{1:i-1})), \quad i = 1, \dots, D. \quad (24.25)$$

11

12 Each output  $x_i$  depends on the corresponding input  $u_i$  and all previous outputs  $\mathbf{x}_{1:i-1} = (x_1, \dots, x_{i-1})$ .  
13 The function  $h(\cdot; \boldsymbol{\theta}) : \mathbb{R} \rightarrow \mathbb{R}$  is a scalar bijection (for example, one of those described in Section 24.2.2),  
14 and is parameterized by  $\boldsymbol{\theta}$ . The function  $\Theta_i$  is a conditioner that outputs the parameters  $\boldsymbol{\theta}_i$  that  
15 yield  $x_i$ , given all previous outputs  $\mathbf{x}_{1:i-1}$ . Like in coupling flows,  $\Theta_i$  can be an arbitrary non-linear  
16 function, and is often parameterized as a deep neural network.

17 Because  $h$  is invertible,  $\mathbf{f}$  is also invertible, and its inverse is given by:

18

$$\underline{19} \quad u_i = h^{-1}(x_i; \Theta_i(\mathbf{x}_{1:i-1})), \quad i = 1, \dots, D. \quad (24.26)$$

20

21 An important property of  $\mathbf{f}$  is that each output  $x_i$  depends on  $\mathbf{u}_{1:i} = (u_1, \dots, u_i)$ , but not on  
22  $\mathbf{u}_{i+1:D} = (u_{i+1}, \dots, u_D)$ ; as a result, the partial derivative  $\partial x_i / \partial u_j$  is identically zero whenever  $j > i$ .  
23 Therefore, the Jacobian matrix  $\mathbf{J}(\mathbf{f})$  is triangular, and its determinant is simply the product of its  
24 diagonal entries:

25

$$\underline{26} \quad \det \mathbf{J}(\mathbf{f}) = \prod_{i=1}^D \frac{\partial x_i}{\partial u_i} = \prod_{i=1}^D \frac{dh}{du_i}. \quad (24.27)$$

27

28 In other words, the autoregressive structure of  $\mathbf{f}$  leads to a Jacobian determinant that can be  
29 computed efficiently in  $O(D)$  time.

30 Although invertible, autoregressive bijections are computationally asymmetric: evaluating  $\mathbf{f}$  is  
31 inherently sequential, whereas evaluating  $\mathbf{f}^{-1}$  is inherently parallel. That is because we need  $\mathbf{x}_{1:i-1}$  to  
32 compute  $x_i$ ; therefore, computing the components of  $\mathbf{x}$  must be done sequentially, by first computing  
33  $x_1$ , then using it to compute  $x_2$ , then using  $x_1$  and  $x_2$  to compute  $x_3$ , and so on. On the other hand,  
34 computing the inverse can be done in parallel for each  $u_i$ , since  $\mathbf{u}$  does not appear on the right-hand  
35 side of Equation (24.26). Hence, in practice it is often faster to compute  $\mathbf{f}^{-1}$  than to compute  $\mathbf{f}$ ,  
36 assuming  $h$  and  $h^{-1}$  have similar computational cost.

37

38 **24.2.4.1 Affine autoregressive flows**

39 For a concrete example, we can take  $h$  to be an affine scalar bijection (Section 24.2.2.1) parameterized  
40 by a log scale  $\alpha$  and a bias  $\mu$ . Such autoregressive flows are known as **affine autoregressive flows**.  
41 The parameters of the  $i$ 'th component,  $\alpha_i$  and  $\mu_i$ , are functions of  $\mathbf{x}_{1:i-1}$ , so  $\mathbf{f}$  takes the following  
42 form:

43

$$\underline{44} \quad x_i = u_i \exp(\alpha_i(\mathbf{x}_{1:i-1})) + \mu_i(\mathbf{x}_{1:i-1}). \quad (24.28)$$

45

46

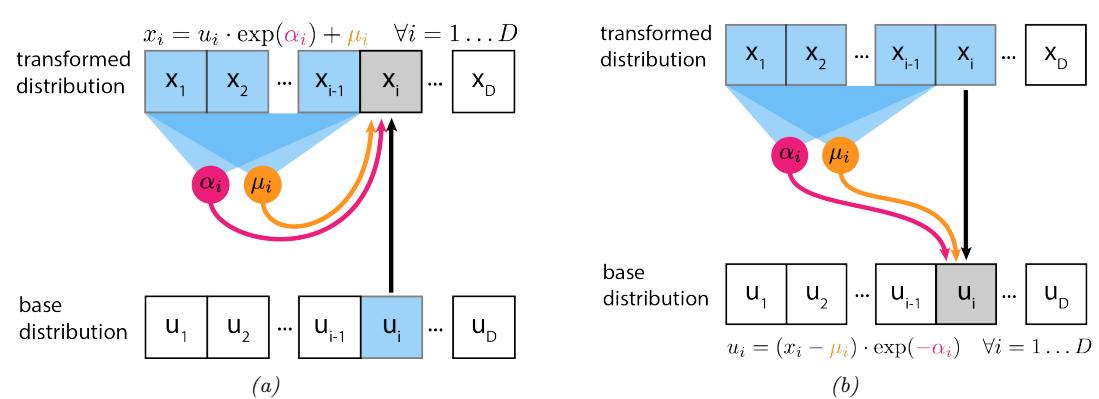


Figure 24.4: (a) Affine autoregressive flow with one layer. In this figure,  $\mathbf{u}$  is the input to the flow (sample from the base distribution) and  $\mathbf{x}$  is its output (sample from the transformed distribution). (b) Inverse of the above. From [Jan18]. Used with kind permission of Eric Jang.

This is illustrated in Figure 24.4(a). We can invert this by

$$u_i = (x_i - \mu_i(\mathbf{x}_{1:i-1})) \exp(-\alpha_i(\mathbf{x}_{1:i-1})). \quad (24.29)$$

This is illustrated in Figure 24.4(b). Finally, we can calculate the log absolute Jacobian determinant by

$$\log |\det \mathbf{J}(\mathbf{f})| = \log \left| \prod_{i=1}^D \exp(\alpha_i(\mathbf{x}_{1:i-1})) \right| = \sum_{i=1}^D \alpha_i(\mathbf{x}_{1:i-1}). \quad (24.30)$$

Let us look at an example of an affine autoregressive flow on a 2d density estimation problem. Consider an affine autoregressive flow  $\mathbf{x} = (x_1, x_2) = \mathbf{f}(\mathbf{u})$ , where  $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  and  $\mathbf{f}$  is a single autoregressive bijection. Since  $x_1$  is an affine transformation of  $u_1 \sim \mathcal{N}(0, 1)$ , it is Gaussian with mean  $\mu_1$  and standard deviation  $\sigma_1 = \exp \alpha_1$ . Similarly, if we consider  $x_1$  fixed,  $x_2$  is an affine transformation of  $u_2 \sim \mathcal{N}(0, 1)$ , so it is *conditionally* Gaussian with mean  $\mu_2(x_1)$  and standard deviation  $\sigma_2(x_1) = \exp \alpha_2(x_1)$ . Thus, a single affine autoregressive bijection will always produce a distribution with Gaussian conditionals, that is, a distribution of the following form:

$$p(x_1, x_2) = p(x_1) p(x_2|x_1) = \mathcal{N}(x_1|\mu_1, \sigma_1^2) \mathcal{N}(x_2|\mu_2(x_1), \sigma_2(x_1)^2) \quad (24.31)$$

This result generalizes to an arbitrary number of dimensions  $D$ .

A single affine bijection is not very powerful, regardless of how flexible the functions  $\alpha_2(x_1)$  and  $\mu_2(x_1)$  are. For example, suppose we want to fit the cross-shaped density shown in Figure 24.5(a) with such a flow. The resulting maximum-likelihood fit is shown in Figure 24.5(b). The red contours show the predictive distribution,  $\hat{p}(\mathbf{x})$ , which clearly fails to capture the true distribution. The green dots show transformed versions of the data samples,  $p(\mathbf{u})$ ; we see that this is far from the Gaussian base distribution.

Fortunately, we can obtain a better fit by composing multiple autoregressive bijections (layers), and reversing the order of the variables after each layer. For example, Figure 24.5(c) shows the

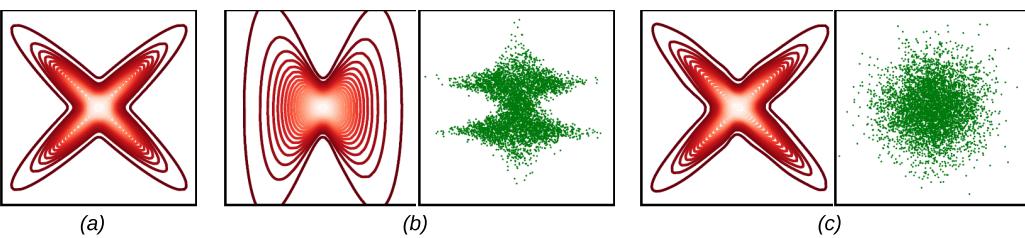


Figure 24.5: Density estimation with affine autoregressive flows, using a Gaussian base distribution. (a) True density. (b) Estimated density using a single autoregressive layer with ordering  $(x_1, x_2)$ . On the left (contour plot) we show  $p(\mathbf{x})$ . On the right (green dots) we show samples of  $\mathbf{u} = \mathbf{f}^{-1}(\mathbf{x})$ , where  $\mathbf{x}$  is sampled from the true density. (c) Same as (b), but using 5 autoregressive layers and reversing the variable ordering after each layer. Adapted from Figure 1 of [PPM17]. Used with kind permission of Iain Murray.

results of an affine autoregressive flow with 5 layers applied to the same problem. The red contours show that we have matched the empirical distribution, and the green dots show we have matched the Gaussian base distribution.

Note that another way to obtain a better fit is to replace the affine bijection  $h$  with a more flexible one, such as a monotonic MLP (Section 24.2.2.3) or a monotonic spline (Section 24.2.2.5).

#### 24.2.4.2 Masked autoregressive flows

As we have seen, the conditioners  $\Theta_i$  can be arbitrary non-linear functions. The most straightforward way to parameterize them is separately for each  $i$ , for example by using  $D$  separate neural networks. However, this can be parameter-inefficient for large  $D$ .

In practice, we often share parameters between conditioners by combining them into a single model  $\Theta$  that takes in  $\mathbf{x}$  and outputs  $(\theta_1, \dots, \theta_D)$ . For the bijection to remain autoregressive, we must constrain  $\Theta$  so that  $\theta_i$  depends only on  $\mathbf{x}_{1:i-1}$  and not on  $\mathbf{x}_{i:D}$ . One way to achieve this is to start with an arbitrary neural network (an MLP, a CNN, a ResNet, etc.), and drop connections (for example, by zeroing out weights) until  $\theta_i$  is only a function of  $\mathbf{x}_{1:i-1}$ .

An example of this approach is the **masked autoregressive flow (MAF)** model of [PPM17]. This model is an affine autoregressive flow combined with permutation layers, as we described in Section 24.2.4.1. MAF implements the combined conditioner  $\Theta$  as follows: it starts with an MLP, and then multiplies (elementwise) the weight matrix of each layer with a binary mask of the same size (different masks are used for different layers). The masks are constructed using the method of [Ger+15]. This ensures that all computational paths from  $x_j$  to  $\theta_i$  are zeroed out whenever  $j \geq i$ , effectively making  $\theta_i$  only a function of  $\mathbf{x}_{1:i-1}$ . Still, evaluating the masked conditioner  $\Theta$  has the same computational cost as evaluating the original (unmasked) MLP.

The key advantage of MAF (and of related models) is that, given  $\mathbf{x}$ , all parameters  $(\theta_1, \dots, \theta_D)$  can be computed efficiently with one neural network evaluation, so the computation of the inverse  $\mathbf{f}^{-1}$  is fast. Thus, we can efficiently evaluate the probability density of the flow model for arbitrary datapoints. However, in order to compute  $\mathbf{f}$ , the conditioner  $\Theta$  must be called a total of  $D$  times, since not all entries of  $\mathbf{x}$  are available to start with. Thus, generating new samples from the flow is  $D$  times more expensive than evaluating its probability density function. This makes MAF suitable

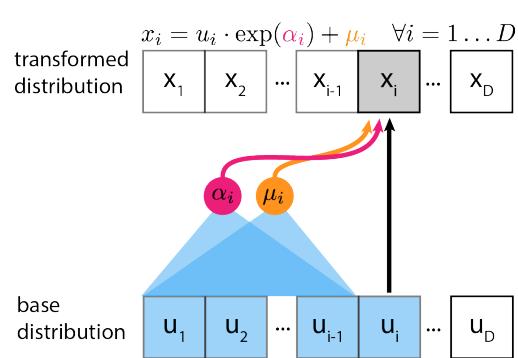


Figure 24.6: Inverse autoregressive flow that uses affine scalar bijections. In this figure,  $\mathbf{u}$  is the input to the flow (sample from the base distribution) and  $\mathbf{x}$  is its output (sample from the transformed distribution). From [Jan18]. Used with kind permission of Eric Jang.

for density estimation, but less so for data generation.

#### 24.2.4.3 Inverse autoregressive flows

As we have seen, the parameters  $\theta_i$  that yield the  $i$ 'th output  $x_i$  are functions of the previous outputs  $\mathbf{x}_{1:i-1}$ . This ensures that the Jacobian  $\mathbf{J}(\mathbf{f})$  is triangular, and so its determinant is efficient to compute.

However, there is another possibility: we can make  $\theta_i$  a function of the previous *inputs* instead, that is, a function of  $\mathbf{u}_{1:i-1}$ . This leads to the following bijection, which is known as **inverse autoregressive**:

$$x_i = h(u_i; \Theta_i(\mathbf{u}_{1:i-1})), \quad i = 1, \dots, D. \quad (24.32)$$

Like its autoregressive counterpart, this bijection has a triangular Jacobian whose determinant is also given by  $\det \mathbf{J}(\mathbf{f}) = \prod_{i=1}^D \frac{dh}{du_i}$ . Figure 24.6 illustrates an inverse autoregressive flow, for the case where  $h$  is affine.

To see why this bijection is called “inverse autoregressive”, compare Equation (24.32) with Equation (24.26). The two formulas differ only notationally: we can get from one to the other by swapping  $\mathbf{u}$  with  $\mathbf{x}$  and  $h$  with  $h^{-1}$ . In other words, the inverse autoregressive bijection corresponds to a direct parameterization of the inverse of an autoregressive bijection.

Since inverse autoregressive bijections swap the forward and inverse directions of their autoregressive counterparts, they also swap their computational properties. This means that the forward direction  $\mathbf{f}$  of an inverse autoregressive flow is inherently parallel and therefore fast, whereas its inverse direction  $\mathbf{f}^{-1}$  is inherently sequential and therefore slow.

An example of an inverse autoregressive flow is their namesake **IAF** model of [Kin+16]. IAF uses affine scalar bijections, masked conditioners and permutation layers, so it is precisely the inverse of the MAF model described in Section 24.2.4.2. Using IAF, we can generate  $\mathbf{u}$  in parallel from the base distribution (using, for example, a diagonal Gaussian), and then sample each element of  $\mathbf{x}$  in parallel. However, evaluating  $p(\mathbf{x})$  for an arbitrary datapoint  $\mathbf{x}$  is slow, because we have to evaluate each element of  $\mathbf{u}$  sequentially. Fortunately, evaluating the likelihood of samples generated from IAF

<sup>1</sup> (as opposed to externally provided samples) incurs no additional cost, since in this case the  $u_i$  terms  
<sup>2</sup> will already have been computed.

<sup>3</sup> Although not so suitable for density estimation or maximum-likelihood training, IAFs are well-  
<sup>4</sup> suited for parameterizing variational posteriors in variational inference. This is because in order to  
<sup>5</sup> estimate the variational lower bound (ELBO), we only need samples from the variational posterior  
<sup>6</sup> and their associated probability densities, both of which are efficient to obtain. See Section 24.1.3.2  
<sup>7</sup> for details.

<sup>8</sup> Another useful application of IAFs is training them to mimic models whose probability density is  
<sup>9</sup> fast to evaluate but which are slow to sample from. A notable example is the **parallel wavenet**  
<sup>10</sup> model of [Oor+18]. This model is an IAF  $p_s$  that it trained to mimic a pretrained wavenet model  $p_t$   
<sup>11</sup> by minimizing the KL divergence  $D_{\text{KL}}(p_s \| p_t)$ . This KL can be easily estimated by first sampling  
<sup>12</sup> from  $p_s$  and then evaluating  $\log p_s$  and  $\log p_t$  at those samples, operations which are all efficient for  
<sup>13</sup> these models. After training, we obtain an IAF that can generate audio of similar quality as the  
<sup>14</sup> original wavenet, but can do so much faster.

<sup>16</sup>

#### <sup>17</sup> 24.2.4.4 Connection with autoregressive models

<sup>18</sup> Autoregressive flows can be thought of as generalizing autoregressive models of continuous random  
<sup>19</sup> variables, discussed in Section 23.1. Specifically, any continuous autoregressive model can be  
<sup>20</sup> reparameterized as a one-layer autoregressive flow, as we describe below.

<sup>21</sup> Consider a general autoregressive model over a continuous random variable  $\mathbf{x} = (x_1, \dots, x_D) \in \mathbb{R}^D$   
<sup>22</sup> written as

$$\begin{aligned} \mathbf{x} &= \prod_{i=1}^D p_i(x_i | \boldsymbol{\theta}_i) \quad \text{where } \boldsymbol{\theta}_i = \Theta_i(\mathbf{x}_{1:i-1}). \end{aligned} \tag{24.33}$$

<sup>23</sup> In the above expression,  $p_i(x_i | \boldsymbol{\theta}_i)$  is the  $i$ -th conditional distribution of the autoregressive model,  
<sup>24</sup> whose parameters  $\boldsymbol{\theta}_i$  are arbitrary functions of the previous variables  $\mathbf{x}_{1:i-1}$ . For example,  $p_i(x_i | \boldsymbol{\theta}_i)$   
<sup>25</sup> can be a mixture of one-dimensional Gaussian distributions, with  $\boldsymbol{\theta}_i$  representing the collection of its  
<sup>26</sup> means, variances and mixing coefficients.

<sup>27</sup> Now consider sampling a vector  $\mathbf{x}$  from the autoregressive model, which can be done by sampling  
<sup>28</sup> one element at a time as follows:

$$x_i \sim p_i(x_i | \Theta_i(\mathbf{x}_{1:i-1})) \quad \text{for } i = 1, \dots, D. \tag{24.34}$$

<sup>29</sup> Each conditional can be sampled from using inverse transform sampling (Section 11.3.1). Let  $U(0, 1)$   
<sup>30</sup> be the uniform distribution on the interval  $[0, 1]$ , and let  $\text{CDF}_i(x_i | \boldsymbol{\theta}_i)$  be the cumulative distribution  
<sup>31</sup> function of the  $i$ -th conditional. Sampling can be written as:

$$x_i = \text{CDF}_i^{-1}(u_i | \Theta_i(\mathbf{x}_{1:i-1})) \quad \text{where } u_i \sim U(0, 1). \tag{24.35}$$

<sup>32</sup> Comparing the above expression with the definition of an autoregressive bijection in Equation (24.25),  
<sup>33</sup> we see that the autoregressive model has been expressed as a one-layer autoregressive flow whose base  
<sup>34</sup> distribution is uniform on  $[0, 1]^D$  and whose scalar bijections correspond to the inverse conditional  
<sup>35</sup> CDFs. Viewing autoregressive models as flows this way has an important advantage, namely that it  
<sup>36</sup> allows us to increase the flexibility of an autoregressive model by composing multiple instances of it  
<sup>37</sup> in a flow, without sacrificing the overall tractability.

<sup>38</sup>

## 24.2.5 Residual flows

A residual network is a composition of **residual connections**, which are functions of the form  $\mathbf{f}(\mathbf{u}) = \mathbf{u} + \mathbf{F}(\mathbf{u})$ . The function  $\mathbf{F} : \mathbb{R}^D \rightarrow \mathbb{R}^D$  is called the **residual block**, and it computes the difference between the output and the input,  $\mathbf{f}(\mathbf{u}) - \mathbf{u}$ .

Under certain conditions on  $\mathbf{F}$ , the residual connection  $\mathbf{f}$  becomes invertible. We will refer to flows composed of invertible residual connections as **residual flows**. In the following, we describe two ways the residual block  $\mathbf{F}$  can be constrained so that the residual connection  $\mathbf{f}$  is invertible.

### 24.2.5.1 Contractive residual blocks

One way to ensure the residual connection is invertible is to choose the residual block to be a contraction. A contraction is a function  $\mathbf{F}$  whose Lipschitz constant is less than 1; that is, there exists  $0 \leq L < 1$  such that for all  $\mathbf{u}_1$  and  $\mathbf{u}_2$  we have:

$$\|\mathbf{F}(\mathbf{u}_1) - \mathbf{F}(\mathbf{u}_2)\| \leq L\|\mathbf{u}_1 - \mathbf{u}_2\|. \quad (24.36)$$

The invertibility of  $\mathbf{f}(\mathbf{u}) = \mathbf{u} + \mathbf{F}(\mathbf{u})$  can be shown as follows. Consider the mapping  $\mathbf{g}(\mathbf{u}) = \mathbf{x} - \mathbf{F}(\mathbf{u})$ . Because  $\mathbf{F}$  is a contraction,  $\mathbf{g}$  is also a contraction. So, by Banach's fixed-point theorem,  $\mathbf{g}$  has a unique fixed point  $\mathbf{u}_*$ . Hence we have

$$\mathbf{u}_* = \mathbf{x} - \mathbf{F}(\mathbf{u}_*) \quad (24.37)$$

$$\Rightarrow \mathbf{u}_* + \mathbf{F}(\mathbf{u}_*) = \mathbf{x} \quad (24.38)$$

$$\Rightarrow \mathbf{f}(\mathbf{u}_*) = \mathbf{x}. \quad (24.39)$$

Because  $\mathbf{u}_*$  is unique, it follows that  $\mathbf{u}_* = \mathbf{f}^{-1}(\mathbf{x})$ .

An example of a residual flow with contractive residual blocks is the **iResNet** model of [Beh+19]. The residual blocks of iResNet are convolutional neural networks, that is, compositions of convolutional layers with non-linear activation functions. Because the Lipschitz constant of a composition is less or equal to the product of the Lipschitz constants of the individual functions, it is enough to ensure the convolutions are contractive, and to use increasing activation functions with slope less or equal to 1. The iResNet model ensures the convolutions are contractive by applying spectral normalization to their weights [Miy+18a].

In general, there is no analytical expression for the inverse  $\mathbf{f}^{-1}$ . However, we can approximate  $\mathbf{f}^{-1}(\mathbf{x})$  using the following iterative procedure:

$$\mathbf{u}_n = \mathbf{g}(\mathbf{u}_{n-1}) = \mathbf{x} - \mathbf{F}(\mathbf{u}_{n-1}). \quad (24.40)$$

Banach's fixed-point theorem guarantees that the sequence  $\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2, \dots$  will converge to  $\mathbf{u}_* = \mathbf{f}^{-1}(\mathbf{x})$  for any choice of  $\mathbf{u}_0$ , and it will do so at a rate of  $O(L^n)$ , where  $L$  is the Lipschitz constant of  $\mathbf{g}$  (which is the same as the Lipschitz constant of  $\mathbf{F}$ ). In practice, it is convenient to choose  $\mathbf{u}_0 = \mathbf{x}$ .

In addition, there is no analytical expression for the Jacobian determinant, whose exact computation costs  $O(D^3)$ . However, there is a computationally efficient stochastic estimator of the log Jacobian determinant. The idea is to express the log Jacobian determinant as a power series. Using the fact that  $\mathbf{f}(\mathbf{x}) = \mathbf{x} + \mathbf{F}(\mathbf{x})$ , we have

$$\log |\det \mathbf{J}(\mathbf{f})| = \log |\det(\mathbf{I} + \mathbf{J}(\mathbf{F}))| = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \text{tr}[\mathbf{J}(\mathbf{F})^k]. \quad (24.41)$$

<sup>1</sup> This power series converges when the matrix norm of  $\mathbf{J}(\mathbf{F})$  is less than 1, which here is guaranteed  
<sup>2</sup> exactly because  $\mathbf{F}$  is a contraction. The trace of  $\mathbf{J}(\mathbf{F})^k$  can be efficiently approximated using  
<sup>3</sup> Jacobian-vector products via the **Hutchinson trace estimator** [Ski89; Hut89; Mey+21]:  
<sup>4</sup>

<sup>5</sup>  $\text{tr}[\mathbf{J}(\mathbf{F})^k] \approx \mathbf{v}^\top \mathbf{J}(\mathbf{F})^k \mathbf{v},$  (24.42)  
<sup>6</sup>

<sup>7</sup> where  $\mathbf{v}$  is a sample from a distribution with zero mean and unit covariance, such as  $\mathcal{N}(\mathbf{0}, \mathbf{I}).$   
<sup>8</sup> Finally, the infinite series can be approximated by a finite one either by truncation [Beh+19], which  
<sup>9</sup> unfortunately yields a biased estimator, or by employing the **Russian-roulette estimator** [Che+19],  
<sup>10</sup> which is unbiased.

<sup>11</sup>

#### <sup>12</sup> 24.2.5.2 Residual blocks with low-rank Jacobian

<sup>13</sup>

<sup>14</sup> There is an efficient way of computing the determinant of a matrix which is a low-rank perturbation  
<sup>15</sup> of an identity matrix. Suppose  $\mathbf{A}$  and  $\mathbf{B}$  are matrices, where  $\mathbf{A}$  is  $D \times M$  and  $\mathbf{B}$  is  $M \times D.$  The  
<sup>16</sup> following formula is known as the **Weinstein–Aronszajn identity**<sup>2</sup>, and is a special case of the  
<sup>17</sup> more general **matrix determinant lemma**:

<sup>18</sup>  $\det(\mathbf{I}_D + \mathbf{AB}) = \det(\mathbf{I}_M + \mathbf{BA}).$  (24.43)  
<sup>19</sup>

<sup>20</sup> We write  $\mathbf{I}_D$  and  $\mathbf{I}_M$  for the  $D \times D$  and  $M \times M$  identity matrices respectively. The significance of  
<sup>21</sup> this formula is that it turns a  $D \times D$  determinant that costs  $O(D^3)$  into an  $M \times M$  determinant  
<sup>22</sup> that costs  $O(M^3).$  If  $M$  is smaller than  $D,$  this saves computation.

<sup>23</sup> With some restrictions on the residual block  $\mathbf{F} : \mathbb{R}^D \rightarrow \mathbb{R}^D,$  we can apply this formula to compute  
<sup>24</sup> the determinant of a residual connection efficiently. The trick is to create a bottleneck inside  $\mathbf{F}.$  We  
<sup>25</sup> do that by defining  $\mathbf{F} = \mathbf{F}_2 \circ \mathbf{F}_1,$  where  $\mathbf{F}_1 : \mathbb{R}^D \rightarrow \mathbb{R}^M,$   $\mathbf{F}_2 : \mathbb{R}^M \rightarrow \mathbb{R}^D$  and  $M \ll D.$  The chain  
<sup>26</sup> rule gives  $\mathbf{J}(\mathbf{F}) = \mathbf{J}(\mathbf{F}_2)\mathbf{J}(\mathbf{F}_1),$  where  $\mathbf{J}(\mathbf{F}_2)$  is  $D \times M$  and  $\mathbf{J}(\mathbf{F}_1)$  is  $M \times D.$  Now we can apply our  
<sup>27</sup> determinant formula as follows:

<sup>28</sup>

<sup>29</sup>  $\det \mathbf{J}(\mathbf{f}) = \det(\mathbf{I}_D + \mathbf{J}(\mathbf{F})) = \det(\mathbf{I}_D + \mathbf{J}(\mathbf{F}_2)\mathbf{J}(\mathbf{F}_1)) = \det(\mathbf{I}_M + \mathbf{J}(\mathbf{F}_1)\mathbf{J}(\mathbf{F}_2)).$  (24.44)  
<sup>30</sup>

<sup>31</sup> Since the final determinant costs  $O(M^3),$  we can make the Jacobian determinant efficient by reducing  
<sup>32</sup>  $M,$  that is, by narrowing the bottleneck.

<sup>33</sup> An example of the above is the **planar flow** of [RM15b]. In this model, each residual block is an  
<sup>34</sup> MLP with one hidden layer and one hidden unit. That is,

<sup>35</sup>  $\mathbf{f}(\mathbf{u}) = \mathbf{u} + \mathbf{v}\sigma(\mathbf{w}^\top \mathbf{u} + b),$  (24.45)  
<sup>36</sup>

<sup>37</sup> where  $\mathbf{v} \in \mathbb{R}^D,$   $\mathbf{w} \in \mathbb{R}^D$  and  $b \in \mathbb{R}$  are the parameters, and  $\sigma$  is the activation function. The residual  
<sup>38</sup> block is the composition of  $\mathbf{F}_1(\mathbf{u}) = \mathbf{w}^\top \mathbf{u} + b$  and  $\mathbf{F}_2(z) = \mathbf{v}\sigma(z),$  so  $M = 1.$  Their Jacobians  
<sup>39</sup> are  $\mathbf{J}(\mathbf{F}_1)(\mathbf{u}) = \mathbf{w}^\top$  and  $\mathbf{J}(\mathbf{F}_2)(z) = \mathbf{v}\sigma'(z).$  Substituting these in the formula for the Jacobian  
<sup>40</sup> determinant we obtain:

<sup>41</sup>

<sup>42</sup>  $\det \mathbf{J}(\mathbf{f})(\mathbf{u}) = 1 + \mathbf{w}^\top \mathbf{v}\sigma'(\mathbf{w}^\top \mathbf{u} + b),$  (24.46)  
<sup>43</sup>

<sup>44</sup> which can be computed efficiently in  $O(D).$  Other examples include the **circular flow** of [RM15b]  
<sup>45</sup> and the **Sylvester flow** of [Ber+18].

<sup>46</sup> 2. See [https://en.wikipedia.org/wiki/Weinstein–Aronszajn\\_identity](https://en.wikipedia.org/wiki/Weinstein–Aronszajn_identity).

<sup>47</sup>

This technique gives an efficient way of computing determinants of residual connections with bottlenecks, but in general there is no guarantee that such functions are invertible. This means that invertibility must be satisfied on a case-by-case basis. For example, the planar flow is invertible when  $\sigma$  is the hyperbolic tangent and  $\mathbf{w}^\top \mathbf{v} > -1$ , but otherwise it may not be.

### 24.2.6 Continuous-time flows

So far we have discussed flows that consist of a sequence of bijections  $\mathbf{f}_1, \dots, \mathbf{f}_N$ . Starting from some input  $\mathbf{x}_0 = \mathbf{u}$ , this creates a sequence of outputs  $\mathbf{x}_1, \dots, \mathbf{x}_N$  where  $\mathbf{x}_n = \mathbf{f}_n(\mathbf{x}_{n-1})$ . However, we can also have flows where the input is transformed into the final output in a continuous way. That is, we start from  $\mathbf{x}_0 = \mathbf{x}(0)$ , create a continuously-indexed sequence  $\mathbf{x}(t)$  for  $t \in [0, T]$  with some fixed  $T$ , and take  $\mathbf{x}(T)$  to be the final output. Thinking of  $t$  as analogous to time, we refer to these as **continuous-time flows**.

The sequence  $\mathbf{x}(t)$  is defined as the solution to a first-order ordinary differential equation (ODE) of the form:

$$\frac{d\mathbf{x}}{dt}(t) = \mathbf{F}(\mathbf{x}(t), t). \quad (24.47)$$

The function  $\mathbf{F} : \mathbb{R}^D \times [0, T] \rightarrow \mathbb{R}^D$  is a time-dependent vector field that parameterizes the ODE. If we think of  $\mathbf{x}(t)$  as the position of a particle in  $D$  dimensions, the vector  $\mathbf{F}(\mathbf{x}(t), t)$  determines the particle's velocity at time  $t$ .

The flow (for time  $T$ ) is a function  $\mathbf{f} : \mathbb{R}^D \rightarrow \mathbb{R}^D$  that takes in an input  $\mathbf{x}_0$ , solves the ODE with initial condition  $\mathbf{x}(0) = \mathbf{x}_0$ , and returns  $\mathbf{x}(T)$ . The function  $\mathbf{f}$  is a well-defined bijection if the solution to the ODE exists for all  $t \in [0, T]$  and is unique. These conditions are not generally satisfied for arbitrary  $\mathbf{F}$ , but they are if  $\mathbf{F}(\cdot, t)$  is Lipschitz continuous with a Lipschitz constant that does not depend on  $t$ . That is,  $\mathbf{f}$  is a well-defined bijection if there exists a constant  $L$  such that for all  $\mathbf{x}_1, \mathbf{x}_2$  and  $t \in [0, T]$  we have:

$$\|\mathbf{F}(\mathbf{x}_1, t) - \mathbf{F}(\mathbf{x}_2, t)\| \leq L \|\mathbf{x}_1 - \mathbf{x}_2\|. \quad (24.48)$$

This result is a consequence of the **Picard–Lindelöf theorem** for ODEs.<sup>3</sup> In practice, we can parameterize  $\mathbf{F}$  using any choice of model, provided the Lipschitz condition is met.

Usually the ODE cannot be solved analytically, but we can solve it approximately by discretizing it. A simple example is **Euler's method**, which corresponds to the following discretization for some small step size  $\epsilon > 0$ :

$$\mathbf{x}(t + \epsilon) = \mathbf{x}(t) + \epsilon \mathbf{F}(\mathbf{x}(t), t). \quad (24.49)$$

This is equivalent to a residual connection with residual block  $\epsilon \mathbf{F}(\cdot, t)$ , so the ODE solver can be thought of as a deep residual network with  $O(T/\epsilon)$  layers. A smaller step size leads to a more accurate solution, but also to more computation. There are several other solution methods varying in accuracy and sophistication, such as those in the broader Runge–Kutta family, some of which use adaptive step sizes.

The inverse of  $\mathbf{f}$  can be easily computed by solving the ODE in reverse. That is, to compute  $\mathbf{f}^{-1}(\mathbf{x}_T)$  we solve the ODE with initial condition  $\mathbf{x}(T) = \mathbf{x}_T$  and return  $\mathbf{x}(0)$ . Unlike some other

<sup>3</sup>. See [https://en.wikipedia.org/wiki/Picard-Lindel%C3%B6f\\_theorem](https://en.wikipedia.org/wiki/Picard-Lindel%C3%B6f_theorem)

1 flows (such as autoregressive flows) which are more expensive to compute in one direction than in  
2 the other, continuous-time flows require the same amount of computation in either direction.  
3

4 In general, there is no analytical expression for the Jacobian determinant of  $\mathbf{f}$ . However, we can  
5 express it as the solution to a separate ODE, which we can then solve numerically. First, we define  
6  $\mathbf{f}_t : \mathbb{R}^D \rightarrow \mathbb{R}^D$  to be the flow for time  $t$ , that is, the function that takes  $\mathbf{x}_0$ , solves the ODE with  
7 initial condition  $\mathbf{x}(0) = \mathbf{x}_0$  and returns  $\mathbf{x}(t)$ . Clearly,  $\mathbf{f}_0$  is the identity function and  $\mathbf{f}_T = \mathbf{f}$ . Let us  
8 define  $L(t) = \log |\det \mathbf{J}(\mathbf{f}_t)(\mathbf{x}_0)|$ . Because  $\mathbf{f}_0$  is the identity function,  $L(0) = 0$ , and because  $\mathbf{f}_T = \mathbf{f}$ ,  
9  $L(T)$  gives the Jacobian determinant of  $\mathbf{f}$  that we are interested in. It can be shown that  $L$  satisfies  
10 the following ODE:

$$\frac{dL}{dt}(t) = \text{tr}[\mathbf{J}(\mathbf{F}(\cdot, t))(\mathbf{x}(t))]. \quad (24.50)$$

11 That is, the rate of change of  $L$  at time  $t$  is equal to the Jacobian trace of  $\mathbf{F}(\cdot, t)$  evaluated at  $\mathbf{x}(t)$ . So  
12 we can compute  $L(T)$  by solving the above ODE with initial condition  $L(0) = 0$ . Moreover, we can  
13 compute  $\mathbf{x}(T)$  and  $L(T)$  simultaneously, by combining their two ODEs into a single ODE operating  
14 on the extended space  $(\mathbf{x}, L)$ .

15 An example of a continuous-time flow is the **Neural ODE** model of [Che+18c], which uses a  
16 neural network to parameterize  $\mathbf{F}$ . To avoid backpropagating gradients through the ODE solver,  
17 which can be computationally demanding, they use the **adjoint sensitivity method** to express the  
18 time evolution of the gradient with respect to  $\mathbf{x}(t)$  as a separate ODE. Solving this ODE gives the  
19 required gradients, and can be thought of as the continuous-time analogue of backpropagation.

20 Another example is the **FFJORD** model of [Gra+18]. This is similar to the Neural ODE model,  
21 except that it uses the Hutchinson trace estimator to approximate the Jacobian trace of  $\mathbf{F}(\cdot, t)$ .  
22 This usage of the Hutchinson trace estimator is analogous to that in contractive residual flows  
23 (Section 24.2.5.1), and it speeds up computation in exchange for a stochastic (but unbiased) estimate.  
24

## 25 24.3 Applications

26 In this section, we highlight some applications of flows for canonical probabilistic machine learning  
27 tasks.  
28

### 29 24.3.1 Density estimation

30 Flow models allow exact density computation and can be used to fit multi-modal densities to observed  
31 data. An early example is Gaussianization [CG00] who applied this idea to fit low-dimensional  
32 densities. Tabak and Vanden-Eijnden [TVE10] and Tabak and Turner [TT13] introduced the modern  
33 idea of flows (including the term ‘normalizing flows’), describing a flow as a composition of simpler  
34 maps. Deep density models [RA13] was one of the first to use neural networks for flows to parametrize  
35 high-dimensional densities. There has been a rich line of follow-up work including **NICE** [DKB15]  
36 and **Real NVP** [DSDB17]. (NVP stands for “non-volume preserving”, which refers to the fact that  
37 the Jacobian of the transform is not unity.) Masked autoregressive flows (Section 24.2.4.2) further  
38 improved performance on unconditional and conditional density estimation tasks.  
39

40 Flows can be used for *hybrid models* which model the joint density of inputs and targets  $p(\mathbf{x}, y)$ , as  
41 opposed to discriminative classification models which just model the conditional  $p(y|\mathbf{x})$  and density  
42 models which just model the marginal  $p(\mathbf{x})$ . Nalisnick et al. [Nal+19b] proposed a flow-based hybrid  
43

model using invertible mappings for representation learning and showed that the joint density  $p(\mathbf{x}, \mathbf{y})$  can be computed efficiently, which can be useful for downstream tasks such as anomaly detection, semi-supervised learning and selective classification. Flow-based hybrid models are memory-efficient since most of the parameters are in the invertible representation which are shared between the discriminative and generative models; furthermore, the density  $p(\mathbf{x}, \mathbf{y})$  can be computed in a single forward pass leading to computational savings. Residual flows [Che+19] use invertible residual mappings [Beh+19] for hybrid modeling which further improves performance. Flows have also been used to fit densities to embeddings [Zha+20b; CZG20] for anomaly detection tasks.

### 24.3.2 Generative Modeling

Another task is generation, which involves generating novel samples from a fitted model  $p^*(\mathbf{x})$ . Generation is a popular downstream task for normalizing flows, which have been applied for different data modalities including images, video, audio, text and structured objects such as graphs and point clouds. Images are arguably the most popular modality for deep generative models: GLOW [KD18b] was one of the first flow-based models to generate compelling high-dimensional images, and has been extended to video to produce RGB frames [Kum+19b]; residual flows [Che+19] have also been shown to produce sharp images.

Oord et al. [Oor+18] used flows for audio synthesis by distilling WaveNet into an IAF (Section 24.2.4.3), which enables faster sampling than WaveNet. Other flow models for audio include WaveFLOW [PVC19] and FlowWaveNet [Kim+19], which directly speed up WaveNet using coupling layers.

Flows have been also used for text. Tran et al. [Tra+19] define a discrete flow over a vocabulary for language-modeling tasks. Another popular approach is to define a latent variable model with discrete observation space but a continuous latent space. For example, Ziegler and Rush [ZR19a] use normalizing flows in latent space for language modeling.

### 24.3.3 Inference

Normalizing flows have been used for probabilistic inference. Rezende and Mohamed [RM15b] popularized normalizing flows in machine learning, and showed how they can be used for modeling variational posterior distributions in latent variable models. Various extensions such as Householder flows [TW16], inverse autoregressive flows [Kin+16], multiplicative normalizing flows [LW17] and Sylvester flows [Ber+18] have been proposed for modeling the variational posterior for latent variable models as well as posteriors for Bayesian neural networks.

Flows have been used as complex proposal distributions for importance sampling; examples include neural importance sampling [Mül+19b] and Boltzmann generators [Noé+19]. Hoffman et al. [Hof+19] used flows to improve the performance of Hamiltonian Monte Carlo (Section 12.5) by defining bijective transformations to transform random variables to simpler distributions and performing HMC in that space instead.

Finally, flows can be used in the context of simulation-based inference, where the likelihood function of the parameters is not available, but simulating data from the model is possible. The main idea is to train a flow on data simulated from the model in order to approximate the posterior distribution or the likelihood function. The flow model can also be used to guide simulations in order to make inference more efficient [PSM19; GNM19]. This approach has been used for inference of simulation

1 models in cosmology [Als+19] and computational neuroscience [Gon+20].  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47

# 25 Energy-based models

*This chapter was co-authored with Yang Song and Durk Kingma, and is an extension of [SK21].*

## 25.1 Introduction

We have now seen several ways of defining deep generative models, including VAEs (Chapter 22), auto-regressive models (Chapter 23) and normalizing flows (Chapter 24). All of the above models can be formulated in terms of directed graphical models (Chapter 4), where we generate the data one step at a time, using locally normalized distributions. In some cases, it is easier to specify a distribution in terms of a set of constraints that valid samples must satisfy, rather than a generative process. This can be done using an undirected graphical model (Chapter 4).

**Energy-based models** or **EBM** are similar to undirected graphical models, but they do not necessarily make any Markov assumptions. Thus they can be written as a Gibbs distribution, as follows:

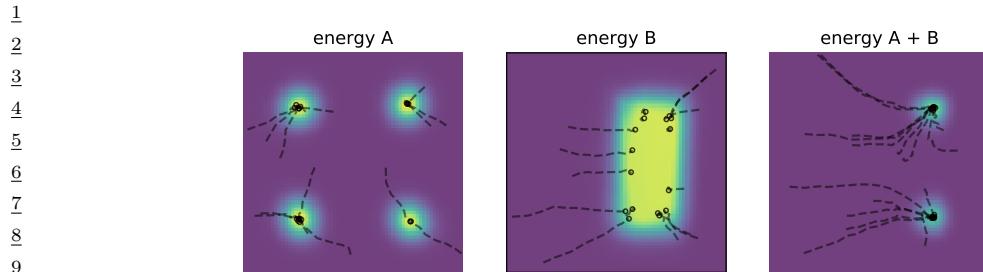
$$p_{\theta}(\mathbf{x}) = \frac{\exp(-E_{\theta}(\mathbf{x}))}{Z_{\theta}} \quad (25.1)$$

where  $E_{\theta}(\mathbf{x}) \geq 0$  is known as the **energy function** with parameters  $\theta$ , and  $Z_{\theta}$  is the **partition function**:

$$Z_{\theta} = \int \exp(-E_{\theta}(\mathbf{x})) \, d\mathbf{x} \quad (25.2)$$

This is constant w.r.t.  $\mathbf{x}$  but is a function of  $\theta$ . In general, evaluating this integral is intractable, although we can use approximate methods, such as annealed importance sampling, discussed in Section 11.5.4.1.

Since the energy function does not need to integrate to one, and just needs to output a single scalar, it can be implemented using a variety of neural network architectures. As such, EBMs have found wide applications in many fields of machine learning, including image generation [Ngi+11; Xie+16; DM19b], discriminative learning [Gra+20b], natural processing [Mik+13; Den+20], density estimation [Wen+19a; Son+19] and reinforcement learning [Haa+17; Haa+18a], to list a few. (More examples can be found at <https://github.com/yataobian/awesome-ebm>.)



*Figure 25.1: Combining two energy functions in 2d by summation, which is equivalent to multiplying the corresponding probability densities. We also illustrate some sampled trajectories towards high probability (low energy) regions. From Figure 14 of [DM19a]. Used with kind permission of Yilun Du.*

### 25.1.1 Example: Products of experts (PoE)

As an example of why energy based models are useful, suppose we want to create a generative model of proteins that are thermally stable at room temperature, and which bind to the COVID-19 spike receptor. Suppose  $p_1(\mathbf{x})$  can generate stable proteins and  $p_2(\mathbf{x})$  can generate proteins that bind. (For example, both of these models could be autoregressive sequence models, trained on different datasets.) We can view each of these models as “experts” about a particular aspect of the data. On their own, they are not an adequate model of the data that we have (or want to have), but we can then combine them, to represent the **conjunction of features**, by computing a **product of experts (PoE)** [Hin02]:

$$p_{12}(\mathbf{x}) = \frac{1}{Z_{12}} p_1(\mathbf{x}) p_2(\mathbf{x}) \quad (25.3)$$

This will assign high probability to proteins that are stable and which bind, and low probability to all others. By contrast, a **mixture of experts** would either generate from  $p_1$  or from  $p_2$ , but would not combine features from both.

If the experts are represented as energy based models (EBM), then the PoE model is also an EBM, with an energy given by

$$\mathcal{E}_{12}(\mathbf{x}) = \mathcal{E}_1(\mathbf{x}) + \mathcal{E}_2(\mathbf{x}) \quad (25.4)$$

Intuitively, we can think of each component of energy as a “soft constraint” on the data. This idea is illustrated in Figure 25.1.

### 25.1.2 Computational difficulties

Although the flexibility of EBMs can provide significant modeling advantages, computation of the likelihood and drawing samples from the model are generally intractable. In this chapter, we will discuss a variety of approximate methods to solve these problems.

## 25.2 Maximum Likelihood Training

The *de facto* standard for learning probabilistic models from i.i.d. data is maximum likelihood estimation (MLE). Let  $p_{\theta}(\mathbf{x})$  be a probabilistic model parameterized by  $\theta$ , and  $p_{\text{data}}(\mathbf{x})$  be the underlying data distribution of a dataset. We can fit  $p_{\theta}(\mathbf{x})$  to  $p_{\text{data}}(\mathbf{x})$  by maximizing the expected log-likelihood function over the data distribution, defined by

$$\ell(\theta) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log p_{\theta}(\mathbf{x})] \quad (25.5)$$

as a function of  $\theta$ . Here the expectation can be easily estimated with samples from the dataset. Maximizing likelihood is equivalent to minimizing the KL divergence between  $p_{\text{data}}(\mathbf{x})$  and  $p_{\theta}(\mathbf{x})$ , because

$$\ell(\theta) = D_{KL}(p_{\text{data}}(\mathbf{x}) \parallel p_{\theta}(\mathbf{x})) - \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log p_{\text{data}}(\mathbf{x})] \quad (25.6)$$

$$= D_{KL}(p_{\text{data}}(\mathbf{x}) \parallel p_{\theta}(\mathbf{x})) - \text{constant}, \quad (25.7)$$

where the second equality holds because  $\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log p_{\text{data}}(\mathbf{x})]$  does not depend on  $\theta$ .

We cannot usually compute the likelihood of an EBM because the normalizing constant  $Z_{\theta}$  is often intractable. Nevertheless, we can still estimate the gradient of the log-likelihood with MCMC approaches, allowing for likelihood maximization with stochastic gradient ascent [You99]. In particular, the gradient of the log-probability of an EBM decomposes as a sum of two terms:

$$\nabla_{\theta} \log p_{\theta}(\mathbf{x}) = -\nabla_{\theta} E_{\theta}(\mathbf{x}) - \nabla_{\theta} \log Z_{\theta}. \quad (25.8)$$

The first gradient term,  $-\nabla_{\theta} E_{\theta}(\mathbf{x})$ , is straightforward to evaluate with automatic differentiation. The challenge is in approximating the second gradient term,  $\nabla_{\theta} \log Z_{\theta}$ , which is intractable to compute exactly. This gradient term can be rewritten as the following expectation:

$$\nabla_{\theta} \log Z_{\theta} = \nabla_{\theta} \log \int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x} \quad (25.9)$$

$$\stackrel{(i)}{=} \left( \int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x} \right)^{-1} \nabla_{\theta} \int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x} \quad (25.10)$$

$$= \left( \int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x} \right)^{-1} \int \nabla_{\theta} \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x} \quad (25.11)$$

$$\stackrel{(ii)}{=} \left( \int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x} \right)^{-1} \int \exp(-E_{\theta}(\mathbf{x})) (-\nabla_{\theta} E_{\theta}(\mathbf{x})) d\mathbf{x} \quad (25.12)$$

$$= \int \left( \int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x} \right)^{-1} \exp(-E_{\theta}(\mathbf{x})) (-\nabla_{\theta} E_{\theta}(\mathbf{x})) d\mathbf{x} \quad (25.13)$$

$$\stackrel{(iii)}{=} \int \exp(-E_{\theta}(\mathbf{x}) - Z_{\theta}) (-\nabla_{\theta} E_{\theta}(\mathbf{x})) d\mathbf{x} \quad (25.14)$$

$$\stackrel{(iv)}{=} \int p_{\theta}(\mathbf{x}) (-\nabla_{\theta} E_{\theta}(\mathbf{x})) d\mathbf{x} \quad (25.15)$$

$$= \mathbb{E}_{\mathbf{x} \sim p_{\theta}(\mathbf{x})} [-\nabla_{\theta} E_{\theta}(\mathbf{x})], \quad (25.16)$$

where steps (i) and (ii) are due to the chain rule of gradients, and (iii) and (iv) are from definitions in Equations (25.1) and (25.2). Thus, we can obtain an unbiased one sample Monte Carlo estimate of the log-likelihood gradient by using

$$\nabla_{\theta} \log Z_{\theta} \simeq -\frac{1}{S} \sum_{s=1}^S \nabla_{\theta} E_{\theta}(\tilde{\mathbf{x}}_s), \quad (25.17)$$

where  $\tilde{\mathbf{x}}_s \sim p_{\theta}(\mathbf{x})$ , i.e., a random sample from the distribution over  $\mathbf{x}$  given by the EBM. Therefore, as long as we can draw random samples from the model, we have access to an unbiased Monte Carlo estimate of the log-likelihood gradient, allowing us to optimize the parameters with stochastic gradient ascent.

Much of the literature has focused on methods for efficient MCMC sampling from EBMs. We discuss some of these methods below.

### 25.2.1 Gradient-based MCMC methods

Some efficient MCMC methods, such as **Langevin MCMC** (Section 12.5.6) or Hamiltonian Monte Carlo (Section 12.5), make use of the fact that the gradient of the log-probability w.r.t.  $\mathbf{x}$  (known as the **score function**) is equal to the (negative) gradient of the energy, and is therefore easy to calculate:

$$\nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x}) = -\nabla_{\mathbf{x}} E_{\theta}(\mathbf{x}) - \underbrace{\nabla_{\mathbf{x}} \log Z_{\theta}}_{=0} = -\nabla_{\mathbf{x}} E_{\theta}(\mathbf{x}). \quad (25.18)$$

For example, when using Langevin MCMC to sample from  $p_{\theta}(\mathbf{x})$ , we first draw an initial sample  $\mathbf{x}^0$  from a simple prior distribution, and then simulate an overdamped Langevin diffusion process for  $K$  steps with step size  $\epsilon > 0$ :

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \frac{\epsilon^2}{2} \underbrace{\nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x}^k)}_{= -\nabla_{\mathbf{x}} E_{\theta}(\mathbf{x})} + \epsilon \mathbf{z}^k, \quad k = 0, 1, \dots, K-1. \quad (25.19)$$

where  $\mathbf{z}^k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  is a Gaussian noise term. We show an example of this process in Figure 25.3d.

When  $\epsilon \rightarrow 0$  and  $K \rightarrow \infty$ ,  $\mathbf{x}^K$  is guaranteed to distribute as  $p_{\theta}(\mathbf{x})$  under some regularity conditions. In practice we have to use a small finite  $\epsilon$ , but the discretization error is typically negligible, or can be corrected with a Metropolis-Hastings step (Section 12.2), leading to the Metropolis-Adjusted Langevin Algorithm (Section 12.5.6).

### 25.2.2 Contrastive divergence

Running MCMC till convergence to obtain a sample  $\mathbf{x} \sim p_{\theta}(\mathbf{x})$  can be computationally expensive. Therefore we typically need approximations to make MCMC-based learning of EBMs practical. One popular method for doing so is **contrastive divergence** (CD) [Hin02]. In CD, one initializes the MCMC chain from the datapoint  $\mathbf{x}$ , and proceeds to perform MCMC for a fixed number of steps. One can show that  $T$  steps of CD minimizes the following objective:

$$\text{CD}_T = D_{\text{KL}}(p_0 \| p_{\infty}) - D_{\text{KL}}(p_T \| p_{\infty}) \quad (25.20)$$

where  $p_t$  is the distribution over  $\mathbf{x}$  after  $t$  MCMC updates, and  $p_0$  is the data distribution. Typically we can get good results with a small value of  $T$ , sometimes just  $T = 1$ . We give the details below.

### 25.2.2.1 Fitting RBMs with CD

CD was initially developed to fit a special kind of latent variable EBM known as a restricted Boltzmann machine (Section 4.3.2.4). This model was specially designed to support fast block Gibbs sampling, which is required by CD (and can also be exploited by standard MCMC-based learning methods [AHS85].)

For simplicity, we will assume the hidden and visible nodes are binary, and we use 1-step contrastive divergence. As discussed in the supplementary material, the binary RBM has the following energy function:

$$\mathcal{E}(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}) = \sum_{d=1}^D \sum_{k=1}^K x_d z_k W_{dk} + \sum_{d=1}^D x_d b_d + \sum_{k=1}^K z_k c_k \quad (25.21)$$

(Henceforth we will drop the unary (bias) terms, which can be emulated by clamping  $z_k = 1$  or  $x_d = 1$ .) This is a loglinear model where we have one binary feature per edge. Thus from Equation (4.115) the gradient of the log-likelihood is given by the clamped expectations minus the unclamped expectations:

$$\frac{\partial \ell}{\partial w_{dk}} = \frac{1}{N} \sum_{n=1}^N \mathbb{E}[x_d z_k | \mathbf{x}_n, \boldsymbol{\theta}] - \mathbb{E}[x_d z_k | \boldsymbol{\theta}] \quad (25.22)$$

We can write rewrite the above gradient in matrix-vector form as follows:

$$\nabla_{\mathbf{w}} \ell = \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})p(\mathbf{z}|\boldsymbol{\theta})} [\mathbf{x}\mathbf{z}^T] - \mathbb{E}_{p(\mathbf{z}|\boldsymbol{\theta})} [\mathbf{x}\mathbf{z}^T] \quad (25.23)$$

(We can derive a similar expression for the gradient of the bias terms by setting  $x_d = 1$  or  $z_k = 1$ .)

The first term in the expression for the gradient in Equation (25.22), when  $\mathbf{x}$  is fixed to a data case, is sometimes called the **clamped phase**, and the second term, when  $\mathbf{x}$  is free, is sometimes called the **unclamped phase**. When the model expectations match the empirical expectations, the two terms cancel out, the gradient becomes zero and learning stops.

We can also make a connection to the principle of **Hebbian learning** in neuroscience. In particular, Hebb's rule says that the strength of connection between two neurons that are simultaneously active should be increased. (This theory is often summarized as "Cells that fire together wire together".<sup>1</sup>) The first term in Equation (25.22) is therefore consider a Hebbian term, and the second an anti-Hebbian term, due to the sign change.

We can leverage the Markov structure of the bipartite graph to approximate the expectations as follows:

$$\mathbf{z}_n \sim p(\mathbf{z} | \mathbf{x}_n, \boldsymbol{\theta}) \quad (25.24)$$

$$\mathbf{x}'_n \sim p(\mathbf{x} | \mathbf{z}_n, \boldsymbol{\theta}) \quad (25.25)$$

$$\mathbf{z}'_n \sim p(\mathbf{z} | \mathbf{x}'_n, \boldsymbol{\theta}) \quad (25.26)$$

<sup>1</sup> See [https://en.wikipedia.org/wiki/Hebbian\\_theory](https://en.wikipedia.org/wiki/Hebbian_theory).

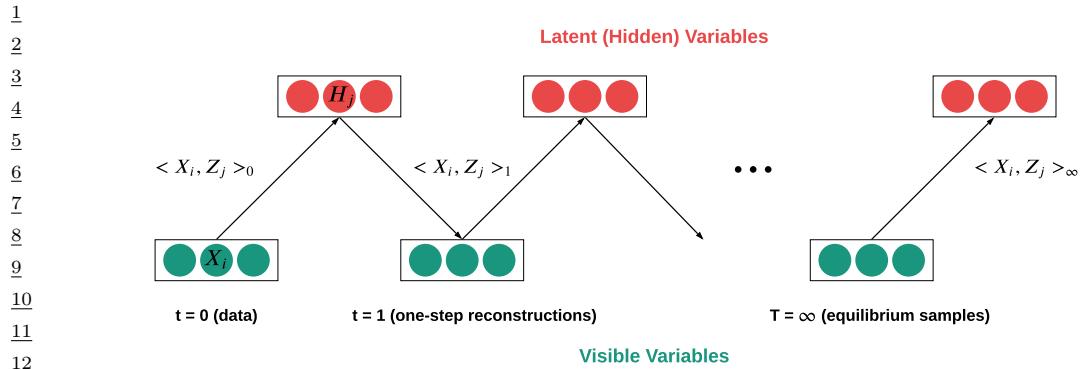


Figure 25.2: Illustration of contrastive divergence sampling for an RBM. The visible nodes are initialized at an example drawn from the data set. Then we sample a hidden vector, then another visible vector, etc. Eventually (at “infinity”) we will be producing samples from the joint distribution  $p(\mathbf{x}, \mathbf{z}|\theta)$ .

We can think of  $\mathbf{x}'_n$  as the model’s best attempt at reconstructing  $\mathbf{x}_n$  after being encoded and then decoded by the model. Such samples are sometimes called **fantasy data**. See Figure 25.2 for an illustration. Given these samples, we then make the approximation

$$\mathbb{E}_{p(\cdot|\theta)} [\mathbf{x}\mathbf{z}^\top] \approx \mathbf{x}_n(\mathbf{z}'_n)^\top \quad (25.27)$$

In practice, it is common to use  $\mathbb{E}[\mathbf{z}|\mathbf{x}'_n]$  instead of a sampled value  $\mathbf{z}'_n$  in the above expression, since this reduces the variance. However, it is not valid to use  $\mathbb{E}[\mathbf{z}|\mathbf{x}_n]$  instead of sampling  $\mathbf{z}_n \sim p(\mathbf{z}|\mathbf{x}_n)$  in Equation (25.24), because then each hidden unit would be able to pass more than 1 bit of information, so it would not act as much of a bottleneck.

The whole procedure is summarized in Algorithm 28. For more details, see [Hin10; Swe+10].

---

**Algorithm 28:** CD-1 training for an RBM with binary hidden and visible units

---

```

32 1 Initialize weights  $\mathbf{W} \in \mathbb{R}^{D \times K}$  randomly;
33 2 for  $t = 1, 2, \dots$  do
34 3   for each minibatch of size  $B$  do
35 4     Set minibatch gradient to zero,  $\mathbf{g} := \mathbf{0}$ 
36 5     for each case  $\mathbf{x}_n$  in the minibatch do
37 6       Compute  $\boldsymbol{\mu}_n = \mathbb{E}[\mathbf{z}|\mathbf{x}_n, \mathbf{W}]$ 
38 7       Sample  $\mathbf{z}_n \sim p(\mathbf{z}|\mathbf{x}_n, \mathbf{W})$ 
39 8       Sample  $\mathbf{x}'_n \sim p(\mathbf{x}|\mathbf{z}_n, \mathbf{W})$ 
40 9       Compute  $\boldsymbol{\mu}'_n = \mathbb{E}[\mathbf{z}|\mathbf{x}'_n, \mathbf{W}]$ 
41 10      Compute gradient  $\nabla_{\mathbf{W}} = (\mathbf{x}_n)(\boldsymbol{\mu}_n)^\top - (\mathbf{x}'_n)(\boldsymbol{\mu}'_n)^\top$ 
42 11      Accumulate  $\mathbf{g} := \mathbf{g} + \nabla_{\mathbf{W}}$ 
43 12    Update parameters  $\mathbf{W} := \mathbf{W} + \eta_t \frac{1}{B} \mathbf{g}$ 

```

---

---

### 25.2.2.2 Persistent CD

One variant of CD that sometimes performs better is **persistent contrastive divergence** (PCD) [Tie08]. In this approach, a single MCMC chain with a persistent state is employed to sample from the EBM. In PCD, we do not restart the MCMC chain when training on a new datapoint; rather, we carry over the state of the previous MCMC chain and use it to initialize a new MCMC chain for the next training step. See Line 12 for some pseudocode. Hence there are two dynamical processes running at different time scales: the states  $\mathbf{x}$  change quickly, and the parameters  $\boldsymbol{\theta}$  change slowly.

---

#### Algorithm 29: Persistent MCMC-SGD for fitting an EBM

---

```

1 Initialize parameters  $\boldsymbol{\theta}$  randomly
2 Initialize chains  $\tilde{\mathbf{x}}_{1:S}$  randomly
3 Initialize learning rate  $\eta$ 
4 for  $t = 1, 2, \dots$  do
5   for  $\mathbf{x}_b$  in minibatch of size  $B$  do
6      $\mathbf{g}_b = \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x}_b)$ 
7     for sample  $s = 1 : S$  do
8       Sample  $\tilde{\mathbf{x}}_s \sim \text{MCMC}(\text{target} = p(\cdot | \boldsymbol{\theta}), \text{init} = \tilde{\mathbf{x}}_s, \text{nsteps} = N)$ 
9        $\tilde{\mathbf{g}}_s = \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_s)$ 
10       $\mathbf{g}_t = -(\frac{1}{B} \sum_{b=1}^B \mathbf{g}_b) - (\frac{1}{S} \sum_{s=1}^S \tilde{\mathbf{g}}_s)$ 
11       $\boldsymbol{\theta} := \boldsymbol{\theta} + \eta \mathbf{g}_t$ 
12      Decrease step size  $\eta$ 

```

---

A theoretical justification for this was given in [You89], who showed that we can start the MCMC chain at its previous value, and just take a few steps, because  $p(\mathbf{x}|\boldsymbol{\theta}_t)$  is likely to be close to  $p(\mathbf{x}|\boldsymbol{\theta}_{t-1})$ , since we only changed the parameters by a small amount in the intervening SGD step.

### 25.2.2.3 Other methods

PCD can be further improved by keeping multiple historical states of the MCMC chain in a replay buffer and initialize new MCMC chains by randomly sampling from it [DM19b]. Other variants of CD include mean field CD [WH02], and multi-grid CD [Gao+18].

EBMs trained with CD may not capture the data distribution faithfully, since truncated MCMC can lead to biased gradient updates that hurt the learning dynamics [SMB10; FI10; Nij+19]. There are several methods that focus on removing this bias for improved MCMC training. For example, one line of work proposes unbiased estimators of the gradient through coupled MCMC [JOA17; QZW19]; and Du et al. [Du+20] propose to reduce the bias by differentiating through the MCMC sampling algorithm and estimating an entropy correction term.

## 25.3 Score Matching (SM)

If two continuously differentiable real-valued functions  $f(\mathbf{x})$  and  $g(\mathbf{x})$  have equal first derivatives everywhere, then  $f(\mathbf{x}) \equiv g(\mathbf{x}) + \text{constant}$ . When  $f(\mathbf{x})$  and  $g(\mathbf{x})$  are log probability density functions

(PDFs) with equal first derivatives, the normalization requirement (Equation (25.1)) implies that  $\int \exp(f(\mathbf{x}))d\mathbf{x} = \int \exp(g(\mathbf{x}))d\mathbf{x} = 1$ , and therefore  $f(\mathbf{x}) \equiv g(\mathbf{x})$ . As a result, one can learn an EBM by (approximately) matching the first derivatives of its log-PDF to the first derivatives of the log-PDF of the data distribution. If they match, then the EBM captures the data distribution exactly. The first-order gradient function of a log-PDF is also called the **score** of that distribution. For training EBMs, it is useful to transform the equivalence of distributions to the equivalence of scores, because the score of an EBM can be easily obtained as follows:

$$\mathbf{s}_{\theta}(\mathbf{x}) \triangleq \nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x}) = -\nabla_{\mathbf{x}} E_{\theta}(\mathbf{x}) \quad (25.28)$$

We see that this does not involve the typically intractable normalizing constant  $Z_{\theta}$ .

Let  $p_{\text{data}}(\mathbf{x})$  be the underlying data distribution, from which we have a finite number of i.i.d. samples but do not know its PDF. The **score matching** objective [Hyv05] minimizes a discrepancy between two distributions called the **Fisher divergence**:

$$D_F(p_{\text{data}}(\mathbf{x}) \parallel p_{\theta}(\mathbf{x})) = \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[ \frac{1}{2} \|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x})\|^2 \right]. \quad (25.29)$$

The expectation w.r.t.  $p_{\text{data}}(\mathbf{x})$ , in this objective and its variants below, admits a trivial unbiased Monte Carlo estimator using the empirical mean of samples  $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ . However, the second term of Equation (25.29),  $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ , is generally impractical to calculate since it requires knowing the PDF of  $p_{\text{data}}(\mathbf{x})$ . We discuss a solution to this below.

### 25.3.1 Basic score matching

Hyvärinen [Hyv05] shows that, under certain regularity conditions, the Fisher divergence can be rewritten using integration by parts, with second derivatives of  $E_{\theta}(\mathbf{x})$  replacing the unknown first derivatives of  $p_{\text{data}}(\mathbf{x})$ :

$$D_F(p_{\text{data}}(\mathbf{x}) \parallel p_{\theta}(\mathbf{x})) = \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[ \frac{1}{2} \sum_{i=1}^d \left( \frac{\partial E_{\theta}(\mathbf{x})}{\partial x_i} \right)^2 + \frac{\partial^2 E_{\theta}(\mathbf{x})}{\partial x_i^2} \right] + \text{constant} \quad (25.30)$$

$$= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[ \frac{1}{2} \|\mathbf{s}_{\theta}(\mathbf{x})\|^2 + \text{tr}(\mathbf{J}_{\mathbf{x}} \mathbf{s}_{\theta}(\mathbf{x})) \right] + \text{constant} \quad (25.31)$$

where  $d$  is the dimensionality of  $\mathbf{x}$ , and  $\mathbf{J}_{\mathbf{x}} \mathbf{s}_{\theta}(\mathbf{x})$  is the Jacobian of the score function. The constant does not affect optimization and thus can be dropped for training. It is shown by [Hyv05] that estimators based on Score Matching are consistent under some regularity conditions, meaning that the parameter estimator obtained by minimizing Equation (25.29) converges to the true parameters in the limit of infinite data. See Figure 25.3 for an example.

An important downside of the objective Equation (25.31) is that it takes  $O(d^2)$  time to compute the trace of the Jacobian. For this reason, the implicit SM formulation of Equation (25.31) has only been applied to relatively simple energy functions where computation of the second derivatives is tractable.

Score Matching assumes a continuous data distribution with positive density over the space, but it can be generalized to discrete or bounded data distributions [Hyv07b; Lyu12]. It is also possible to consider higher-order gradients of log-PDFs beyond first derivatives [PDL+12].

47

### 25.3.2 Denoising Score Matching (DSM)

The Score Matching objective in Equation (25.31) requires several regularity conditions for  $\log p_{\text{data}}(\mathbf{x})$ , *e.g.*, it should be continuously differentiable and finite everywhere. However, these conditions may not always hold in practice. For example, a distribution of digital images is typically discrete and bounded, because the values of pixels are restricted to the range  $\{0, 1, \dots, 255\}$ . Therefore,  $\log p_{\text{data}}(\mathbf{x})$  in this case is discontinuous and is negative infinity outside the range, and thus SM is not directly applicable.

To alleviate this, one can add a bit of noise to each datapoint:  $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\epsilon}$ . As long as the noise distribution  $p(\boldsymbol{\epsilon})$  is smooth, the resulting noisy data distribution  $q(\tilde{\mathbf{x}}) = \int q(\tilde{\mathbf{x}} | \mathbf{x}) p_{\text{data}}(\mathbf{x}) d\mathbf{x}$  is also smooth, and thus the Fisher divergence  $D_F(q(\tilde{\mathbf{x}}) \| p_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}))$  is a proper objective. [KL10] showed that the objective with noisy data can be approximated by the noiseless Score Matching objective of Equation (25.31) plus a regularization term; this regularization makes Score Matching applicable to a wider range of data distributions, but still requires expensive second-order derivatives.

[Vin11] proposed an elegant and scalable solution to the above difficulty, by showing that:

$$D_F(q(\tilde{\mathbf{x}}) \| p_{\boldsymbol{\theta}}(\tilde{\mathbf{x}})) = \mathbb{E}_{q(\tilde{\mathbf{x}})} \left[ \frac{1}{2} \|\nabla_{\mathbf{x}} \log q(\tilde{\mathbf{x}}) - \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\tilde{\mathbf{x}})\|_2^2 \right] \quad (25.32)$$

$$= \mathbb{E}_{q(\mathbf{x}, \tilde{\mathbf{x}})} \left[ \frac{1}{2} \|\nabla_{\mathbf{x}} \log q(\tilde{\mathbf{x}} | \mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\tilde{\mathbf{x}})\|_2^2 \right] + \text{constant}, \quad (25.33)$$

where the expectation is again approximated by the empirical average of samples, thus completely avoiding both the unknown term  $p_{\text{data}}(\mathbf{x})$  and computationally expensive second-order derivatives. The constant term does not affect optimization and can be ignored without changing the optimal solution. This estimation method is called **Denoising Score Matching** (DSM) by [Vin11]. Similar formulations were also explored by Raphan and Simoncelli [RS07; RS11] and can be traced back to Tweedie's formula [Efr11] and Stein's Unbiased Risk Estimation [Ste81].

#### 25.3.2.1 Difficulties

The major drawback of adding noise to data arises when  $p_{\text{data}}(\mathbf{x})$  is already a well-behaved distribution that satisfies the regularity conditions required by Score Matching. In this case,  $D_F(q(\tilde{\mathbf{x}}) \| p_{\boldsymbol{\theta}}(\tilde{\mathbf{x}})) \neq D_F(p_{\text{data}}(\mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{x}))$ , and DSM is not a consistent objective because the optimal EBM matches the noisy distribution  $q(\tilde{\mathbf{x}})$ , not  $p_{\text{data}}(\mathbf{x})$ . This inconsistency becomes non-negligible when  $q(\tilde{\mathbf{x}})$  significantly differs from  $p_{\text{data}}(\mathbf{x})$ .

One way to attenuate the inconsistency of DSM is to choose  $q \approx p_{\text{data}}$ , *i.e.*, use a small noise perturbation. However, this often significantly increases the variance of objective values and hinders optimization. As an example, suppose  $q(\tilde{\mathbf{x}} | \mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}} | \mathbf{x}, \sigma^2 I)$  and  $\sigma \approx 0$ . The corresponding DSM objective is

$$\begin{aligned} D_F(q(\tilde{\mathbf{x}}) \| p_{\boldsymbol{\theta}}(\tilde{\mathbf{x}})) &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(0, I)} \left[ \frac{1}{2} \left\| \frac{\mathbf{z}}{\sigma} + \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x} + \sigma \mathbf{z}) \right\|_2^2 \right] \\ &\simeq \frac{1}{2N} \sum_{i=1}^N \left\| \frac{\mathbf{z}^{(i)}}{\sigma} + \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)} + \sigma \mathbf{z}^{(i)}) \right\|_2^2, \end{aligned} \quad (25.34)$$

where  $\{\mathbf{x}^{(i)}\}_{i=1}^N \stackrel{\text{i.i.d.}}{\sim} p_{\text{data}}(\mathbf{x})$ , and  $\{\mathbf{z}^{(i)}\}_{i=1}^N \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I})$ . When  $\sigma \rightarrow 0$ , we can leverage Taylor series expansion to rewrite the Monte Carlo estimator in Equation (25.34) to

$$\frac{1}{2N} \sum_{i=1}^N \left[ \frac{2}{\sigma} (\mathbf{z}^{(i)})^\top \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) + \frac{\|\mathbf{z}^{(i)}\|_2^2}{\sigma^2} \right] + \text{constant.} \quad (25.35)$$

When estimating the above expectation with samples, the variances of  $(\mathbf{z}^{(i)})^\top \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})/\sigma$  and  $\|\mathbf{z}^{(i)}\|_2^2/\sigma^2$  will both grow unbounded as  $\sigma \rightarrow 0$  due to division by  $\sigma$  and  $\sigma^2$ . This enlarges the variance of DSM and makes optimization challenging. Various methods have been proposed to reduce this variance (see e.g., [Wan+20c]).

### 25.3.3 Sliced Score Matching (SSM)

By adding noise to data, DSM avoids the expensive computation of second-order derivatives. However, as mentioned before, the optimal EBM that minimizes the DSM objective corresponds to the distribution of noise-perturbed data  $q(\tilde{\mathbf{x}})$ , not the original noise-free data distribution  $p_{\text{data}}(\mathbf{x})$ . In other words, DSM does not give a consistent estimator of the data distribution, *i.e.*, one cannot directly obtain an EBM that exactly matches the data distribution even with unlimited data.

**Sliced Score Matching (SSM)** [Son+19] is one alternative to Denoising Score Matching that is both consistent and computationally efficient. Instead of minimizing the Fisher divergence between two vector-valued scores, SSM randomly samples a projection vector  $\mathbf{v}$ , takes the inner product between  $\mathbf{v}$  and the two scores, and then compares the resulting two scalars. More specifically, Sliced Score Matching minimizes the following divergence called the **sliced Fisher divergence**:

$$D_{SF}(p_{\text{data}}(\mathbf{x}) || p_{\boldsymbol{\theta}}(\mathbf{x})) = \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{p(\mathbf{v})} \left[ \frac{1}{2} (\mathbf{v}^\top \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \mathbf{v}^\top \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}))^2 \right], \quad (25.36)$$

where  $p(\mathbf{v})$  denotes a projection distribution such that  $\mathbb{E}_{p(\mathbf{v})}[\mathbf{v}\mathbf{v}^\top]$  is positive definite. Similar to Fisher divergence, sliced Fisher divergence has an implicit form that does not involve the unknown  $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ , which is given by

$$D_{SF}(p_{\text{data}}(\mathbf{x}) || p_{\boldsymbol{\theta}}(\mathbf{x})) = \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{p(\mathbf{v})} \left[ \frac{1}{2} \sum_{i=1}^d \left( \frac{\partial E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_i} v_i \right)^2 + \sum_{i=1}^d \sum_{j=1}^d \frac{\partial^2 E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_i \partial x_j} v_i v_j \right] + C. \quad (25.37)$$

All expectations in the above objective can be estimated with empirical means, and again the constant term  $C$  can be removed without affecting training. The second term involves second-order derivatives of  $E_{\boldsymbol{\theta}}(\mathbf{x})$ , but contrary to SM, it can be computed efficiently with a cost linear in the dimensionality  $d$ . This is because

$$\sum_{i=1}^d \sum_{j=1}^d \frac{\partial^2 E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_i \partial x_j} v_i v_j = \sum_{i=1}^d \frac{\partial}{\partial x_i} \underbrace{\left( \sum_{j=1}^d \frac{\partial E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_j} v_j \right)}_{:=f(\mathbf{x})} v_i, \quad (25.38)$$

where  $f(\mathbf{x})$  is the same for different values of  $i$ . Therefore, we only need to compute it once with  $O(d)$  computation, *plus* another  $O(d)$  computation for the outer sum to evaluate Equation (25.38), whereas the original SM objective requires  $O(d^2)$  computation.

For many choices of  $p(\mathbf{v})$ , part of the SSM objective (Equation (25.37)) can be evaluated in closed form, potentially leading to lower variance. For example, when  $p(\mathbf{v}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ , we have

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{p(\mathbf{v})} \left[ \frac{1}{2} \sum_{i=1}^d \left( \frac{\partial E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_i} v_i \right)^2 \right] = \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[ \frac{1}{2} \sum_{i=1}^d \left( \frac{\partial E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_i} \right)^2 \right] \quad (25.39)$$

and as a result,

$$D_{SF}(p_{\text{data}}(\mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{x})) = \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \frac{1}{2} \sum_{i=1}^d \left( \frac{\partial E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_i} \right)^2 + \sum_{i=1}^d \sum_{j=1}^d \frac{\partial^2 E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_i \partial x_j} v_i v_j \right] + C \quad (25.40)$$

$$= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \frac{1}{2} (\mathbf{v}^T \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}))^2 + \mathbf{v}^T [\mathbf{J}\mathbf{v}] \right] \quad (25.41)$$

where  $\mathbf{J} = \mathbf{J}_x \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})$ . (Note that  $\mathbf{J}\mathbf{v}$  can be computed using a Jacobian vector product operation.)

The above objective Equation (25.40) can also be obtained by approximating the sum of second-order gradients in the standard SM objective (Equation (25.31)) with the Hutchinson trace estimator [Ski89; Hut89; Mey+21]. It often (but not always) has lower variance than Equation (25.37), and can perform better in some applications [Son+19].

### 25.3.4 Connection to Contrastive Divergence

Though Score Matching and Contrastive Divergence (Section 25.2.2) are seemingly very different approaches, they are closely connected to each other. In fact, Score Matching can be viewed as a special instance of Contrastive Divergence in the limit of a particular MCMC sampler [Hyy07a]. Moreover, the Fisher divergence optimized by Score Matching is related to the derivative of KL divergence [Cov99], which is the underlying objective of Contrastive Divergence.

Contrastive Divergence requires sampling from the Energy-Based Model  $E_{\boldsymbol{\theta}}(\mathbf{x})$ , and one popular method for doing so is Langevin MCMC. Recall from Section 25.2.1 that given any initial data point  $\mathbf{x}^0$ , the Langevin MCMC method executes the following

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \frac{\epsilon}{2} \nabla_{\mathbf{x}} E_{\boldsymbol{\theta}}(\mathbf{x}^k) + \sqrt{\epsilon} \mathbf{z}^k, \quad (25.42)$$

iteratively for  $k = 0, 1, \dots, K-1$ , where  $\mathbf{z}^k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  and  $\epsilon > 0$  is the step size.

Suppose we only run one-step Langevin MCMC for Contrastive Divergence. In this case, the gradient of the log-likelihood is given by

$$\begin{aligned} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} [\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x})] &= -\mathbb{E}_{p_{\text{data}}(\mathbf{x})} [\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x})} [\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x})] \\ &\simeq -\mathbb{E}_{p_{\text{data}}(\mathbf{x})} [\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}} \left( \mathbf{x} - \frac{\epsilon^2}{2} \nabla_{\mathbf{x}} E_{\boldsymbol{\theta}'}(\mathbf{x}) + \epsilon \mathbf{z} \right) \Big|_{\boldsymbol{\theta}'=\boldsymbol{\theta}} \right]. \end{aligned} \quad (25.43)$$

1 After Taylor series expansion with respect to  $\epsilon$  followed by some algebraic manipulations, the above  
 2 equation can be transformed to the following (see Hyvarinen [Hyv07a])  
 3

4

$$\frac{\epsilon^2}{2} \nabla_{\theta} D_F(p_{\text{data}}(\mathbf{x}) \| p_{\theta}(\mathbf{x})) + o(\epsilon^2). \quad (25.44)$$

5

6

7 When  $\epsilon$  is sufficiently small, it corresponds to the re-scaled gradient of the Score Matching objective.  
 8

9 In general, Score Matching minimizes the Fisher divergence  $D_F(p_{\text{data}}(\mathbf{x}) \| p_{\theta}(\mathbf{x}))$ , whereas  
 10 Contrastive Divergence minimizes an objective related to the KL divergence  $D_{KL}(p_{\text{data}}(\mathbf{x}) \| p_{\theta}(\mathbf{x}))$ ,  
 11 as shown in Equation (25.20). The above connection of Score Matching and Contrastive Divergence  
 12 is a natural consequence of the connection between those two statistical divergences, as characterized  
 13 by *de Bruijin's identity* [Cov99; Lyu12]:  
 14

15

$$\frac{d}{dt} D_{KL}(q_t(\tilde{\mathbf{x}}) \| p_{\theta,t}(\tilde{\mathbf{x}})) = -\frac{1}{2} D_F(q_t(\tilde{\mathbf{x}}) \| p_{\theta,t}(\tilde{\mathbf{x}})).$$

16

17 Here  $q_t(\tilde{\mathbf{x}})$  and  $p_{\theta,t}(\tilde{\mathbf{x}})$  denote smoothed versions of  $p_{\text{data}}(\mathbf{x})$  and  $p_{\theta}(\mathbf{x})$ , resulting from adding  
 18 Gaussian noise to  $\mathbf{x}$  with variance  $t$ ; i.e.,  $\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, t\mathbf{I})$ .  
 19

### 20 25.3.5 Score-Based Generative Models

21

22 One typical application of EBMs is creating new samples that are similar to the training data. Towards  
 23 this end, we can first train an EBM with Score Matching, and then sample from it with MCMC  
 24 approaches. Many efficient sampling methods for EBMs, such as Langevin MCMC, rely on just the  
 25 score of the EBM (see Equation (25.19)). In addition, Score Matching objectives (Equations (25.31),  
 26 (25.33) and (25.37)) depend solely on the scores of EBMs. Therefore, we only need a model for  
 27 the score when training with Score Matching and sampling with score-based MCMC, and do not  
 28 have to model the energy explicitly. (Note that this loses the fact that the score is derived from the  
 29 derivative of the scalar energy, which can be a useful constraint.) By building such a score model, we  
 30 save the gradient computation of EBMs and can make training and sampling more efficient. These  
 31 kind of models are named **score-based generative models** [SE19; SE20b; Son+21]. (See also  
 32 Section 26.4.1 for a discussion of the related approach to denoising diffusion probabilistic models.)

33 We can optimize the score function  $s_{\theta}(\mathbf{x})$  using score matching, sliced score matching, or denoising  
 34 score matching. Figure 25.3 gives a simple example in 2d. In Figure 25.3a, we show the **swiss roll**  
 35 dataset. We estimate the score function by fitting an MLP with 2 hidden layers, each with 128  
 36 hidden units. In Figure 25.3b, we showed the output of the network after training for 10,000 steps of  
 37 SGD. We see that there are no major false negatives (since wherever there is high data the gradient  
 38 field is zero, as it should be), but there are some false positives (since some regions of zero gradient  
 39 do not correspond to data regions). The comparison of the predicted outputs with the empirical  
 40 data density is shown more clearly in Figure 25.3c. In Figure 25.3d, we show some samples from the  
 41 learned model.

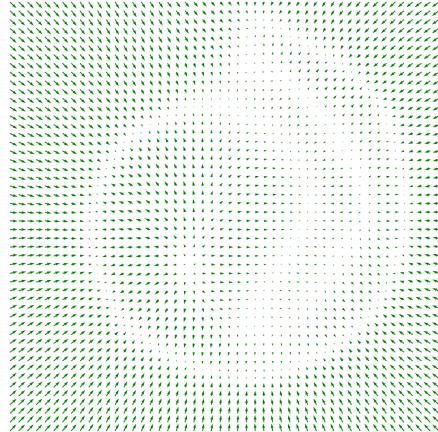
42

#### 43 25.3.5.1 Adding noise at multiple scales

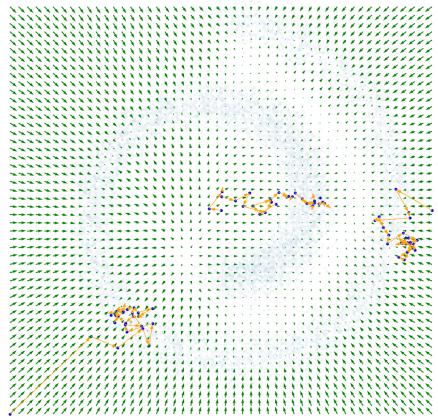
44

45 In general, score matching can have difficulty when there are regions of low data density. To see this,  
 46 suppose  $p_{\text{data}}(\mathbf{x}) = \pi p_0(\mathbf{x}) + (1 - \pi)p_1(\mathbf{x})$ . Let  $\mathcal{S}_0 := \{\mathbf{x} \mid p_0(\mathbf{x}) > 0\}$  and  $\mathcal{S}_1 := \{\mathbf{x} \mid p_1(\mathbf{x}) > 0\}$  be  
 47

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21



(a)



(c)

Figure 25.3: Fitting a score-based generative model to the 2d swiss roll dataset. (a) Training set. (b) Learned score function trained using the basic score matching. (c) Superposition of learned score function and empirical density. (d) Langevin sampling applied to the learned model. We show 3 different trajectories, each of length 25. Generated by `score_matching_swiss_roll.ipynb`.

43  
44  
45  
46  
47

1 the supports of  $p_0(\mathbf{x})$  and  $p_1(\mathbf{x})$  respectively. When they are disjoint from each other, the score of  
2  $p_{\text{data}}(\mathbf{x})$  is given by  
3

$$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) = \begin{cases} \nabla_{\mathbf{x}} \log p_0(\mathbf{x}), & \mathbf{x} \in \mathcal{S}_0 \\ \nabla_{\mathbf{x}} \log p_1(\mathbf{x}), & \mathbf{x} \in \mathcal{S}_1, \end{cases} \quad (25.45)$$

4  
5 which does not depend on the weight  $\pi$ . Hence score matching cannot correctly recover the true  
6 distribution. Furthermore, Langevin sampling will have difficulty traversing between modes. (In  
7 practice this will happen even when the different modes only have approximately disjoint supports.)  
8

9 Song and Ermon [SE19; SE20b] and Song et al. [Son+21] overcome this difficulty by perturbing  
10 training data with different scales of noise. Specifically, they use  
11

$$q_{\sigma}(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2 \mathbf{I}) \quad (25.46)$$

$$q_{\sigma}(\tilde{\mathbf{x}}) = \int p_{\text{data}}(\mathbf{x}) q_{\sigma}(\tilde{\mathbf{x}}|\mathbf{x}) d\mathbf{x} \quad (25.47)$$

12 For a large noise perturbation, different modes are connected due to added noise, and the estimated  
13 weights between them are therefore accurate. For a small noise perturbation, different modes are  
14 more disconnected, but the noise-perturbed distribution is closer to the original unperturbed data  
15 distribution. Using a sampling method such as annealed Langevin dynamics [SE19; SE20b; Son+21]  
16 or diffusion sampling [SD+15a; HJA20; Son+21], we can sample from the most noise-perturbed  
17 distribution first, then smoothly reduce the magnitude of noise scales until reaching the smallest one.  
18 This procedure helps combine information from all noise scales, and maintains the correct estimation  
19 of weights from higher noise perturbations when sampling from smaller ones.

20 In practice, all score models share weights and are implemented with a single neural network  
21 conditioned on the noise scale; this is called a **Noise Conditional Score Network**, and has the  
22 form  $s_{\theta}(\mathbf{x}, \sigma)$ . Scores of different scales are estimated by training a mixture of Score Matching  
23 objectives, one per noise scale. If we use the denoising score matching objective in Equation (25.33),  
24 we get

$$\mathcal{L}(\theta; \sigma) = \mathbb{E}_{q(\mathbf{x}, \tilde{\mathbf{x}})} \left[ \frac{1}{2} \|\nabla_{\mathbf{x}} \log p_{\theta}(\tilde{\mathbf{x}}, \sigma) - \nabla_{\mathbf{x}} \log q(\tilde{\mathbf{x}}|\mathbf{x})\|_2^2 \right] \quad (25.48)$$

$$= \frac{1}{2} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \sigma^2 \mathbf{I})} \left\{ \left\| s_{\theta}(\tilde{\mathbf{x}}, \sigma) + \frac{(\tilde{\mathbf{x}} - \mathbf{x})}{\sigma^2} \right\|_2^2 \right\} \quad (25.49)$$

25 These losses are combined in a weighted fashion using  
26

$$\mathcal{L}(\theta; \sigma_{1:L}) = \frac{1}{L} \sum_{i=1}^L \lambda(\sigma_i) \mathcal{L}(\theta; \sigma_i) \quad (25.50)$$

27 where we choose  $\sigma_1 > \sigma_2 > \dots > \sigma_L$ , and the weighting term satisfies  $\lambda(\sigma) > 0$ .

28 Empirically Song and Ermon [SE19] found that setting  $\lambda(\sigma_i) = \sigma_i$  works well. To set the  $\sigma_o$ ,  
29 we choose  $\sigma_1$  small enough such that  $p_{\sigma_1}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$ , and we choose  $\sigma_L$  large enough that  
30  $p_{\sigma_L}(\mathbf{x}) \approx \mathcal{N}(\mathbf{x}|\mathbf{0}, \sigma_L^2 \mathbf{I})$ . (See [SE20b] for further discussion.)

31 In [Son+21], this technique is extended to an infinite number of noise levels, by working with  
32 stochastic differential equations in continuous time. At the time of writing (May 2021), this method  
33

is amongst the best generative modeling approaches for high-resolution image generation (see samples in Figure 21.2) and audio generation [Che+20b].

## 25.4 Noise Contrastive Estimation

Another principle for learning the parameters of EBMs is **Noise Contrastive Estimation** (NCE), introduced by [GH10]. It is based on the idea that we can learn an Energy-Based Model by contrasting it with another distribution with known density.

Let  $p_{\text{data}}(\mathbf{x})$  be our data distribution, and let  $p_n(\mathbf{x})$  be a chosen distribution with known density, called a noise distribution. This noise distribution is usually simple and has a tractable PDF, like  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ , such that we can compute the PDF and generate samples from it efficiently. Strategies exist to learn the noise distribution, as referenced below. Furthermore, let  $y$  be a binary variable with Bernoulli distribution, which we use to define a mixture distribution of noise and data:  $p_{n,\text{data}}(\mathbf{x}) = p(y=0)p_n(\mathbf{x}) + p(y=1)p_{\text{data}}(\mathbf{x})$ . According to the Bayes' rule, given a sample  $\mathbf{x}$  from this mixture, the posterior probability of  $y=0$  is

$$p_{n,\text{data}}(y=0 \mid \mathbf{x}) = \frac{p_{n,\text{data}}(\mathbf{x} \mid y=0)p(y=0)}{p_{n,\text{data}}(\mathbf{x})} = \frac{p_n(\mathbf{x})}{p_n(\mathbf{x}) + \nu p_{\text{data}}(\mathbf{x})} \quad (25.51)$$

where  $\nu = p(y=1)/p(y=0)$ .

Let our Energy-Based Model  $p_{\boldsymbol{\theta}}(\mathbf{x})$  be defined as:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \exp(-E_{\boldsymbol{\theta}}(\mathbf{x}))/Z_{\boldsymbol{\theta}} \quad (25.52)$$

Contrary to most other EBMs,  $Z_{\boldsymbol{\theta}}$  is treated as a learnable (scalar) parameter in NCE. Given this model, similar to the mixture of noise and data above, we can define a mixture of noise and the model distribution:  $p_{n,\boldsymbol{\theta}}(\mathbf{x}) = p(y=0)p_n(\mathbf{x}) + p(y=1)p_{\boldsymbol{\theta}}(\mathbf{x})$ . The posterior probability of  $y=0$  given this noise/model mixture is:

$$p_{n,\boldsymbol{\theta}}(y=0 \mid \mathbf{x}) = \frac{p_n(\mathbf{x})}{p_n(\mathbf{x}) + \nu p_{\boldsymbol{\theta}}(\mathbf{x})} \quad (25.53)$$

In NCE, we indirectly fit  $p_{\boldsymbol{\theta}}(\mathbf{x})$  to  $p_{\text{data}}(\mathbf{x})$  by fitting  $p_{n,\boldsymbol{\theta}}(y \mid \mathbf{x})$  to  $p_{n,\text{data}}(y \mid \mathbf{x})$  through a standard conditional maximum likelihood objective:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \mathbb{E}_{p_{n,\text{data}}(\mathbf{x})} [D_{KL}(p_{n,\text{data}}(y \mid \mathbf{x}) \parallel p_{n,\boldsymbol{\theta}}(y \mid \mathbf{x}))] \quad (25.54)$$

$$= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \mathbb{E}_{p_{n,\text{data}}(\mathbf{x}, y)} [\log p_{n,\boldsymbol{\theta}}(y \mid \mathbf{x})], \quad (25.55)$$

which can be solved using stochastic gradient ascent. Just like any other deep classifier, when the model is sufficiently powerful,  $p_{n,\boldsymbol{\theta}^*}(y \mid \mathbf{x})$  will match  $p_{n,\text{data}}(y \mid \mathbf{x})$  at the optimum. In that case:

$$p_{n,\boldsymbol{\theta}^*}(y=0 \mid \mathbf{x}) \equiv p_{n,\text{data}}(y=0 \mid \mathbf{x}) \quad (25.56)$$

$$\iff \frac{p_n(\mathbf{x})}{p_n(\mathbf{x}) + \nu p_{\boldsymbol{\theta}^*}(\mathbf{x})} \equiv \frac{p_n(\mathbf{x})}{p_n(\mathbf{x}) + \nu p_{\text{data}}(\mathbf{x})} \quad (25.57)$$

$$\iff p_{\boldsymbol{\theta}^*}(\mathbf{x}) \equiv p_{\text{data}}(\mathbf{x}) \quad (25.58)$$

Consequently,  $E_{\theta^*}(\mathbf{x})$  is an unnormalized energy function that matches the data distribution  $p_{\text{data}}(\mathbf{x})$ , and  $Z_{\theta^*}$  is the corresponding normalizing constant.

As one unique feature that Contrastive Divergence and Score Matching do not have, NCE provides the normalizing constant of an Energy-Based Model as a by-product of its training procedure. When the EBM is very expressive, *e.g.*, a deep neural network with many parameters, we can assume it is able to approximate a normalized probability density and absorb  $Z_\theta$  into the parameters of  $E_\theta(\mathbf{x})$  [MT12], or equivalently, fixing  $Z_\theta = 1$ . The resulting EBM trained with NCE will be self-normalized, *i.e.*, having a normalizing constant close to 1.

In practice, choosing the right noise distribution  $p_n(\mathbf{x})$  is critical to the success of NCE, especially for structured and high-dimensional data. As argued in Gutmann and Hirayama [GH12b], NCE works the best when the noise distribution is close to the data distribution (but not exactly the same). Many methods have been proposed to automatically tune the noise distribution, such as Adversarial Contrastive Estimation [BLC18], Conditional NCE [CG18] and Flow Contrastive Estimation [Gao+20]. NCE can be further generalized using Bregman divergences (Section 6.5.1), where the formulation introduced here reduces to a special case.

17

### 25.4.1 Connection to Score Matching

Noise Contrastive Estimation provides a family of objectives that vary for different  $p_n(\mathbf{x})$  and  $\nu$ . This flexibility may allow adaptation to special properties of a task with hand-tuned  $p_n(\mathbf{x})$  and  $\nu$ , and may also give a unified perspective for different approaches. In particular, when using an appropriate  $p_n(\mathbf{x})$  and a slightly different parameterization of  $p_{n,\theta}(y | \mathbf{x})$ , we can recover Score Matching from NCE [GH12b].

Specifically, we choose the noise distribution  $p_n(\mathbf{x})$  to be a perturbed data distribution: given a small (deterministic) vector  $\mathbf{v}$ , let  $p_n(\mathbf{x}) = p_{\text{data}}(\mathbf{x} - \mathbf{v})$ . It is efficient to sample from this  $p_n(\mathbf{x})$ , since we can first draw any datapoint  $\mathbf{x}' \sim p_{\text{data}}(\mathbf{x}')$  and then compute  $\mathbf{x} = \mathbf{x}' + \mathbf{v}$ . It is, however, difficult to evaluate the density of  $p_n(\mathbf{x})$  because  $p_{\text{data}}(\mathbf{x})$  is unknown. Since the original parameterization of  $p_{n,\theta}(y | \mathbf{x})$  in NCE (Equation (25.53)) depends on the PDF of  $p_n(\mathbf{x})$ , we cannot directly apply the standard NCE objective. Instead, we replace  $p_n(\mathbf{x})$  with  $p_\theta(\mathbf{x} - \mathbf{v})$  and parameterize  $p_{n,\theta}(y = 0 | \mathbf{x})$  with the following form

$$p_{n,\theta}(y = 0 | \mathbf{x}) := \frac{p_\theta(\mathbf{x} - \mathbf{v})}{p_\theta(\mathbf{x}) + p_\theta(\mathbf{x} - \mathbf{v})} \quad (25.59)$$

In this case, the NCE objective (Equation (25.55)) reduces to:

$$\theta^* = \operatorname{argmin}_\theta \mathbb{E}_{p_{\text{data}}(\mathbf{x})} [\log(1 + \exp(E_\theta(\mathbf{x}) - E_\theta(\mathbf{x} - \mathbf{v})) + \log(1 + \exp(E_\theta(\mathbf{x}) - E_\theta(\mathbf{x} + \mathbf{v})))] \quad (25.60)$$

At  $\theta^*$ , we have a solution where:

$$p_{n,\theta^*}(y = 0 | \mathbf{x}) \equiv p_{\text{data}}(y = 0 | \mathbf{x}) \quad (25.61)$$

$$\Rightarrow \frac{p_{\theta^*}(\mathbf{x} - \mathbf{v})}{p_{\theta^*}(\mathbf{x}) + p_{\theta^*}(\mathbf{x} - \mathbf{v})} \equiv \frac{p_{\text{data}}(\mathbf{x} - \mathbf{v})}{p_{\text{data}}(\mathbf{x}) + p_{\text{data}}(\mathbf{x} - \mathbf{v})} \quad (25.62)$$

which implies that  $p_{\theta^*}(\mathbf{x}) \equiv p_{\text{data}}(\mathbf{x})$ , *i.e.*, our model matches the data distribution.

47

As noted in Gutmann and Hirayama [GH12b] and Song et al. [Son+19], when  $\|\mathbf{v}\|_2 \approx 0$ , the NCE objective Equation (25.55) has the following equivalent form by Taylor expansion

$$\operatorname{argmin}_{\boldsymbol{\theta}} \frac{1}{4} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[ \frac{1}{2} \sum_{i=1}^d \left( \frac{\partial E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_i} v_i \right)^2 + \sum_{i=1}^d \sum_{j=1}^d \frac{\partial^2 E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_i \partial x_j} v_i v_j \right] + 2 \log 2 + o(\|\mathbf{v}\|_2^2). \quad (25.63)$$

Comparing against Equation (25.37), we immediately see that the above objective equals that of SSM, if we ignore small additional terms hidden in  $o(\|\mathbf{v}\|_2^2)$  and take the expectation with respect to  $\mathbf{v}$  over a user-specified distribution  $p(\mathbf{v})$ .

## 25.5 Other Methods

Aside from MCMC-based training, Score Matching and Noise Contrastive Estimation, there are also other methods for learning EBMs. Below we briefly survey some examples of them. Interested readers can learn more details from references therein.

### 25.5.1 Minimizing Differences/Derivatives of KL Divergences

The overarching strategy for learning probabilistic models from data is to minimize the KL divergence between data and model distributions. However, because the normalizing constants of EBMs are typically intractable, it is hard to directly evaluate the KL divergence when the model is an EBM (see the discussion in Section 25.2.1). One generic idea that has frequently circumvented this difficulty is to consider differences/derivatives of KL divergences. It turns out that the unknown partition functions of EBMs are often cancelled out after taking the difference of two closely related KL divergences, or computing the derivatives.

Typical examples of this strategy include minimum velocity learning [Mov08; Wan+20c], minimum probability flow [SDBD11] and minimum KL contraction [Lyu11], to name a few. In minimum velocity learning and minimum probability flow, a Markov chain is designed such that it starts from the data distribution  $p_{\text{data}}(\mathbf{x})$  and converges to the EBM distribution  $p_{\boldsymbol{\theta}}(\mathbf{x}) = e^{-E_{\boldsymbol{\theta}}(\mathbf{x})}/Z_{\boldsymbol{\theta}}$ . Specifically, the Markov chain satisfies  $p_0(\mathbf{x}) \equiv p_{\text{data}}(\mathbf{x})$  and  $p_{\infty}(\mathbf{x}) \equiv p_{\boldsymbol{\theta}}(\mathbf{x})$ , where we denote by  $p_t(\mathbf{x})$  the state distribution at time  $t \geq 0$ .

This Markov chain will evolve towards  $p_{\boldsymbol{\theta}}(\mathbf{x})$  unless  $p_{\text{data}}(\mathbf{x}) \equiv p_{\boldsymbol{\theta}}(\mathbf{x})$ . Therefore, we can fit the EBM distribution  $p_{\boldsymbol{\theta}}(\mathbf{x})$  to  $p_{\text{data}}(\mathbf{x})$  by minimizing the modulus of the “velocity” of this evolution, defined by

$$\frac{d}{dt} \mathbb{KL}(p_t(\mathbf{x}) \parallel p_{\boldsymbol{\theta}}(\mathbf{x})) \Big|_{t=0} \quad \text{or} \quad \frac{d}{dt} \mathbb{KL}(p_{\text{data}}(\mathbf{x}) \parallel p_t(\mathbf{x})) \Big|_{t=0} \quad (25.64)$$

in minimum velocity learning and minimum probability flow respectively. These objectives typically do not require computing the normalizing constant  $Z_{\boldsymbol{\theta}}$ .

In minimum KL contraction [Lyu11], a distribution transformation  $\Phi$  is chosen such that

$$\mathbb{KL}(p(\mathbf{x}) \parallel q(\mathbf{x})) \geq \mathbb{KL}(\Phi\{p(\mathbf{x})\} \parallel \Phi\{q(\mathbf{x})\}) \quad (25.65)$$

with equality if and only if  $p(\mathbf{x}) = q(\mathbf{x})$ . We can leverage this  $\Phi$  to train an EBM, by minimizing

$$\mathbb{KL}(p_{\text{data}}(\mathbf{x}) \parallel p_{\boldsymbol{\theta}}(\mathbf{x})) - \mathbb{KL}(\Phi\{p_{\text{data}}(\mathbf{x})\} \parallel \Phi\{p_{\boldsymbol{\theta}}(\mathbf{x})\}). \quad (25.66)$$

1 This objective does not require computing the partition function  $Z_\theta$  whenever  $\Phi$  is linear.

2 Minimum velocity learning, minimum probability flow, and minimum KL contraction can all be  
3 viewed as generalizations to Score Matching and Noise Contrastive Estimation [Mov08; SDBD11;  
4 Lyu11].  
5

### 7 25.5.2 Minimizing the Stein Discrepancy

9 We can train EBMs by minimizing the Stein discrepancy, defined by

$$\underline{11} \quad D_{\text{Stein}}(p_{\text{data}}(\mathbf{x}) \parallel p_\theta(\mathbf{x})) := \sup_{\mathbf{f} \in \mathcal{F}} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} [\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})^\top \mathbf{f}(\mathbf{x}) + \text{trace}(\nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x}))], \quad (25.67)$$

14 where  $\mathcal{F}$  is a family of vector-valued functions, and  $\nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x})$  denotes the Jacobian of  $\mathbf{f}(\mathbf{x})$ . With  
15 some regularity conditions [GM15; LLJ16], we have  $D_S(p_{\text{data}}(\mathbf{x}) \parallel p_\theta(\mathbf{x})) \geq 0$ , where the equality  
16 holds if and only if  $p_{\text{data}}(\mathbf{x}) \equiv p_\theta(\mathbf{x})$ . Similar to Score Matching (Equation (25.31)), the objective  
17 Equation (25.67) only involves the score function of  $p_\theta(\mathbf{x})$ , and does not require computing the EBM's  
18 partition function. Still, the trace term in Equation (25.67) may demand expensive computation,  
19 and does not scale well to high dimensional data.

20 There are two common methods that sidestep this difficulty. Gorham and Mackey [GM15] and  
21 Liu, Lee, and Jordan [LLJ16] discovered that when  $\mathcal{F}$  is a unit ball in a Reproducing Kernel Hilbert  
22 Space (RKHS) with a fixed kernel, the Stein discrepancy becomes kernelized Stein discrepancy, where  
23 the trace term is a constant and does not affect optimization. Otherwise,  $\text{trace}(\nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x}))$  can be  
24 approximated with the Skilling-Hutchinson trace estimator [Ski89; Hut89; Gra+20c].  
25

### 26 25.5.3 Adversarial Training

28 Recall from Section 25.2.1 that when training EBMs with maximum likelihood estimation (MLE),  
29 we need to sample from the EBM per training iteration. However, sampling using multiple MCMC  
30 steps is expensive and requires careful tuning of the Markov chain. One way to avoid this difficulty is  
31 to use non-MLE methods that do not need sampling, such as Score Matching and Noise Contrastive  
32 Estimation. Here we introduce another family of methods that sidestep costly MCMC sampling by  
33 learning an auxiliary model through adversarial training, which allows fast sampling.  
34

34 Using the definition of EBMs, we can rewrite the maximum likelihood objective by introducing a  
35 variational distribution  $q_\phi(\mathbf{x})$  parameterized by  $\phi$ :

$$\begin{aligned} \underline{37} \quad \mathbb{E}_{p_{\text{data}}(\mathbf{x})} [\log p_\theta(\mathbf{x})] &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} [-E_\theta(\mathbf{x})] - \log Z_\theta \\ \underline{38} \quad &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} [-E_\theta(\mathbf{x})] - \log \int e^{-E_\theta(\mathbf{x})} d\mathbf{x} \\ \underline{39} \quad &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} [-E_\theta(\mathbf{x})] - \log \int q_\phi(\mathbf{x}) \frac{e^{-E_\theta(\mathbf{x})}}{q_\phi(\mathbf{x})} d\mathbf{x} \\ \underline{40} \quad &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} [-E_\theta(\mathbf{x})] - \int q_\phi(\mathbf{x}) \log \frac{e^{-E_\theta(\mathbf{x})}}{q_\phi(\mathbf{x})} d\mathbf{x} \\ \underline{41} \quad &\stackrel{(i)}{\leq} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} [-E_\theta(\mathbf{x})] - \int q_\phi(\mathbf{x}) \log \frac{e^{-E_\theta(\mathbf{x})}}{q_\phi(\mathbf{x})} d\mathbf{x} \\ \underline{42} \quad &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} [-E_\theta(\mathbf{x})] - \mathbb{E}_{q_\phi(\mathbf{x})} [-E_\theta(\mathbf{x})] - H(q_\phi(\mathbf{x})), \end{aligned} \quad (25.68)$$

where  $H(q_\phi(\mathbf{x}))$  denotes the entropy of  $q_\phi(\mathbf{x})$ . Step (i) is due to Jensen’s inequality. Equation (25.68) provides an upper bound to the expected log-likelihood. For EBM training, we can first minimize the upper bound Equation (25.68) with respect to  $q_\phi(\mathbf{x})$  so that it is closer to the likelihood objective, and then maximize Equation (25.68) with respect to  $E_\theta(\mathbf{x})$  as a surrogate for maximizing likelihood. This amounts to using the following maximin objective

$$\max_{\theta} \min_{\phi} \mathbb{E}_{q_\phi(\mathbf{x})}[E_\theta(\mathbf{x})] - \mathbb{E}_{p_{\text{data}}(\mathbf{x})}[E_\theta(\mathbf{x})] - H(q_\phi(\mathbf{x})). \quad (25.69)$$

Optimizing the above objective is similar to training GANs (Chapter 27), and can be achieved by adversarial training. The variational distribution  $q_\phi(\mathbf{x})$  should allow both fast sampling and efficient entropy evaluation to make Equation (25.69) tractable. This limits the model family of  $q_\phi(\mathbf{x})$ , and usually restricts our choice to invertible probabilistic models, such as inverse autoregressive flow (Section 24.2.4.3). See Dai et al. [Dai+19b] for an example on designing  $q_\phi(\mathbf{x})$  and training EBMs with Equation (25.69).

Kim and Bengio [KB16] and Zhai et al. [Zha+16] propose to represent  $q_\phi(\mathbf{x})$  with neural samplers, like the generator of GANs. A neural sampler is a deterministic mapping  $g_\phi$  that maps a random Gaussian noise  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  directly to a sample  $\mathbf{x} = g_\phi(\mathbf{z})$ . When using a neural sampler as  $q_\phi(\mathbf{x})$ , it is efficient to draw samples through the deterministic mapping, but  $H(q_\phi(\mathbf{x}))$  is intractable since the density of  $q_\phi(\mathbf{x})$  is unknown. Kim and Bengio [KB16] and Zhai et al. [Zha+16] propose several heuristics to approximate this entropy function. Kumar et al. [Kum+19c] propose to estimate the entropy through its connection to mutual information:  $H(q_\phi(\mathbf{z})) = I(g_\phi(\mathbf{z}), \mathbf{z})$ , which can be estimated from samples with variational lower bounds [NWJ10b; NCT16b]. Dai et al. [Dai+19a] noticed that when defining  $p_\theta(\mathbf{x}) = p_0(\mathbf{x})e^{-E_\theta(\mathbf{x})}/Z_\theta$ , with  $p_0(\mathbf{x})$  being a fixed base distribution, the entropy term  $-H(q_\phi(\mathbf{x}))$  in Equation (25.69) equates  $\text{KL}(q_\phi(\mathbf{x}) \parallel p_0(\mathbf{x}))$ , which can also be approximated with variational lower bounds using samples from  $q_\phi(\mathbf{x})$  and  $p_0(\mathbf{x})$ , without requiring the density of  $q_\phi(\mathbf{x})$ .

Grathwohl et al. [Gra+20a] represent  $q_\phi(\mathbf{x})$  as a noisy neural sampler, where samples are obtained via  $g_\phi(\mathbf{z}) + \sigma\epsilon$ , assuming  $\mathbf{z}, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ . With a noisy neural sampler,  $\nabla_\phi H(q_\phi(\mathbf{x}))$  becomes particularly easy to estimate, which allows gradient-based optimization for the minimax objective in Equation (25.68). A related approach is proposed in Xie et al. [Xie+18], where authors train a noisy neural sampler with samples obtained from MCMC, and initialize new MCMC chains with samples generated from the neural sampler. This cooperative sampling scheme improves the convergence of MCMC, but may still require multiple MCMC steps for sample generation. It does not optimize the objective in Equation (25.68).

When using both adversarial training and MCMC sampling, Yu et al. [Yu+20] noticed that EBMs can be trained with an arbitrary  $f$ -divergence, including KL, reverse KL, total variation, Hellinger, etc.. The method proposed by Yu et al. [Yu+20] allows us to explore the trade-offs and inductive bias of different statistical divergences for more flexible EBM training.

40  
41  
42  
43  
44  
45  
46  
47



# 26 Denoising diffusion models

In this section, we consider a class of models called **denoising diffusion probabilistic models** or **DDPM**. This model was first introduced in [SD+15b] and has been extended in several subsequent papers (e.g., [HJA20; Son+21; DN21]).

The basic insight is the following: it can be hard to convert noise into structured data, but it is easy to convert structured data into noise. In particular, we can use a **forwards process** or **diffusion process** to gradually convert the observed data  $\mathbf{x} = \mathbf{x}_0$  into a noisy version  $\mathbf{z} = \mathbf{x}_T$  with no structure by passing the data through  $T$  steps of a stochastic encoder  $q(\mathbf{x}_t | \mathbf{x}_{t-1})$ . We then learn a **reverse process** to undo this, by passing the noise through  $T$  steps of a decoder  $p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t)$  until we generate  $\mathbf{x}_0$ . See Figure 26.1 for an overall sketch of the approach, and Section 26.1 for the details. (Note that diffusion models is a very active field of research, so this section is just a brief summary. For some JAX tutorials, see [https://github.com/acids-ircam/diffusion\\_models](https://github.com/acids-ircam/diffusion_models).)

## 26.1 Model definition

In this section, we describe DDPMs in more detail. The diffusion process is designed to add a small amount of noise to the data at each step. For real-valued data, we can use

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t | \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}) \quad (26.1)$$

where  $\beta_t$  is a small positive term that controls the variance of the noise. We can sample from this by first sampling  $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ , and then setting

$$\mathbf{x}_t = \sqrt{1 - \beta_t} \mathbf{x}_{t-1} + \beta_t \boldsymbol{\epsilon}_t \quad (26.2)$$

By first scaling down  $\mathbf{x}_{t-1}$  before adding noise, the overall variance of each intermediate version of the data remains bounded. (We assume the observed data is already standardized.)

For binary data, we can use the following “noisy channel” model:

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \text{Ber}(\mathbf{x}_t | (1 - \beta_t) \mathbf{x}_{t-1} + 0.5 \beta_t) \quad (26.3)$$

where we use the following shorthand for a factored Bernoulli distribution:

$$\text{Ber}(\mathbf{x} | \boldsymbol{\mu}) = \prod_{d=1}^D \text{Ber}(x_d | \mu_d) \quad (26.4)$$

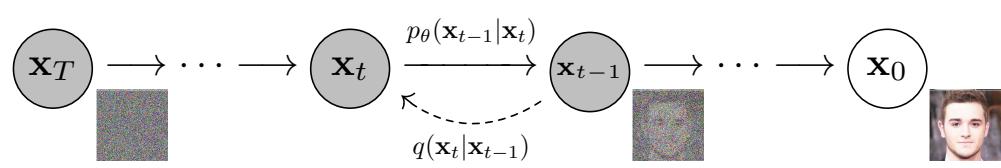


Figure 26.1: Denoising diffusion probabilistic model. The forwards diffusion process,  $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ , implements the (non-learned) inference network; this just adds Gaussian noise at each step. The reverse diffusion process,  $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ , implements the decoder; this is a learned Gaussian model. From Figure 1 of [HJA20]. Used with kind permission of Jonathan Ho.

11

12

If we keep sampling from the diffusion process, we eventually get to the stationary distribution, defined by

$$\pi(\mathbf{x}) = \int d\mathbf{x}' q(\mathbf{x}|\mathbf{x}') \quad (26.5)$$

17

We usually choose  $\pi(\mathbf{x})$  to be a simple maximum entropy distribution, such as an isotropic Gaussian (for real data), or product of uniform Bernoullis (for binary data). We choose the noise schedule  $\beta_{1:T}$  so that  $q(\mathbf{x}_T|\mathbf{x}_0) \approx \pi(\mathbf{x}_T)$ . That is, after  $T$  steps, we have “destroyed” all the information in the data. The overall encoder can be represented as follows:

22

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1}) \quad (26.6)$$

25

To generate from this model, we first sample a latent noise vector  $\mathbf{x}_T \sim \pi(\mathbf{x}_T)$ , and then gradually “denoise” it by sampling from the reverse process,  $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ , until we get the final output  $\mathbf{x}_0$ . (Note that each latent sample has the same size as the observed data, so there is no dimensionality reduction.)

The reverse conditionals  $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$  are designed to “undo” the effects of noise added by the diffusion process. For small enough  $\beta_t$ , one can show that the reverse conditionals will have the same form as the forward conditionals. Thus for real-valued data we can use

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}|f_\mu(\mathbf{x}_t, t; \boldsymbol{\theta}), f_\Sigma(\mathbf{x}_t, t; \boldsymbol{\theta})) \quad (26.7)$$

35

and for binary data, we can use

36

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \text{Ber}(\mathbf{x}_{t-1}|f(\mathbf{x}_t, t; \boldsymbol{\theta})) \quad (26.8)$$

38

The overall decoder can be represented as follows:

40

$$p_\theta(\mathbf{x}_{0:T}) = \pi(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad (26.9)$$

43

The marginal probability of the observed data (the output of the model) is given by

45

$$p_\theta(\mathbf{x}_0) = \int d\mathbf{x}_{1:T} p(\mathbf{x}_{0:T}) \quad (26.10)$$

47

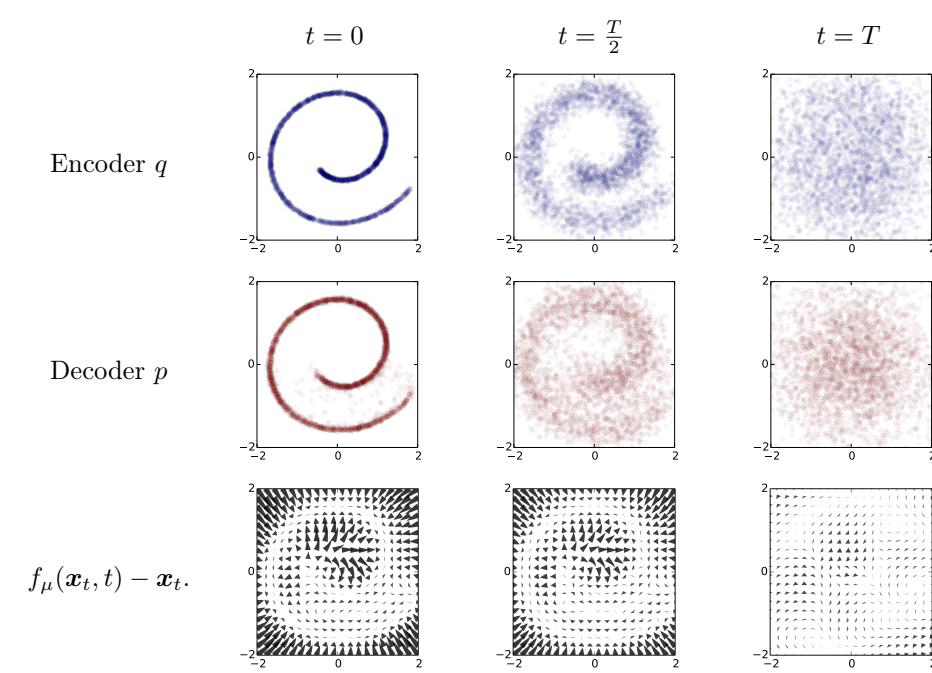


Figure 26.2: Diffusion on 2d swiss roll. Top row: samples from the encoder  $q(\mathbf{x}_{0:T})$  at 3 different time slices. (Here  $q(\mathbf{x}_0)$  is the input data.) Middle row: samples from the generator  $p_{\theta}(\mathbf{x}_{0:T})$  at 3 different time slices. Bottom row: Visualization of the residual or drift term,  $f_{\mu}(\mathbf{x}_t, t) - \mathbf{x}_t$ . We see that this resembles the score function, in that it points towards regions of high data density (see Section 26.4.1). From Figure 1 of [SD+15b]. Used with kind permission of Jascha Sohl-Dickstein.

This is intractable to compute. However, we can use a method similar to annealed importance sampling (Section 11.5.4) to rewrite this as follows:

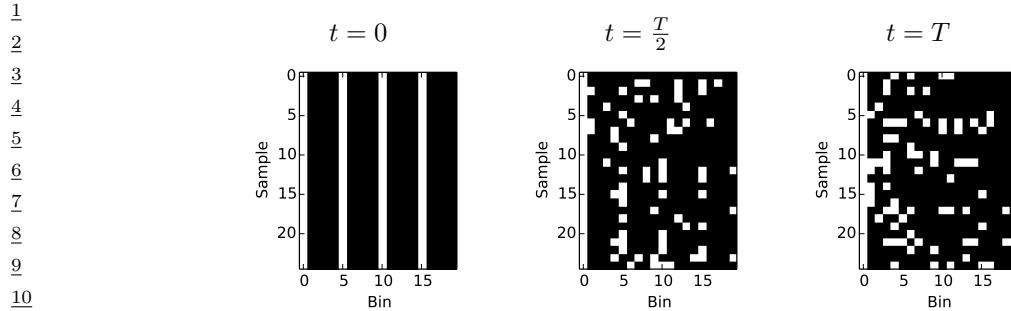
$$p(\mathbf{x}_0) = \int d\mathbf{x}_{1:T} p_{\theta}(\mathbf{x}_{0:T}) \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} = \int d\mathbf{x}_{1:T} q(\mathbf{x}_{1:T}|\mathbf{x}_0) \frac{p_{\theta}(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \quad (26.11)$$

$$= \int d\mathbf{x}_{1:T} q(\mathbf{x}_{1:T}|\mathbf{x}_0) \pi(\mathbf{x}_T) \prod_{t=1}^T \frac{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \quad (26.12)$$

which we can approximate by Monte Carlo sampling. We will leverage this result below.

## 26.2 Examples

In Figure 26.2, we show an example of a Gaussian DDPM fit to the 2d swiss roll dataset. The model uses  $T = 40$  steps. The generator network  $f(\mathbf{x}_t, t; \theta)$  was defined to be a radial basis function network with 16 hidden units. It has two output heads, one for  $\mu$ , and one for the diagonal  $\Sigma$ . (The latter uses the sigmoid nonlinearity to ensure the variances are between 0 and 1.) A separate output head



12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47

Figure 26.3: Diffusion on binary data. Each row contains an independent sample from  $p(\mathbf{x}_{0:T})$  at 3 different time steps. From Figure 2 of [SD+15b]. Used with kind permission of Jascha Sohl-Dickstein.

was learned for each time step  $t$ , but all the other parameters were shared across heads and across time.

In Figure 26.3, we show an example of a binary DDPM fit to a synthetic distribution consisting of bit vectors of length 20, in which every 5th time step is on. The model uses  $T = 2000$  time steps and a prior of the form  $\pi(\mathbf{x}_T) = \text{Ber}(\mathbf{x}_T|0.2)$ . The generator network  $f(\mathbf{x}_t, t; \theta)$  was defined to be an MLP with sigmoid nonlinearities, 20 input units and 3 hidden layers, each with 50 units. The final sigmoid later was learned separately for each time step  $t$ , but all the other parameters were shared across time.

In more recent papers (e.g., [HJA20; Son+21; DN21]), DDPMs have been scaled up to much more complex image datasets, such as CIFAR, CelebA and Imagenet. The resulting generated images have a visual realism which matches or exceeds other methods, such as GANs (Chapter 27). In addition, the samples are more diverse, and the log likelihoods are higher. These results rely on more complex neural network architectures, borrowing pieces from U-net, PixelCNN, transformers, etc.

### 26.3 Model training

To fit the generator  $p$ , we minimize the cross entropy between the empirical distribution  $p_{\mathcal{D}}$  and the model  $p$ , or equivalently we maximize

$$L = \int d\mathbf{x}_0 p_{\mathcal{D}}(\mathbf{x}_0) \log p_{\theta}(\mathbf{x}_0) \tag{26.13}$$

To simplify notation, let us define  $p_{\mathcal{D}}(\mathbf{x}) = q(\mathbf{x}_0)$ . Then we have

$$L = \int d\mathbf{x}_{0:T} q(\mathbf{x}_0) q(\mathbf{x}_{1:T}|\mathbf{x}_0) \pi(\mathbf{x}_T) \prod_{t=1}^T \frac{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \tag{26.14}$$

By Jensen's inequality we find that  $L \geq \bar{L}$ , where  $\bar{L}$  is the ELBO:

$$\bar{L} = \mathbb{E}_q \left[ \log \pi(\mathbf{x}_T) + \sum_{t=1}^T \log \frac{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] \tag{26.15}$$

We now proceed to rewrite the lower bound in Equation (26.15) in a simplified form that is more amenable to optimization. First note that

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) \quad (26.16)$$

since the diffusion process is Markov. Hence by Bayes rule, we have

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) = \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) q(\mathbf{x}_t | \mathbf{x}_0)}{q(\mathbf{x}_{t-1} | \mathbf{x}_0)} \quad (26.17)$$

Thus

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{x}_{0:T})} \left[ \log \pi(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t)}{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)} \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_0)}{q(\mathbf{x}_t | \mathbf{x}_0)} + \log \frac{p_{\theta}(\mathbf{x}_0 | \mathbf{x}_1)}{q(\mathbf{x}_1 | \mathbf{x}_0)} \right] \quad (26.18)$$

$$= \mathbb{E}_{q(\mathbf{x}_{0:T})} \left[ \log \pi(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t)}{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)} + \underbrace{\sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_0)}{q(\mathbf{x}_t | \mathbf{x}_0)}}_{*} + \log \frac{p_{\theta}(\mathbf{x}_0 | \mathbf{x}_1)}{q(\mathbf{x}_1 | \mathbf{x}_0)} \right] \quad (26.19)$$

The term marked \* is a telescoping sum, and can be simplified as follows:

$$* = \log q(\mathbf{x}_{T-1} | \mathbf{x}_0) + \log q(\mathbf{x}_{T-2} | \mathbf{x}_0) + \cdots + \log q(\mathbf{x}_1 | \mathbf{x}_0) \quad (26.20)$$

$$- \log q(\mathbf{x}_T | \mathbf{x}_0) - \log q(\mathbf{x}_{T-1} | \mathbf{x}_0) - \cdots - \log q(\mathbf{x}_2 | \mathbf{x}_0) \quad (26.21)$$

$$= \log q(\mathbf{x}_1 | \mathbf{x}_0) - \log q(\mathbf{x}_T | \mathbf{x}_0) \quad (26.22)$$

Hence

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{x}_{0:T})} \left[ \log \frac{\pi(\mathbf{x}_T)}{q(\mathbf{x}_T | \mathbf{x}_0)} + \sum_{t=2}^T \log \frac{p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t)}{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)} + \log p_{\theta}(\mathbf{x}_0 | \mathbf{x}_1) \right] \quad (26.23)$$

$$= -\mathbb{E}_{q(\mathbf{x}_0)} \left[ \underbrace{D_{\text{KL}}(q(\mathbf{x}_T | \mathbf{x}_0) \| \pi(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^T \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \| p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_{\theta}(\mathbf{x}_0 | \mathbf{x}_1)}_{L_0} \right] \quad (26.24)$$

The  $L_T$  term is a constant, since  $q$  has no free parameters. The  $L_0$  term can be optimized using the reparameterization trick. Finally, each  $L_t$  term can be computed analytically, as we discuss below. Furthermore, we can compute a stochastic approximation to the overall objective terms by sampling a single timestep at random, and just optimizing a single  $L_t$  term. This allows us to efficiently train very deep (large  $T$ ) models.

To compute the  $L_t$  terms, we will focus on the Gaussian case. First note that  $q(\mathbf{x}_t | \mathbf{x}_0)$  corresponds to  $t$  steps through a linear-Gaussian Markov chain, and so the result is a Gaussian. To derive its moments, let us define  $\alpha_t = 1 - \beta$  and  $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$ . Then we have

$$q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t | \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I}) \quad (26.25)$$

To compute  $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ , we use Bayes rule for Gaussians to get another Gaussian:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \propto q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)q(\mathbf{x}_{t-1}|\mathbf{x}_0) \quad (26.26)$$

$$= \mathcal{N}(\mathbf{x}_{t-1}|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}) \quad (26.27)$$

$$\tilde{\boldsymbol{\mu}}_t = \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad (26.28)$$

$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \quad (26.29)$$

Finally, recall from Equation (5.66) that the KL divergence between two Gaussian is

$$D_{\text{KL}}(\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_1, \Sigma_1) \| \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_2, \Sigma_2)) \\ = \frac{1}{2} \left[ \text{tr}(\Sigma_2^{-1}\Sigma_1) + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^\top \Sigma_2^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) - D + \log \left( \frac{\det(\Sigma_2)}{\det(\Sigma_1)} \right) \right] \quad (26.30)$$

Thus every  $L_t$  terms can be tractably computed.

To finish specifying the model, we have to define the noise schedule for  $\beta_t$  in the encoder  $q$ . It is possible to learn  $\beta_t$ , but it is more common to set it using a deterministic schedule. In [SD+15b], they define  $\beta_t = 1/(T-t+1)$ , which erases a fraction  $1/T$  of the information per step. In [HJA20], they set  $\beta_t$  to a constant.

## 26.4 Connections with other generative models

DDPMs are closely related to several other model types, as summarized in Figure 21.1. We summarize these connections below.

### 26.4.1 Connection with score matching

Suppose we use a Gaussian DDPM model of the form  $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}|\mu_\theta(\mathbf{x}_t, t), \sigma^2 \mathbf{I})$ . In this case, there is a very close connection between DDPM and score matching (Section 25.3.5).

To see this, note that we can rewrite the  $L_{t-1}$  term in the ELBO in Equation (26.24) as follows:

$$L_{t-1} = \mathbb{E}_{q(\mathbf{x}_0, \mathbf{x}_t)} \left[ \frac{1}{2\sigma_t^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 \right] + C \quad (26.31)$$

where  $C$  is a constant that does not depend on  $\theta$ , and  $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0)$  is the predicted mean value of  $\mathbf{x}_{t-1}$ . Since  $\mathbf{x}_t = \mathbf{x}_{t-1} + \sigma^2 \boldsymbol{\epsilon}_t$ , the network  $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$  needs to predict the noise term  $\boldsymbol{\epsilon}_t = \frac{\mathbf{x}_t - \mathbf{x}_{t-1}}{\sigma^2}$  if it is to reliably predict  $\mathbf{x}_{t-1}$ . But predicting the noise term is essentially what a denoising score matching method does (see Section 25.3.2 for details). In particular, if we use a noise distribution of the form  $p_\sigma(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t|\mathbf{x}_{t-1}, \sigma^2 \mathbf{I})$ , then the score function is given by

$$\nabla_{\mathbf{x}_t} \log p_\sigma(\mathbf{x}_t|\mathbf{x}_{t-1}) = -\frac{\mathbf{x}_t - \mathbf{x}_{t-1}}{\sigma^2} \quad (26.32)$$

Thus we see that score networks and DDPM models are trained to predict the same targets. (For a more detailed discussion of the similarities and differences between these two models, see [Son+21].)

1 **26.4.2 Connection with VAEs**

2 There is a close connection between DDPMs and variational autoencoders (Chapter 22). In particular,  
3 we can view a DDPM as a degenerate VAE with many latent Gaussian layers, all of which have  
4 the same size as the input, and where the encoder  $q$  has no free parameters, but just adds a small  
5 amount of Gaussian noise at each step. The decoder (generator) learns to map the noise back to data.  
6 The main advantage of DDPMs compared to other VAEs is that the model is easier to train. In  
7 particular, we will see that we can sample a layer at random, and then optimize a least squares loss  
8 for that layer. Furthermore, we do not need to worry about posterior collapse (Section 22.4). The  
9 disadvantage is that there is no dimensionality reduction, so there is no compact latent representation  
10 (although there is some work on combining VAEs and DDPMs to create a low dimensional encoding,  
11 see e.g., [Pan+22]). Furthermore, sampling can be slow, since the model often needs many time steps  
12 of the reverse diffusion process (although there is recent work to speed things up, see e.g., [SH22]).  
13

14 **26.4.3 Connection with flow models**

15 DDPMs are also similar in spirit to normalizing flow models (Chapter 24), since they transform noise  
16 into data vectors with the same size through a series of small steps. The main advantage of DDPMs  
17 compared to flows is that the architecture is unconstrained, and does not need to be invertible. The  
18 main disadvantage is that we can only compute a lower bound on the likelihood, rather than the  
19 exact likelihood.

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47



# 27 Generative adversarial networks

This chapter is written by Mihaela Rosca, Shakir Mohamed and Balaji Lakshminarayanan.

## 27.1 Introduction

In this chapter, we focus on **implicit generative models**, which are a kind of probabilistic model without an explicit likelihood function [ML16]. This includes the family of **Generative Adversarial Networks** or **GANs** [Goo16]. In this chapter, we provide an introduction to this topic, focusing on a probabilistic perspective.

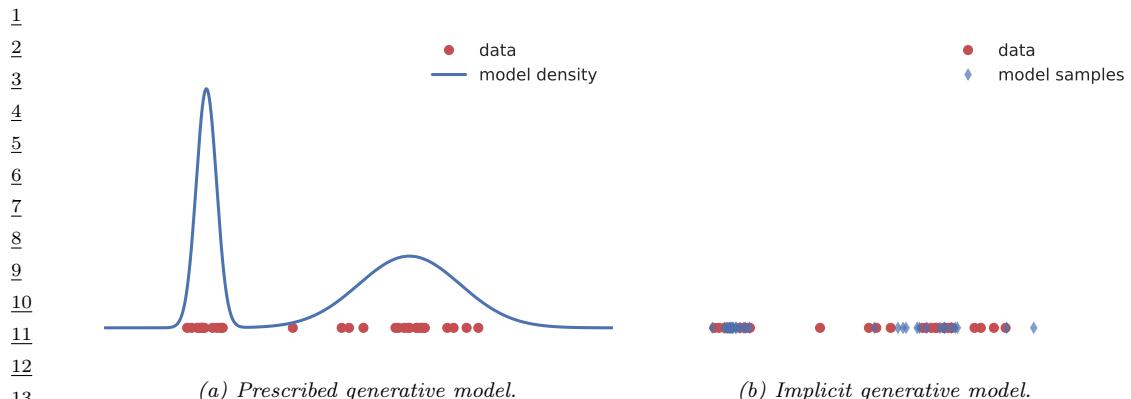
To develop a probabilistic formulation for GANs, it is useful to first distinguish between two types of probabilistic models: “**prescribed probabilistic models**” and “**implicit probabilistic models**” [DG84]. Prescribed probabilistic models, which we will call **explicit probabilistic models**, provide an explicit parametric specification of the distribution of an observed random variable  $\mathbf{x}$ , specifying a log-likelihood function  $\log q_\theta(\mathbf{x})$  with parameters  $\theta$ . Most models we encountered in this book thus far are of this form, whether they be state-of-the-art classifiers, large-vocabulary sequence models, or fine-grained spatio-temporal models. Alternatively, we can specify an **implicit probabilistic model** that defines a stochastic procedure to directly generate data. Such models are the natural approach for problems in climate and weather, population genetics, and ecology, since the mechanistic understanding of such systems can be used to directly describe the generative model [citations](#). We illustrate the difference between implicit and explicit models in Figure 27.1.

The form of implicit generative models we focus on in this chapter can be expressed as a probabilistic latent variable model, similar to VAEs (Chapter 22). Implicit generative models use a latent variable  $\mathbf{z}$  and transform it using a deterministic function  $G_\theta$  that maps from  $\mathbb{R}^m \rightarrow \mathbb{R}^d$  using parameters  $\theta$ . Implicit generative models do not include a likelihood function or observation model. Instead, the generating procedure defines a valid density on the output space that forms an effective likelihood function:

$$\mathbf{x} = G_\theta(\mathbf{z}'); \quad \mathbf{z}' \sim q(\mathbf{z}) \tag{27.1}$$

$$q_\theta(\mathbf{x}) = \frac{\partial}{\partial x_1} \cdots \frac{\partial}{\partial x_d} \int_{\{G_\theta(\mathbf{z}) \leq \mathbf{x}\}} q(\mathbf{z}) d\mathbf{z}, \tag{27.2}$$

where  $q(\mathbf{z})$  is a distribution over latent variables that provides the external source of randomness. Equation (27.2) is the definition of the transformed density  $q_\theta(\mathbf{x})$  defined as the derivative of a cumulative distribution function, and hence integrates the distribution  $q(\mathbf{z})$  over all events defined



*Figure 27.1: Visualizing the difference between prescribed and implicit generative models. Prescribed models provide direct access to the learned density (sometimes unnormalized). Implicit models only provide access to a simulator which can be used to generate samples from an implied density. Generated by [genmo\\_types\\_implicit\\_explicit.ipynb](#)*

by the set  $\{G_{\theta}(\mathbf{z}) \leq \mathbf{x}\}$ . When the latent and data dimension are equal ( $m = d$ ) and the function  $G_{\theta}(\mathbf{z})$  is invertible or has easily characterized roots, we recover the rule for transformations of probability distributions. This transformation of variables property is also used in normalizing flows (Chapter 24). In diffusion models (Chapter 26), we also transform noise into data and vice versa, but the transformation is not strictly invertible.

We can develop more general and flexible implicit generative models where the function  $G$  is a non-linear function with  $d > m$ , e.g., specified using by deep networks. When using deep networks, implicit generative models have also been referred to as generator networks, generative neural samplers, or as (differentiable) simulator-models. The integral (27.2) is intractable in the general case: we will be unable to determine the set  $\{G_{\theta}(\mathbf{z}) \leq \mathbf{x}\}$ , the integral is often unknown and the derivative is high-dimensional and difficult to compute. As we covered in other chapters on deep latent variable models (Chapter 22), intractability is also a challenge for explicit latent variable models, but the lack of a likelihood term significantly reduces the approaches available for learning, and learning methods are often referred to as **likelihood-free inference** or **simulation-based inference**.

Likelihood-free inference also forms the basis of the field known as **Approximate Bayesian Computation** or **ABC**, which we briefly discuss in Section 13.6.5. ABC and GANs give us two different algorithmic frameworks for learning in implicit generative models. Both approaches rely on a learning principle based on *comparing real and simulated data*. This type of learning by comparison instantiates a core principle of likelihood-free inference, and expanding on this idea is the focus of the next section. The subsequent sections will then focus on GANs specifically, to develop a more detailed foundation and practical considerations.

## 27.2 Learning by Comparison

In most of this book, we rely on the principle of maximum likelihood for learning. By maximizing the likelihood we effectively minimize the KL divergence between the model  $q_{\theta}$  (with parameters  $\theta$ )



Figure 27.2: The aim of implicit generative modelling objectives: to measure distances between distributions only from samples, in order to distinguish between distributions which are further apart (left) compared to those which are closer (right).

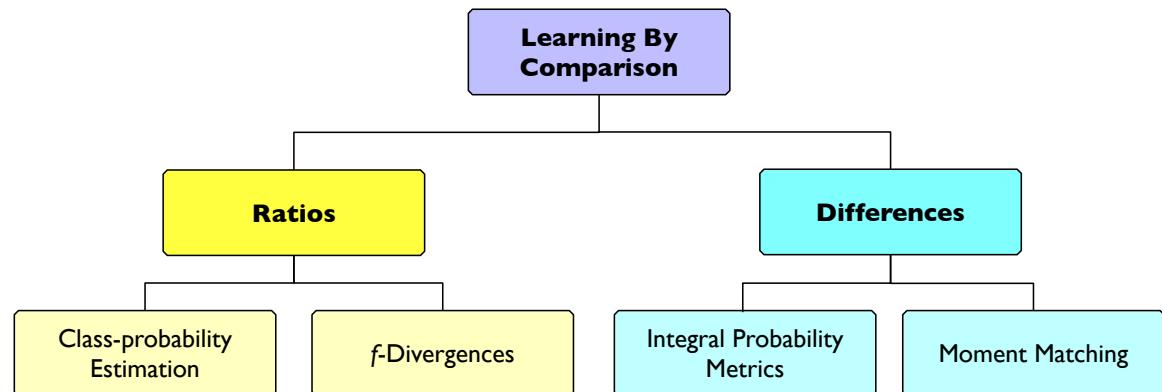


Figure 27.3: Overview of approaches for learning in implicit generative models

and the unknown true data distribution  $p^*$ . Recalling equation (27.2), in implicit models we cannot evaluate  $q_\theta(\mathbf{x})$ , and thus cannot use maximum likelihood training. As implicit models provide a sampling procedure, we instead are searching for learning principles that only use *samples from the model*.

Figure 27.2 shows that the task of learning in implicit models is to determine, from two sets of samples whether their distributions are close to each other and to quantify the distance between them. We can think of this as a ‘two sample’ or likelihood-free approach to learning by comparison. There are many ways of doing this, including using distributional divergences or distances through binary classification, the method of moments, and other approaches. Figure 27.3 shows an overview of different approaches for learning by comparison.

In this chapter, we will not explore a Bayesian approach to learning the model parameters (but see Section 13.6.5). The general approach we take is to find a way to quantify the distance between model and data distributions, use those distances to specify learning objectives, and then train the implicit model parameters  $\theta$  to minimize this measure of distributional divergence.

### 27.2.1 Guiding principles

We are looking for objectives  $\mathcal{D}(p^*, q)$  that satisfy the following desiderata:

1. Provide guarantees about learning the data distribution:  $\operatorname{argmin}_q \mathcal{D}(p^*, q) = p^*$ .

- 1 2. Can be evaluated only using samples from the data and model distribution.  
2  
3 3. Are computationally cheap to evaluate.  
4

5 Many distributional distances and divergences satisfy the first requirement, since by definition they  
6 satisfy the following:

7  $\mathcal{D}(p^*, q) \geq 0; \quad \mathcal{D}(p^*, q) = 0 \iff p^* = q$  (27.3)

8 Many distributional distances and divergences, however, fail to satisfy the other two requirements:  
9 they cannot be evaluated only using samples — such as the KL divergence, or are computationally  
10 intractable — such as the Wasserstein distance. The main approach to overcome these challenges is  
11 to *approximate the desired quantity through optimization* by introducing a comparison model—often  
12 called a **discriminator** or a **critic**  $D$ , such that:

13  $\mathcal{D}(p^*, q) = \underset{D}{\operatorname{argmax}} \mathcal{F}(D, p^*, q)$  (27.4)

14 where  $\mathcal{F}$  is a functional that depends on  $p^*$  and  $q$  only through samples. For the cases we discuss, both  
15 the model and the critic are parametric with parameters  $\theta$  and  $\phi$  respectively; instead of optimizing  
16 over distributions or functions, we optimize with respect to parameters. For the critic, this results in  
17 the optimization problem  $\operatorname{argmax}_\phi \mathcal{F}(D_\phi, p^*, q_\theta)$ . For the model parameters  $\theta$ , the exact objective  
18  $\mathcal{D}(p^*, q_\theta)$  is replaced with the tractable approximation provided through the use of  $D_\phi$ .

19 A convenient approach to ensure that  $\mathcal{F}(D_\phi, p^*, q_\theta)$  can be estimated using only samples from the  
20 model and the unknown data distribution is to depend on the two distributions only in expectation:

21  $\mathcal{F}(D_\phi, p^*, q_\theta) = \mathbb{E}_{p^*(\mathbf{x})} f(\mathbf{x}, \phi) + \mathbb{E}_{q_\theta(\mathbf{x})} g(\mathbf{x}, \phi)$  (27.5)

22 where  $f$  and  $g$  are real valued functions whose choice will define  $\mathcal{F}$ . In the case of implicit generative  
23 models, this can be rewritten to use the sampling path  $\mathbf{x} = G_\theta(\mathbf{z}), \quad \mathbf{z} \sim q(\mathbf{z})$ :

24  $\mathcal{F}(D_\phi, p^*, q_\theta) = \mathbb{E}_{p^*(\mathbf{x})} f(\mathbf{x}, \phi) + \mathbb{E}_{q(\mathbf{z})} g(G_\theta(\mathbf{z}), \phi)$  (27.6)

25 which can be estimated using Monte Carlo estimation

26  $\mathcal{F}(D_\phi, p^*, q_\theta) \approx \frac{1}{N} \sum_{i=1}^N f(\hat{\mathbf{x}}_i, \phi) + \frac{1}{M} \sum_{i=1}^M g(G_\theta(\hat{\mathbf{z}}_i), \phi); \quad \hat{\mathbf{x}}_i \sim p^*(\mathbf{x}); \quad \hat{\mathbf{z}}_i \sim q(\mathbf{z})$  (27.7)

27 Next, we will see how to instantiate these guiding principles in order to find the functions  $f$   
28 and  $g$  and thus the objective  $\mathcal{F}$  which can be used to train implicit models: class probability  
29 estimation (Section 27.2.2), bounds on  $f$ -divergences (Section 27.2.3), Integral Probability Metrics  
30 (Section 27.2.4) and moment matching (Section 27.2.5).

31

## 32 27.2.2 Class probability estimation

33 One way to compare two distributions  $p^*$  and  $q_\theta$  is to compute their density ratio  $r(\mathbf{x}) = \frac{p^*(\mathbf{x})}{q_\theta(\mathbf{x})}$ . The  
34 distributions are the same if and only if the ratio is 1 everywhere in space and density ratios like this  
35 are common in probabilistic learning. Since we cannot evaluate we cannot evaluate the densities  
36

37

Loss	Objective Function ( $D := D(\mathbf{x}; \phi) \in [0, 1]$ )
Bernoulli loss	$\mathbb{E}_{p^*(\mathbf{x})}[\log D] + \mathbb{E}_{q_\theta(\mathbf{x})}[\log(1 - D)]$
Brier score	$\mathbb{E}_{p^*(\mathbf{x})}[-(1 - D)^2] + \mathbb{E}_{q_\theta(\mathbf{x})}[-D^2]$
Exponential loss	$\mathbb{E}_{p^*(\mathbf{x})} \left[ \left(-\frac{1-D}{D}\right)^{\frac{1}{2}} \right] + \mathbb{E}_{q_\theta(\mathbf{x})} \left[ \left(-\frac{D}{1-D}\right)^{\frac{1}{2}} \right]$
Misclassification	$\mathbb{E}_{p^*(\mathbf{x})}[-\mathbb{I}[D \leq 0.5]] + \mathbb{E}_{q_\theta(\mathbf{x})}[-\mathbb{I}[D > 0.5]]$
Hinge loss	$\mathbb{E}_{p^*(\mathbf{x})} \left[ -\max \left( 0, 1 - \log \frac{D}{1-D} \right) \right] + \mathbb{E}_{q_\theta(\mathbf{x})} \left[ -\max \left( 0, 1 + \log \frac{D}{1-D} \right) \right]$
Spherical	$\mathbb{E}_{p^*(\mathbf{x})}[\alpha D] + \mathbb{E}_{q_\theta(\mathbf{x})}[\alpha(1 - D)]; \quad \alpha = (1 - 2D + 2D^2)^{-\frac{1}{2}}$

Table 27.1: Proper scoring rules that can be maximized in class probability-based learning of implicit generative models. Based on [ML16].

of implicit models, we must instead develop techniques to compute the density ratio from samples alone, following the guiding principles established above.

To compute the density ratio of the data and implicit model distributions, we assign the label  $y = 1$  to samples from the data distribution  $\mathbf{x}^* \sim p^*$ , and  $y = 0$  to the samples from the model distribution  $\mathbf{x} \sim q_\theta$ . By this construction,  $p^*(\mathbf{x}) = p(\mathbf{x}|y=1)$  and  $q_\theta(\mathbf{x}) = p(\mathbf{x}|y=0)$ . Using this insight along with Bayes rule, we can rewrite the density ratio as:

$$\frac{p^*(\mathbf{x})}{q_\theta(\mathbf{x})} = \frac{p(\mathbf{x}|y=1)}{p(\mathbf{x}|y=0)} = \frac{p(y=1|\mathbf{x})p(\mathbf{x})}{p(y=0)} \Big/ \frac{p(y=0|\mathbf{x})p(\mathbf{x})}{p(y=0)} = \frac{p(y=1|\mathbf{x})}{p(y=0|\mathbf{x})} = \frac{D(\mathbf{x})}{1 - D(\mathbf{x})}. \quad (27.8)$$

This derivation shows one can recast the problem of ratio estimation as binary classification. For simplicity, we have assumed that the same number of data and model samples are used to learn the classifier, so  $p(y=0) = p(y=1) = \frac{1}{2}$ . Since we only need to classify samples to the positive class, we denote the classifier  $p(y=1|\mathbf{x})$  as  $D(\mathbf{x})$ , the discriminator or critic.

For parametric classification, we can learn discriminators  $D_\phi(\mathbf{x}) \in [0, 1]$  with parameters  $\phi$ . Using knowledge and insight about probabilistic classification, we can learn the parameters by minimizing any proper scoring rule [GR07b] (see also Section 14.2.1). For the familiar Bernoulli log-loss (or binary cross entropy loss), we obtain the objective:

$$\begin{aligned} V(q_\theta, p^*) &= \arg \max_{\phi} \mathbb{E}_{p(\mathbf{x}|y)p(y)}[y \log D_\phi(\mathbf{x}) + (1 - y) \log(1 - D_\phi(\mathbf{x}))] \\ &= \arg \max_{\phi} \mathbb{E}_{p(\mathbf{x}|y=1)p(y=1)} \log D_\phi(\mathbf{x}) + \mathbb{E}_{p(\mathbf{x}|y=0)p(y=0)} \log(1 - D_\phi(\mathbf{x})) \end{aligned} \quad (27.9)$$

$$= \arg \max_{\phi} \frac{1}{2} \mathbb{E}_{p^*(\mathbf{x})} \log D_\phi(\mathbf{x}) + \frac{1}{2} \mathbb{E}_{q_\theta(\mathbf{x})} \log(1 - D_\phi(\mathbf{x})). \quad (27.10)$$

The same procedure can be extended beyond the Bernoulli log-loss to other proper scoring rules used for binary classification, such as those presented in Table 27.1, adapted from [ML16]. The optimal discriminator  $D$  is  $\frac{p^*(\mathbf{x})}{p^*(\mathbf{x}) + q_\theta(\mathbf{x})}$ , since:

$$\frac{p^*(\mathbf{x})}{q_\theta(\mathbf{x})} = \frac{D^*(\mathbf{x})}{1 - D^*(\mathbf{x})} \implies D^*(\mathbf{x}) = \frac{p^*(\mathbf{x})}{p^*(\mathbf{x}) + q_\theta(\mathbf{x})} \quad (27.11)$$

By substituting the optimal discriminator into the scoring rule (27.10), we can show that the objective

1  $V$  can also be interpreted as the minimization of the Jensen-Shannon divergence.  
2

$$\frac{3}{4} V^*(q_\theta, p^*) = \frac{1}{2} \mathbb{E}_{p^*(\mathbf{x})} [\log \frac{p^*(\mathbf{x})}{p^*(\mathbf{x}) + q_\theta(\mathbf{x})}] + \frac{1}{2} \mathbb{E}_{q_\theta(\mathbf{x})} [\log(1 - \frac{p^*(\mathbf{x})}{p^*(\mathbf{x}) + q_\theta(\mathbf{x})})] \quad (27.12)$$

$$\frac{5}{6} = \frac{1}{2} \mathbb{E}_{p^*(\mathbf{x})} [\log \frac{p^*(\mathbf{x})}{\frac{p^*(\mathbf{x}) + q_\theta(\mathbf{x})}{2}}] + \frac{1}{2} \mathbb{E}_{q_\theta(\mathbf{x})} [\log(\frac{q_\theta(\mathbf{x})}{\frac{p^*(\mathbf{x}) + q_\theta(\mathbf{x})}{2}})] - \log 2 \quad (27.13)$$

$$\frac{9}{10} = \frac{1}{2} D_{\text{KL}} \left( p^* \middle\| \frac{p^* + q_\theta}{2} \right) + \frac{1}{2} D_{\text{KL}} \left( q_\theta \middle\| \frac{p^* + q_\theta}{2} \right) - \log 2 \quad (27.14)$$

$$\underline{11} = JSD(p^*, q_\theta) - \log 2 \quad (27.15)$$

12 where  $JSD$  denotes the Jensen-Shannon divergence:  
13

$$\frac{14}{15} JSD(p^*, q_\theta) = \frac{1}{2} D_{\text{KL}} \left( p^* \middle\| \frac{p^* + q_\theta}{2} \right) + \frac{1}{2} D_{\text{KL}} \left( q_\theta \middle\| \frac{p^* + q_\theta}{2} \right) \quad (27.16)$$

17 This establishes a connection between *optimal* binary classification and distributional divergences.  
18 By using binary classification, we were able to compute the distributional divergence using only  
19 samples, which is the important property needed for learning implicit generative models; as expressed  
20 in the guiding principles (Section 27.2.1), we have turned an intractable estimation problem — how  
21 to estimate the JSD divergence, into an optimization problem — how to learn a classifier which can  
22 be used to approximate that divergence.

23 We would like to train the parameters  $\theta$  of generative model to minimise the divergence:

$$\frac{24}{25} \min_{\theta} JSD(p^*, q_\theta) = \min_{\theta} V^*(q_\theta, p^*) + \log 2 \quad (27.17)$$

$$\frac{26}{27} = \min_{\theta} \frac{1}{2} \mathbb{E}_{p^*(\mathbf{x})} \log D^*(\mathbf{x}) + \frac{1}{2} \mathbb{E}_{q_\theta(\mathbf{x})} \log(1 - D^*(\mathbf{x})) + \log 2 \quad (27.18)$$

29 Since we do not have access to the optimal classifier  $D^*$  but only to the neural approximation  $D_\phi$   
30 obtained using the optimization in (27.10) , this results in a min-max optimization problem:

$$\frac{31}{32} \min_{\theta} \max_{\phi} \frac{1}{2} \mathbb{E}_{p^*(\mathbf{x})} [\log D_\phi(\mathbf{x})] + \frac{1}{2} \mathbb{E}_{q_\theta(\mathbf{x})} [\log(1 - D_\phi(\mathbf{x}))] \quad (27.19)$$

34 By replacing the generating procedure (27.1) in (27.19) we obtain the objective in terms of the  
35 latent variables  $\mathbf{z}$  of the implicit generative model:  
36

$$\frac{37}{38} \min_{\theta} \max_{\phi} \frac{1}{2} \mathbb{E}_{p^*(\mathbf{x})} [\log D_\phi(\mathbf{x})] + \frac{1}{2} \mathbb{E}_{q(\mathbf{z})} [\log(1 - D_\phi(G_\theta(\mathbf{z})))], \quad (27.20)$$

40 which recovers the definition proposed in the original Generative Adversarial Network (GAN)[Goo+14].  
41 The core principle behind GANs is to train a discriminator, in this case a binary classifier, to  
42 approximate a distance or divergence between the model and data distributions, and to then train  
43 the generative model to minimize this approximation of the divergence or distance.

44 Beyond the use of the Bernoulli scoring rule used above, other scoring rules have been used to  
45 train generative models via min-max optimization. The Brier scoring rule, which under discriminator  
46 optimality conditions can be shown to correspond to minimizing the Pearson  $\chi^2$  divergence via  
47

1 similar arguments as the ones shown above has lead to LS-GAN [Mao+17]. The hinge scoring rule  
2 has become popular [Miy+18b; BDS18], and under discriminator optimality conditions corresponds  
3 to minimizing the total variational distance [NWJ+09].

4 The connection between proper scoring rules and distributional divergences allows the construction  
5 of converge guarantees for the learning criteria above, under infinite capacity of the discriminator and  
6 generator: since the minimizer of distributional divergence is the true data distribution (Equation 27.3),  
7 if the discriminator is optimal and the generator has enough capacity, it will learn the data distribution.  
8 In practice however, this assumption will not hold, as discriminators are rarely optimal; we will  
9 discuss this at length in Section 27.3.

### 11 27.2.3 Bounds on $f$ -divergences

12 As we saw with the appearance of the Jensen-Shannon divergence in the previous section, we can  
13 consider directly using a measure of distributional divergence to derive methods for learning in  
14 implicit models. One general class of divergences are the  $f$ -divergences [LV06; CS04], defined as:

$$\mathcal{D}_f [p^*(\mathbf{x}) \| q_\theta(\mathbf{x})] = \int q_\theta(\mathbf{x}) f\left(\frac{p^*(\mathbf{x})}{q_\theta(\mathbf{x})}\right) d\mathbf{x} \quad (27.21)$$

17 where  $f$  is a convex function such that  $f(1) = 0$ . For different choices of  $f$ , we can recover known  
18 distributional divergences such as the KL, reverse KL, and Jensen Shannon divergence. We discuss  
19 such connections in Section 2.9.1, and provide a summary in Table 27.2.

20 To evaluate Equation (27.21) we will need to evaluate the density of the data  $p^*(\mathbf{x})$  and the model  
21  $q_\theta(\mathbf{x})$ , neither of which are available. In the previous section we overcame the challenge of evaluating  
22 the density ratio by transforming it into a problem of binary classification. In this section, we will  
23 instead look towards the role of lower bounds on  $f$ -divergences, which is an approach for tractability  
24 that is also used for variational inference (Chapter 10).

25  $f$ -divergences have a widely-developed theory in convex analysis and information theory. Since the  
26 function  $f$  in Equation (27.21) is convex, we know that we can find a tangent that bounds it from  
27 below. The variational formulation of the  $f$ -divergence is [NWJ10b; NCT16c]:

$$\mathcal{D}_f [p^*(\mathbf{x}) \| q_\theta(\mathbf{x})] = \int q_\theta(\mathbf{x}) f\left(\frac{p^*(\mathbf{x})}{q_\theta(\mathbf{x})}\right) d\mathbf{x} \quad (27.22)$$

$$= \int q_\theta(\mathbf{x}) \sup_{t: \mathcal{X} \rightarrow \mathbb{R}} \left[ t(\mathbf{x}) \frac{p^*(\mathbf{x})}{q_\theta(\mathbf{x})} - f^\dagger(t(\mathbf{x})) \right] d\mathbf{x} \quad (27.23)$$

$$= \int \sup_{t: \mathcal{X} \rightarrow \mathbb{R}} p^*(\mathbf{x}) t(\mathbf{x}) - q_\theta(\mathbf{x}) f^\dagger(t(\mathbf{x})) d\mathbf{x} \quad (27.24)$$

$$\geq \sup_{t \in \mathcal{T}} \mathbb{E}_{p^*(\mathbf{x})}[t(\mathbf{x})] - \mathbb{E}_{q_\theta(\mathbf{x})}[f^\dagger(t(\mathbf{x}))]. \quad (27.25)$$

41 In the second line we use the result from convex analysis, discussed in the supplementary material,  
42 that re-expresses the convex function  $f$  using  $f(u) = \sup_t ut - f^\dagger(t)$ , where  $f^\dagger$  is the convex conjugate  
43 of the function  $f$ , and  $t$  is a parameter we optimize over. Since we apply  $f$  at  $u = \frac{p^*(\mathbf{x})}{q_\theta(\mathbf{x})}$  for all  $\mathbf{x} \in \mathcal{X}$ ,  
44 we make the parameter  $t$  be a function  $t(\mathbf{x})$ . The final inequality comes from replacing the supremum  
45 over all functions from the data domain  $\mathcal{X}$  to  $\mathbb{R}$  with the supremum over a family of functions  $\mathcal{T}$

Divergence	$f$	$f^\dagger$	Optimal Critic
KL	$u \log u$	$e^{u-1}$	$1 + \log r(\mathbf{x})$
Reverse KL	$-\log u$	$-1 - \log(-u)$	$-1/r(\mathbf{x})$
JSD	$u \log u - (u+1) \log \frac{u+1}{2}$	$-\log(2 - e^u)$	$\frac{2}{1+1/r(\mathbf{x})}$
Pearson $\chi^2$	$(u-1)^2$	$\frac{1}{4}u^2 + u$	$\left(\sqrt{r(\mathbf{x})} - 1\right) \sqrt{1/r(\mathbf{x})}$

Table 27.2: Standard divergences as  $f$  divergences for various choices of  $f$ . The optimal critic is written as a function of the density ratio  $r(\mathbf{x}) = \frac{p^*(\mathbf{x})}{q_\theta(\mathbf{x})}$ .

(such as the family of functions expressible by a neural network architecture), which might not be able to capture the true supremum. The function  $t$  takes the role of the discriminator or critic.

The final expression in Equation (27.25) follows the general desired form of Equation 27.5: it is the difference of two expectations, and these expectations can be computed by Monte Carlo estimation using only samples, as in Equation (27.7); despite starting with an objective (Equation 27.21) which contravened the desired principles for training implicit generative models, variational bounds have allowed us to construct an approximation which satisfies all desiderata.

Using bounds on the  $f$ -divergence, we obtain an objective (27.25) that allows learning both the generator and critic parameters. We use a critic  $D$  with parameters  $\phi$  to estimate the bound, and then optimize the parameters  $\theta$  of the generator to minimise the approximation of the  $f$ -divergence provided by the critic (we replace  $t$  above with  $D_\phi$ , to retain standard GAN notation):

$$\min_{\theta} \mathcal{D}_f(p^*, q_\theta) \geq \min_{\theta} \max_{\phi} \mathbb{E}_{p^*(\mathbf{x})}[D_\phi(\mathbf{x})] - \mathbb{E}_{q_\theta(\mathbf{x})}[f^\dagger(D_\phi(\mathbf{x}))] \quad (27.26)$$

$$= \min_{\theta} \max_{\phi} \mathbb{E}_{p^*(\mathbf{x})}[D_\phi(\mathbf{x})] - \mathbb{E}_{q(\mathbf{z})}[f^\dagger(D_\phi(G_\theta(\mathbf{z})))] \quad (27.27)$$

This approach to train an implicit generative model leads to  $f$ -GANs [NCT16c]. It is worth noting that there exists an equivalence between the scoring rules in the previous section and bounds on  $f$ -divergences [RW11]: for each scoring rule we can find an  $f$ -divergence that leads to the same training criteria and the same min-max game of Equation 27.27. An intuitive way to grasp the connection between  $f$ -divergences and proper scoring rules is through their use of density ratios: in both cases the optimal critic approximates a quantity directly related to the density ratio (see Table 27.2 for  $f$ -divergences and Equation (27.11) for scoring rules).

34

### 27.2.4 Integral probability metrics

Instead of comparing distributions by using their ratio as we did in the previous two sections, we can instead study their difference. A general class of measure of difference is given by the Integral Probability Metrics (IPMs) defined as:

$$I_{\mathcal{F}}(p^*(\mathbf{x}), q_\theta) = \sup_{f \in \mathcal{F}} |\mathbb{E}_{p^*(\mathbf{x})} f(\mathbf{x}) - \mathbb{E}_{q_\theta(\mathbf{x})} f(\mathbf{x})|. \quad (27.28)$$

The function  $f$  is a test or witness function that will take the role of the discriminator or critic. To use IPMs we must define the class of real valued, measurable functions  $\mathcal{F}$  over which the supremum is taken, and this choice will lead to different distances, just as choosing different convex functions  $f$  leads to different  $f$ -divergences. Integral probability metrics are distributional distances: beyond

47

satisfying the conditions for distributional divergences  $\mathcal{D}(p^*, q) \geq 0$ ;  $\mathcal{D}(p^*, q) = 0 \iff p^* = q$  (Equation (27.3)), they are also symmetric  $\mathcal{D}(p, q) = \mathcal{D}(q, p)$  and satisfy the triangle inequality  $\mathcal{D}(p, q) \leq \mathcal{D}(p, r) + \mathcal{D}(r, q)$ .

Not all function families satisfy these conditions of create a valid distance  $I_{\mathcal{F}}$ . To see why consider the case where  $\mathcal{F} = \{z\}$  where  $z$  is the function  $z(\mathbf{x}) = 0$ . This choice of  $\mathcal{F}$  entails that regardless of the two distributions chosen, the value in Equation 27.28 would be 0, violating the requirement that distance between two distributions be 0 only if the two distributions are the same. A popular choice of  $\mathcal{F}$  for which  $I_{\mathcal{F}}$  satisfies the conditions of a valid distributional distance is the set of 1-Lipschitz functions, which leads to the Wasserstein distance [Vil08]:

$$W_1(p^*(\mathbf{x}), q_\theta(\mathbf{x})) = \sup_{f: \|f\|_{\text{Lip}} \leq 1} \mathbb{E}_{p^*(\mathbf{x})} f(\mathbf{x}) - \mathbb{E}_{q_\theta(\mathbf{x})} f(\mathbf{x}) \quad (27.29)$$

We show an example of a Wasserstein critic in Figure 27.4a. The supremum over the set of 1-Lipschitz functions is intractable for most cases, which again suggests the introduction of a learned critic:

$$W_1(p^*(\mathbf{x}), q_\theta(\mathbf{x})) = \sup_{f: \|f\|_{\text{Lip}} \leq 1} \mathbb{E}_{p^*(\mathbf{x})} f(\mathbf{x}) - \mathbb{E}_{q_\theta(\mathbf{x})} f(\mathbf{x}) \quad (27.30)$$

$$\geq \max_{\phi: \|D_\phi\|_{\text{Lip}} \leq 1} \mathbb{E}_{p^*(\mathbf{x})} D_\phi(\mathbf{x}) - \mathbb{E}_{q_\theta(\mathbf{x})} D_\phi(\mathbf{x}), \quad (27.31)$$

where the critic  $D_\phi$  has to be regularized to be 1-Lipschitz (various techniques for Lipschitz regularization via gradient penalties or spectral normalization methods have been used [ACB17; Gul+17]). As was the case with  $f$ -divergences, we replace an intractable quantity which requires a supremum over a class of functions with a bound obtained using a subset of this function class, a subset which can be modeled using neural networks.

To train a generative model, we again introduce a min max game:

$$\min_{\theta} W_1(p^*(\mathbf{x}), q_\theta(\mathbf{x})) \geq \min_{\theta} \max_{\phi: \|D_\phi\|_{\text{Lip}} \leq 1} \mathbb{E}_{p^*(\mathbf{x})} D_\phi(\mathbf{x}) - \mathbb{E}_{q_\theta(\mathbf{x})} D_\phi(\mathbf{x}) \quad (27.32)$$

$$= \min_{\theta} \max_{\phi: \|D_\phi\|_{\text{Lip}} \leq 1} \mathbb{E}_{p^*(\mathbf{x})} D_\phi(\mathbf{x}) - \mathbb{E}_{q(\mathbf{z})} D_\phi(G_\theta(\mathbf{z})) \quad (27.33)$$

This leads to the popular WassersteinGAN [ACB17].

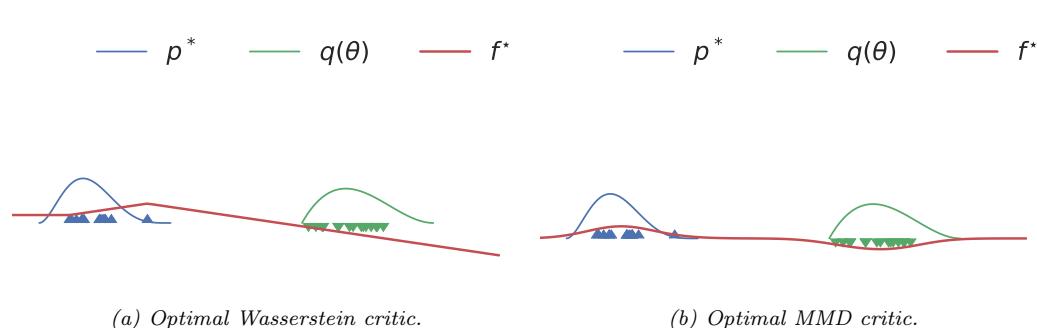
If we replace the choice of function family  $\mathcal{F}$  to that of functions in an RKHS (Section 18.3.7.1) with norm one, we obtained the maximum mean discrepancy MMD discussed in Section 2.9.3:

$$\text{MMD}(p^*(\mathbf{x}), q_\theta(\mathbf{x})) = \sup_{f: \|f\|_{\text{RKHS}} = 1} \mathbb{E}_{p^*(\mathbf{x})} f(\mathbf{x}) - \mathbb{E}_{q_\theta(\mathbf{x})} f(\mathbf{x}). \quad (27.34)$$

We show an example of an MMD critic in Figure 27.4b. It is often more convenient to use the square MMD loss [LSZ15; DRG15], which can be evaluated using the kernel  $\mathcal{K}$  (Section 18.3.7.1):

$$\text{MMD}^2(p^*, q_\theta) = \mathbb{E}_{p^*(\mathbf{x})} \mathbb{E}_{p^*(\mathbf{x}')} \mathcal{K}(\mathbf{x}, \mathbf{x}') - 2 \mathbb{E}_{p^*(\mathbf{x})} \mathbb{E}_{q_\theta(\mathbf{y})} \mathcal{K}(\mathbf{x}, \mathbf{y}) + \mathbb{E}_{q_\theta(\mathbf{y})} \mathbb{E}_{q_\theta(\mathbf{y}')} \mathcal{K}(\mathbf{y}, \mathbf{y}') \quad (27.35)$$

$$= \mathbb{E}_{p^*(\mathbf{x})} \mathbb{E}_{p^*(\mathbf{x}')} \mathcal{K}(\mathbf{x}, \mathbf{x}') - 2 \mathbb{E}_{p^*(\mathbf{x})} \mathbb{E}_{q(\mathbf{z})} \mathcal{K}(\mathbf{x}, G_\theta(\mathbf{z})) + \mathbb{E}_{q(\mathbf{z})} \mathbb{E}_{q(\mathbf{z}')} \mathcal{K}(G_\theta(\mathbf{z}), G_\theta(\mathbf{z}')) \quad (27.36)$$



**Figure 27.4:** Optimal critics in Integral Probability Metrics (IPMs). Generated by IPM divergences.ipynb

<sup>15</sup> The MMD can be directly used to learn a generative model, often called a generative matching network [LSZ15]:

$$\min_{\theta} \text{MMD}^2(p^*, q_{\theta}) \quad (27.37)$$

The choice of kernel is important. Using a fixed or predefined kernel such as a radial basis function (RBF) kernel might not be appropriate for all data modalities, such as high dimensional images. Thus we are looking for a way to learn a feature function  $\zeta$  such that we can  $\mathcal{K}(\zeta(\mathbf{x}), \zeta(\mathbf{x}'))$  is a valid kernel; luckily, we can use that for any characteristic kernel  $\mathcal{K}(\mathbf{x}, \mathbf{x}')$  and injective function  $\zeta$ ,  $\mathcal{K}(\zeta(\mathbf{x}), \zeta(\mathbf{x}'))$  is also a characteristic kernel. While this tells us that we can use feature functions in the MMD objective, it does not tell us how to learn the features. In order to ensure that the learned features are sensitive to differences between the data distribution  $p^*(\mathbf{x})$  and the model distribution  $q_\theta(\mathbf{x})$ , the kernel parameters are trained to *maximize* the square MMD. This again casts the problem into a familiar min max objective by learning the projection  $\zeta$  with parameters  $\phi$  [Li+17a]:

$$\min_{\theta} \text{MMD}_{\zeta}^2(p_{\mathcal{D}}, q_{\theta}) \quad (27.38)$$

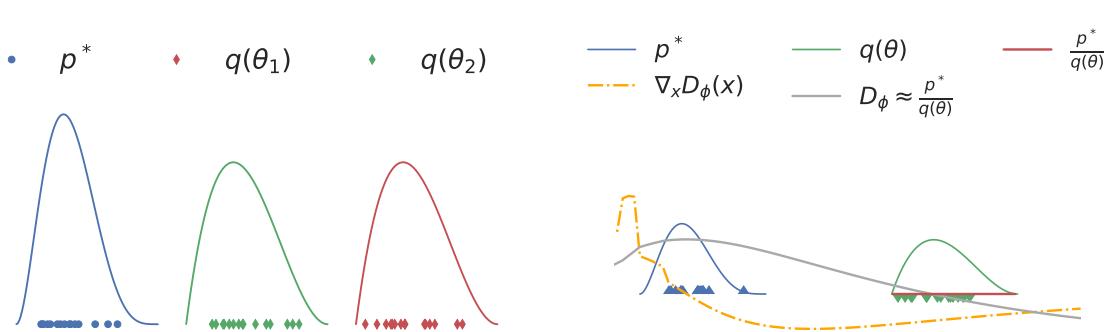
$$\begin{aligned}
&= \min_{\theta} \max_{\phi} \mathbb{E}_{p^*(\mathbf{x})} \mathbb{E}_{p^*(\mathbf{x}')} \mathcal{K}(\zeta_{\phi}(\mathbf{x}), \zeta_{\phi}(\mathbf{x}')) \\
&\quad - 2 \mathbb{E}_{p^*(\mathbf{x})} \mathbb{E}_{q_{\theta}(\mathbf{y})} \mathcal{K}(\zeta_{\phi}(\mathbf{x}), \zeta_{\phi}(\mathbf{y})) \\
&\quad + \mathbb{E}_{q_{\theta}(\mathbf{y})} \mathbb{E}_{q_{\theta}(\mathbf{y}')}) \mathcal{K}(\zeta_{\phi}(\mathbf{y}), \zeta_{\phi}(\mathbf{y}'))
\end{aligned} \tag{27.39}$$

<sup>37</sup> where  $\zeta_\phi$  regularized to be injective, though this is sometimes relaxed [Bin+18]. Unlike the Wasserstein  
<sup>38</sup> distance and  $f$ -divergences, Equation (27.39) can be estimated using Monte Carlo estimation, without  
<sup>39</sup> requiring a lower bound on the original objective.

## 41 27.2.5 Moment matching

<sup>42</sup> More broadly than distances defined by integral probability metrics, for a set of test statistics  $s$ , one can define a **moment matching** criteria [Pea36], also known as the method of moments:

$$\min_{\theta} \left\| \mathbb{E}_{p^*(\mathbf{x})} s(\mathbf{x}) - \mathbb{E}_{q_\theta(\mathbf{x})} s(\mathbf{x}) \right\|_2^2 \quad (27.40)$$



(a) Failure of the KL divergence to distinguish between distributions with non-overlapping support:  $D_{\text{KL}}(p^* \| q_{\theta_1}) = D_{\text{KL}}(p^* \| q_{\theta_2}) = \infty$ , despite  $q_{\theta_2}$  being closer to  $p^*$  than  $q_{\theta_1}$ .

(b) The density ratio  $\frac{p^*}{q_{\theta}}$  used by the KL divergence and a smooth estimate given by an MLP, together with the gradient it provides with respect to the input variable.

Figure 27.5: The KL divergence cannot provide learning signal for distributions without overlapping support (left), while the smooth approximation given by a learned decision surface like an MLP can (right). Generated by [IPM\\_divergences.ipynb](#)

where  $m(\boldsymbol{\theta}) = \mathbb{E}_{q_{\theta}(\mathbf{x})} s(\mathbf{x})$  is the *moment function*. The choice of statistic  $s(\mathbf{x})$  is crucial, since as with distributional divergences and distances, we would like to ensure that if the objective is minimized and reaches the minimal value 0, the two distributions are the same  $p^*(\mathbf{x}) = q_{\theta}(\mathbf{x})$ . To see that not all functions  $s$  satisfy this requirement consider the function  $s(\mathbf{x}) = \mathbf{x}$ : simply matching the means of two distributions is not sufficient to match higher moments (such as variance). For likelihood based models the score function  $s(\mathbf{x}) = \log q_{\theta}(\mathbf{x})$  satisfies the above requirement and leads to a consistent estimator [Vaa00], but this choice of  $s$  is not available for implicit generative models.

This motivates the search for other approaches of integrating the method of moments for implicit models. The MMD can be seen as a moment matching criteria, by matching the means of the two distributions after lifting the data into the feature space of a Reproducing Kernel Hilbert Space. But moment matching can go beyond integral probability metrics: Ravuri et al. [Rav+18] show that one can learn useful moments by using  $s$  as the set of features containing the gradients of a trained discriminator classifier  $D_{\phi}$  together with the feature of the learned critic:  $s_{\phi}(\mathbf{x}) = [\nabla_{\phi} D_{\phi}(\mathbf{x}), h_1(\mathbf{x}), \dots, h_n(\mathbf{x})]$  where  $h_1(\mathbf{x}), \dots, h_n(\mathbf{x})$  are the hidden activations of the learned critic. Both features and gradients are needed: the gradients  $\nabla_{\phi} D_{\phi}(\mathbf{x})$  are required to ensure the estimator for the parameters  $\boldsymbol{\theta}$  is consistent, since the number of moments  $s(\mathbf{x})$  needs to be larger than the number of parameters  $\boldsymbol{\theta}$ , which will be true if the critic will have more parameters than the model; the features  $h_i(\mathbf{x})$  are added since they have been shown empirically to improve performance, thus showcasing the importance of the choice of test statistics  $s$  used to train implicit models.

### 27.2.6 On density ratios and differences

We have seen how density ratios (Sections 27.2.2 and 27.2.3) and density differences (Section 27.2.4) can be used to define training objectives for implicit generative models. We now explore some of the

1 distinctions between using ratios and differences for learning by comparison, as well as explore the  
2 effects of using approximations to these objectives using function classes such as neural networks has  
3 on these distinctions.  
4

5 One often stated downside of using divergences that rely on density ratios (such as  $f$ -divergences)  
6 is their poor behavior when the distributions  $p^*$  and  $q_\theta$  do not have overlapping support. For  
7 non-overlapping support, the density ratio  $\frac{p^*}{q_\theta}$  will be  $\infty$  in the parts of the space where  $p^*(\mathbf{x}) > 0$  but  
8  $q_\theta(\mathbf{x}) = 0$ , and 0 otherwise. In that case, the  $D_{\text{KL}}(p^* \| q_\theta) = \infty$  and the  $JSD(p^*, q_\theta) = \log 2$ , regardless  
9 of the value of  $\theta$ . Thus  *$f$ -divergences cannot distinguish between different model distributions when*  
10 *they do not have overlapping support with the data distribution*, as visualized in Figure 27.5a. This is  
11 in contrast with difference based methods such as IPMs such as the Wasserstein distance and the  
12 MMD, which have smoothness requirements built in the *definition* of the method, by constraining  
13 the norm of the critic (Equations (27.29) and (27.34)). We can see the effect of these constraints  
14 in Figure 27.4: both the Wasserstein distance and the MMD provide useful signal in the case of  
15 distributions with non-overlapping support.

16 While the *definition* of  $f$ -divergences relies on density ratios (Equation (27.21)), we have seen that  
17 to train implicit generative models we use approximations to those divergences obtained using a  
18 parametric critic  $D_\phi$ . If the function family of the critic used to approximate the divergence (via the  
19 bound or class probability estimation) contains only smooth functions, it will not be able to model  
20 the sharp true density ratio which jumps from 0 to  $\infty$ , but instead provide a smooth approximation.  
21 We show an example in Figure 27.5b, where we show the density ratio for two distributions without  
22 overlapping support and an approximation provided by an MLP trained to approximate the KL  
23 divergence using Equation 27.25. Here, the smooth decision surface provided by the MLP can be  
24 used to train a generative model while the underlying KL divergence cannot be; the learned MLP  
25 provides the gradient signal on how to move distribution mass to areas with more density under  
26 the data distribution, while the KL divergence provides a zero gradient almost everywhere in the  
27 space. This ability of approximations to  $f$ -divergences to overcome non-overlapping support issues is  
28 a desirable property of generative modeling training criteria, as it allows models to learn the data  
29 distribution regardless of initialization [Fed+18]. Thus while the case of non-overlapping support  
30 provides an important theoretical difference between IPMs and  $f$ -divergences, it is less significant in  
31 practice since bounds on  $f$ -divergences or class probability estimation are used with smooth critics  
32 to approximate the underlying divergence.

33 Some density ratio and density difference based approaches also share commonalities: bounds are  
34 used both for  $f$ -divergences (variational bounds in Equation 27.25) and for the Wasserstein distance  
35 (Equation (27.31)). These bounds to distributional divergence and distances have their own set of  
36 challenges: since the generator minimizes a lower bound of the underlying divergence or distance,  
37 minimizing this objective provides no guarantees that the divergence will decrease in training. To see  
38 this, we can look at Equation 27.26: its RHS can get arbitrarily low without decreasing the LHS,  
39 the divergence we are interested in minimizing; this is unlike variational *upper* bound on the KL  
40 divergence used to train Variational Autoencoders Chapter 22.

41

42

### 43 27.3 Generative Adversarial Networks

44

45 We have looked at different learning principles that do not require the use of explicit likelihoods, and  
46 thus can be used to train implicit models. These learning principles specify training criteria, but do  
47

not tell us how to *train* models or parametrize models. To answer these questions, we now look at algorithms for training implicit models, where the models (both the discriminator and generator) are deep neural networks; this leads us to Generative Adversarial Networks (GANs). We cover how to turn learning principles into loss functions for training GANs (Section 27.3.1); how to train models using gradient descent (Section 27.3.2); how to improve GAN optimization (Section 27.3.4) and how to assess GAN convergence (Section 27.3.5).

### 27.3.1 From learning principles to loss functions

In Section 27.2 we discussed learning principles for implicit generative models: class probability estimation, bounds on  $f$ -divergences, Integral Probability Metrics and moment matching. These principles can be used to formulate loss functions to train the model parameters  $\theta$  and the critic parameters  $\phi$ . Many of these objectives follow use **zero-sum losses** via a **min-max** formulation: the generator’s goal is to minimize the same function the discriminator is maximizing. We can formalize this as:

$$\min \max V(\phi, \theta) \quad (27.41)$$

As an example, we recover the original GAN with the Bernoulli log-loss (Equation (27.19)) when

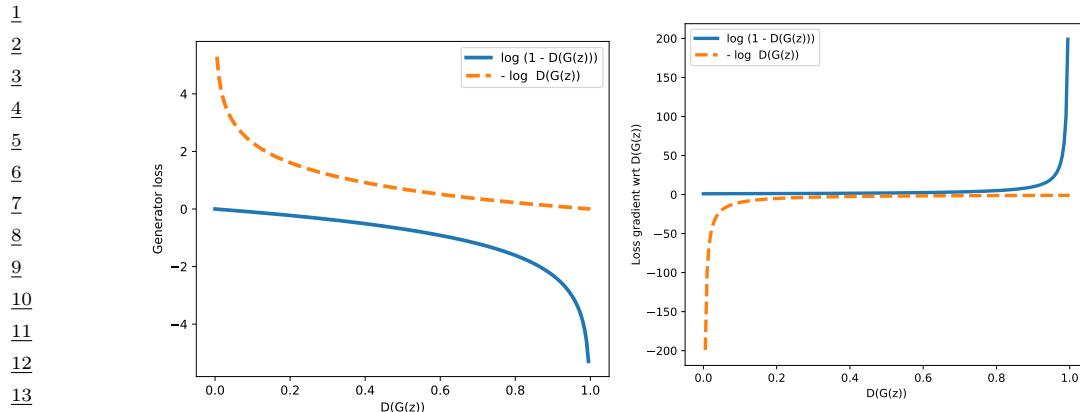
$$V(\phi, \theta) = \frac{1}{2} \mathbb{E}_{p^*(\mathbf{x})} [\log D_\phi(\mathbf{x})] + \frac{1}{2} \mathbb{E}_{q_\theta(\mathbf{x})} [\log(1 - D_\phi(\mathbf{x}))]. \quad (27.42)$$

The reason most of the learning principles we have discussed lead to zero-sum losses is due to their underlying structure: the critic maximizes a quantity in order to approximate a divergence or distance — such as an  $f$ -divergence or Integral Probability Metric — and the model minimizes this approximation to the divergence or distance. That need not be the case, however. Intuitively, the discriminator training criteria needs to ensure that the discriminator can distinguish between data and model samples, while the generator loss function needs to ensure that model samples are indistinguishable from data according to the discriminator.

To construct a GAN that is not zero-sum, consider the zero-sum criteria in the original GAN (Equation 27.42), induced by the Bernoulli scoring rule. The discriminator tries to distinguish between data and model samples by classifying the data as real (label 1) and samples as fake (label 0), while the goal of the generator is to minimize the probability that the discriminator classifies its samples as fake:  $\min_{\theta} \mathbb{E}_{q_\theta(\mathbf{x})} \log(1 - D_\phi(\mathbf{x}))$ . An equally intuitive goal for the generator is to maximize the probability that the discriminator classifies its samples as real. While the difference might seem subtle, this loss, known as the “non-saturating loss” [Goo+14], defined as  $\mathbb{E}_{q_\theta(\mathbf{x})} - \log D_\phi(\mathbf{x})$ , enjoys better gradient properties early in training, as shown in Figure 27.6: the non-saturating loss provides a stronger learning signal (via the gradient) when the generator is performing poorly, and the discriminator can easily distinguish its samples from data, i.e.  $D(G(\mathbf{z}))$  is low; more on the gradients properties the saturating and non-saturating losses can be found in [AB17; Fed+18].

There exist many other GAN losses which are not zero-sum, including formulations of LS-GAN [Mao+17], GANs trained using the hinge loss [LY17] and RelativisticGANs [JM18]. We can thus generally write a GAN formulation as follows:

$$\min_{\phi} L_D(\phi, \theta); \quad \min_{\theta} L_G(\phi, \theta). \quad (27.43)$$



*(a) Generator loss as a function of discriminator score.*

*(b) The gradients of the generator loss with respect to the discriminator score.*

*Figure 27.6: Saturating  $\log(1 - D(G(z)))$  vs non-saturating  $-\log D(G(z))$  loss functions. The non-saturating loss provides stronger gradients when the discriminator is easily detecting that generated samples are fake.*  
 Generated by [GAN\\_loss\\_types.ipynb](#)

We recover the zero-sum formulations if  $-L_D(\phi, \theta) = L_G(\phi, \theta) = V(\phi, \theta)$ . Despite departing from the zero-sum structure, the nested form of the optimization remains in the general formulation, as we will discuss in Section 27.3.2.

The loss functions for the discriminator and generator,  $L_D$  and  $L_G$  respectively, follow the general form in Equation 27.5, which allows them to be used to efficiently train implicit generative models. The majority of loss functions considered here can thus be written as follows:

$$L_D(\phi, \theta) = \mathbb{E}_{p^*(\mathbf{x})} g(D_\phi(\mathbf{x})) + \mathbb{E}_{q_\theta(\mathbf{x})} h(D_\phi(\mathbf{x})) = \mathbb{E}_{p^*(\mathbf{x})} g(D_\phi(\mathbf{x})) + \mathbb{E}_{q(\mathbf{z})} h(D_\phi(G_\theta(\mathbf{z}))) \quad (27.44)$$

$$L_G(\phi, \theta) = \mathbb{E}_{q_\theta(\mathbf{x})} l(D_\phi(\mathbf{x})) = \mathbb{E}_{q(\mathbf{z})} l(D_\phi(G_\theta(\mathbf{z}))) \quad (27.45)$$

where  $g, h, l : \mathbb{R} \rightarrow \mathbb{R}$ . We recover the original GAN for  $g(t) = -\log t$ ,  $h(t) = -\log(1-t)$  and  $l(t) = \log(1-t)$ ; the non-saturating loss for  $g(t) = -\log t$ ,  $h(t) = -\log(1-t)$  and  $l(t) = -\log(t)$ ; the Wasserstein distance formulation for  $g(t) = t$ ,  $h(t) = -t$  and  $l(t) = t$ ; for  $f$ -divergences  $g(t) = t$ ,  $h(t) = -f^\dagger(t)$  and  $l(t) = f^\dagger(t)$ .

### 27.3.2 Gradient Descent

GANs employ the learning principles discussed above in conjunction with gradient based learning for the parameters of the discriminator and generator. We assume a general formulation with a discriminator loss function  $L_D(\phi, \theta)$  and a generator loss function  $L_G(\phi, \theta)$ . Since the discriminator is often introduced to approximate a distance or divergence  $D(p^*, q_\theta)$  (Section 27.2), for the generator to minimize a good approximation of that divergence one should solve the discriminator optimization fully for each generator update. That would entail that for each generator update one would first find the optimal discriminator parameters  $\phi^* = \arg \min_{\phi} L_D(\phi, \theta)$  in order to perform a gradient update given by  $\nabla_{\theta} L_G(\phi^*, \theta)$ . Fully solving the inner optimization problem  $\phi^* = \arg \min_{\phi} L_D(\phi, \theta)$

for each optimization step of the generator is computationally prohibitive, which motivates the use of alternating updates: performing a few gradient steps to update the discriminator parameters, followed by a generator update. Note that when updating the discriminator, we keep the generator parameters fixed, and when updating the generator, we keep the discriminator parameters fixed. We show a general algorithm for these alternative updates in Algorithm 30.

---

**Algorithm 30:** General GAN training algorithm with alternating updates

---

```

1 Initialize  $\phi, \theta$ ;
2 for each training iteration do
3   for  $K$  steps do
4     └ Update the discriminator parameters  $\phi$  using the gradient  $\nabla_\phi L_D(\phi, \theta)$ ;
5     └ Update the generator parameters  $\theta$  using the gradient  $\nabla_\theta L_G(\phi, \theta)$  ;
6 Return  $\phi, \theta$ 

```

---

We are thus interested in computing  $\nabla_\phi L_D(\phi, \theta)$  and  $\nabla_\theta L_G(\phi, \theta)$ . Given the choice of loss functions follows the general form in Equations 27.44 and 27.45 both for the discriminator and generator, we can compute the gradients that can be used for training. To compute the discriminator gradients, we write:

$$\nabla_\phi L_D(\phi, \theta) = \nabla_\phi [\mathbb{E}_{p^*(\mathbf{x})} g(D_\phi(\mathbf{x})) + \mathbb{E}_{q_\theta(\mathbf{x})} h(D_\phi(\mathbf{x}))] \quad (27.46)$$

$$= \mathbb{E}_{p^*(\mathbf{x})} \nabla_\phi g(D_\phi(\mathbf{x})) + \mathbb{E}_{q_\theta(\mathbf{x})} \nabla_\phi h(D_\phi(\mathbf{x})) \quad (27.47)$$

where  $\nabla_\phi g(D_\phi(\mathbf{x}))$  and  $\nabla_\phi h(D_\phi(\mathbf{x}))$  can be computed via backpropagation, and each expectation can be estimated using Monte Carlo estimation. For the generator, we would like to compute the gradient:

$$L_G(\phi, \theta) = \nabla_\theta \mathbb{E}_{q_\theta(\mathbf{x})} l(D_\phi(\mathbf{x})) \quad (27.48)$$

Here we cannot change the order of differentiation and integration since the distribution under the integral depends on the differentiation parameter  $\theta$ . Instead, we will use that  $q_\theta(\mathbf{x})$  is the distribution induced by an implicit generative model (also known as the “reparametrization trick”, see Section 6.6.4):

$$\nabla_\theta L_G(\phi, \theta) = \nabla_\theta \mathbb{E}_{q_\theta(\mathbf{x})} l(D_\phi(\mathbf{x})) = \nabla_\theta \mathbb{E}_{q(\mathbf{z})} l(D_\phi(G_\theta(\mathbf{z}))) = \mathbb{E}_{q(\mathbf{z})} \nabla_\theta l(D_\phi(G_\theta(\mathbf{z}))) \quad (27.49)$$

and again use Monte Carlo estimation to approximate the gradient using samples from the prior  $q(\mathbf{z})$ . Replacing the choice of loss functions and Monte Carlo estimation in Algorithm 30 leads to Algorithm 31, which is often used to train GANs. See <https://github.com/probml/pyprobml/tree/master/gan> for some sample (PyTorch) code which trains various kinds of (convolutional) GANs on the CelebA face dataset.

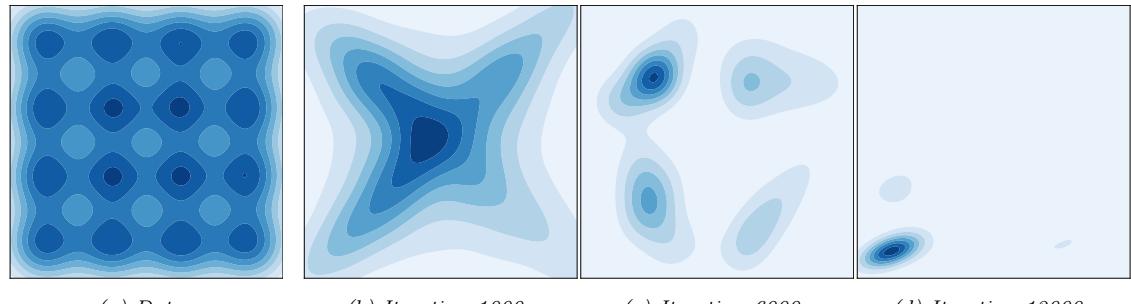
### 27.3.3 Challenges with GAN training

Due to the adversarial game nature of GANs the optimizing dynamics of GANs are both hard to study in theory, and to stabilize in practice. GANs are known to suffer from **mode collapse**, a

---

1  
2   **Algorithm 31:** GAN training algorithm  
3   1 Initialize  $\phi, \theta$ ;  
4   2 **for** each training iteration **do**  
5   3   **for**  $K$  steps **do**  
6   4     Sample minibatch of  $M$  noise vectors  $\mathbf{z}_m \sim q(\mathbf{z})$ ;  
7   5     Sample minibatch of  $M$  examples  $\mathbf{x}_m \sim p^*(\mathbf{x})$ ;  
8   6     Update the discriminator by performing stochastic gradient descent using this gradient:  
9   7        $\nabla_{\phi} \frac{1}{M} \sum_{m=1}^M [g(D_{\phi}(\mathbf{x}_m)) + \nabla_{\phi} h(D_{\phi}(G_{\theta}(\mathbf{z}_m)))]$ . ;  
10   8     Sample minibatch of  $M$  noise vectors  $\mathbf{z}_m \sim q(\mathbf{z})$ ;  
11   9     Update the generator by performing stochastic gradient descent using this gradient:  
12   10        $\nabla_{\theta} \frac{1}{M} \sum_{m=1}^M l(D_{\phi}(G_{\theta}(\mathbf{z}_m)))$ . ;  
13  
14   9   Return  $\phi, \theta$   
15

---



26   Figure 27.7: Mode hopping and collapse behavior in GAN training for a grid-structured mixture of Gaussians  
27   dataset. Left: the dataset, a mixture of 16 Gaussians in 2 dimensions. The other panels depict the progression  
28   of samples of a GAN trained on this dataset, showing how mass accumulates on one mode out of the 16.  
29   Generated by [gan\\_mog\\_mode\\_hopping.ipynb](#).

30  
31  
32  
33  
34 phenomenon where the generator converges to a distribution which does not cover not all the modes  
35 (peaks) of the data distribution, thus the model underfits the distribution. We show an example  
36 in Figure 27.7: while the data is a mixture of Gaussians with 16 modes (Figure 27.7a), the model  
37 converges only to one mode (Figure 27.7d). Alternatively, another problematic behavior is **mode**  
38 **hopping**, where the generator “hops” between generating different modes of the data distribution.  
39 An intuitive explanation for this behavior is as follows: if the generator becomes good at generating  
40 data from one mode, it will generate more from that mode. If the discriminator cannot learn to  
41 distinguish between real and generated data in this mode, the generator has no incentive to expand  
42 its support and generate data from other modes. On the other hand, if the discriminator eventually  
43 learns to distinguish between the real and generated data inside this mode, the generator can simply  
44 move (hop) to a new mode, and this game of cat and mouse can continue.

45   While mode collapse and mode hopping are often associated with GANs, many improvements have  
46 made GAN training more stable, and these behaviors more rare. These improvements include using  
47

1 large batch sizes, increasing the discriminator neural capacity, using discriminator and generator  
2 regularization, as well as more complex optimization methods.  
3

#### 4 5 27.3.4 Improving GAN optimization

6 Hyperparameter choices such as the choice of momentum can be crucial when training GANs, with  
7 lower momentum values being preferred compared to the usual high momentum used in supervised  
8 learning. Algorithms such as Adam[KB14a] provide a great boost in performance [RMC16]. Many  
9 other optimization methods have been successfully applied to GANs, such as those which target  
10 variance reduction [Cha+19c]; those which backpropagate through gradient steps, thus ensuring not  
11 that generator that does well not against the current discriminator, but to the discriminator *after it*  
12 *has been updated* [Met+16]; or using a local bilinear approximation of the two player game [SA19].  
13 While promising, these advanced optimization methods tend to have a higher computational cost,  
14 making them harder to scale to large models or large datasets compared to less efficient optimization  
15 methods.  
16

#### 17 18 27.3.5 Convergence of GAN training

19 The challenges with GAN optimization make it hard to quantify when convergence has occurred. In  
20 Section 27.2 we saw how global convergence guarantees can be provided under optimality conditions for  
21 multiple objectives constructed starting with different distributional divergences and distances: if the  
22 discriminator is optimal, the generator is minimising a distributional divergence or distance between  
23 the data and model distribution, and thus under infinite capacity and perfect optimization can learn  
24 the data distribution. This type of argument has been used since the original GAN paper [Goo+14]  
25 to connect GANs to standard objectives in generative models, and obtain the associated theoretical  
26 guarantees. From a game theory perspective, this type of convergence guarantee provides an existence  
27 proof of a global Nash equilibrium for the GAN game, though under strong assumptions. A Nash  
28 equilibrium is achieved when both players (the discriminator and generator) would incur a loss if  
29 they decide to act by changing their parameters. Consider the original GAN defined by the objective  
30 in Equation 27.19; then  $q_\theta = p^*$  and  $D_\phi(\mathbf{x}) = \frac{p^*(\mathbf{x})}{p^*(\mathbf{x}) + q_\theta(\mathbf{x})} = \frac{1}{2}$  is a global Nash equilibrium, since  
31 for a given  $q_\theta$ , the ratio  $\frac{p^*(\mathbf{x})}{p^*(\mathbf{x}) + q_\theta(\mathbf{x})}$  is the optimal discriminator (Equation 27.11), and given an  
32 optimal discriminator, the data distribution is the optimal generator as it is the minimizer of the  
33 Jensen-Shannon divergence (Equation 27.15).  
34

35 While these global theoretical guarantees provide useful insights about the GAN game, they do  
36 not account for optimization challenges that arise with accounting for the optimization trajectories  
37 of the two players, or for neural network parametrization since they assume infinite capacity both for  
38 the discriminator and generator. In practice GANs do not decrease a distance or divergence at every  
39 optimization step [Fed+18] and global guarantees are difficult to obtain when using optimization  
40 methods such as gradient descent. Instead, the focus shifts towards local convergence guarantees,  
41 such as reaching a local Nash equilibrium. A local Nash equilibrium requires that both players are at  
42 a local, not global minimum: a local Nash equilibrium is a stationary point (the gradients of the two  
43 loss functions are zero, i.e  $\nabla_\phi L_D(\phi, \theta) = \mathbf{0}$  and  $\nabla_\theta L_G(\phi, \theta) = \mathbf{0}$ ), and the eigenvalues of the Hessian  
44 of each player ( $\nabla_\phi \nabla_\phi L_D(\phi, \theta) L_D$  and  $\nabla_\theta \nabla_\theta L_G(\phi, \theta)$ ) are non-negative; for a longer discussion on  
45 Nash equilibria in continuous games see [RBS16]. For the general GAN game, it is not guaranteed  
46 that a local Nash equilibrium always exists [FO20], and weaker conditions such as stationarity or  
47

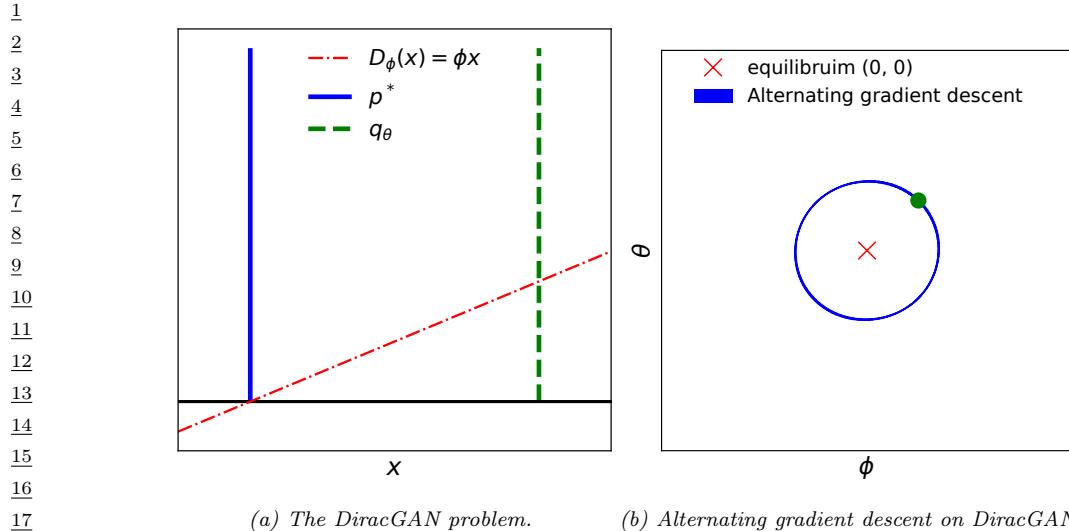


Figure 27.8: Visualizing divergence using a simple GAN: DiracGAN. Generated by [DiracGAN.ipynb](#)

locally stable stationarity have been studied [Ber+19]; other equilibrium definitions inspired by game theory have also been used [JNJ20; HLC19].

To motivate why convergence analysis is important in the case of GANs, we visualize an example of a GAN that does not converge trained with gradient descent. In DiracGAN [MGN18a] the data distribution  $p^*(\mathbf{x})$  is Dirac delta distribution with mass at zero. The generator is modeling a Dirac delta distribution with parameter  $\theta$ :  $G_\theta(z) = \theta$  and the discriminator is a linear function of the input with learned parameter  $\phi$ :  $D_\phi(x) = \phi x$ . We also assume a GAN formulation where  $g = h = -l$  in the general loss functions  $L_D$  and  $L_G$  defined above, see Equations (27.44) and (27.45). This results in the zero-sum game given by:

$$L_D = \mathbb{E}_{p^*(x)} - l(D_\phi(x)) + \mathbb{E}_{q_\theta(x)} - l(D_\phi(x)) = -l(0) - l(\theta\phi) \quad (27.50)$$

$$L_G = \mathbb{E}_{p^*(x)} l(D_\phi(x)) + \mathbb{E}_{q_\theta(x)} l(D_\phi(x)) = +l(0) + l(\theta\phi) \quad (27.51)$$

where  $l$  depends on the GAN formulation used ( $l(z) = -\log(1 + e^{-z})$  for instance). The unique equilibrium point is  $\theta = \phi = 0$ . We visualize the DiracGAN problem in Figure 27.8 and show that DiracGANs with alternating gradient descent (Algorithm 30) do not reach the equilibrium point, but instead takes a circular trajectory around the equilibrium.

There are two main theoretical approaches taken to understand GAN convergence behavior around an equilibrium: by analyzing either the discrete dynamics of gradient descent, or the underlying continuous dynamics of the game using approaches such as stability analysis. To understand the difference between the two approaches, consider the discrete dynamics defined by gradient descent with learning rates  $\alpha h$  and  $\lambda h$ , either via alternating updates (as we have seen in Algorithm 30):

$$\phi_t = \phi_{t-1} - \alpha h \nabla_\phi L_D(\phi_{t-1}, \theta_{t-1}), \quad (27.52)$$

$$\theta_t = \theta_{t-1} - \lambda h \nabla_\theta L_G(\phi_t, \theta_{t-1}) \quad (27.53)$$

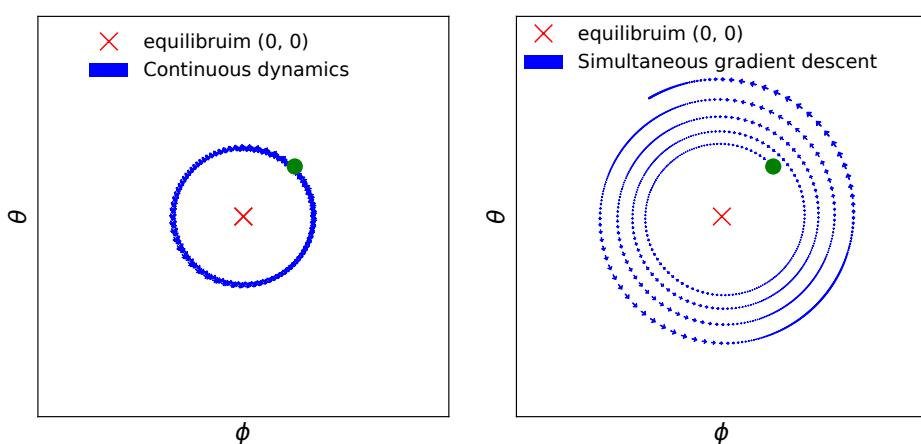


Figure 27.9: Continuous (left) and discrete dynamics (right) take different trajectories in DiracGAN.  
Generated by [DiracGAN.ipynb](#)

or simultaneous updates, where instead of alternating the gradient updates between the two players, they are both updated simultaneously:

$$\phi_t = \phi_{t-1} - \alpha h \nabla_\phi L_D(\phi_{t-1}, \theta_{t-1}), \quad (27.54)$$

$$\theta_t = \theta_{t-1} - \lambda h \nabla_\theta L_G(\phi_{t-1}, \theta_{t-1}) \quad (27.55)$$

The above dynamics of gradient descent are obtained using Euler numerical integration from the ODEs that describes the game dynamics of the two players:

$$\dot{\phi} = -\nabla_\phi L_D(\phi, \theta), \quad (27.56)$$

$$\dot{\theta} = -\nabla_\theta L_G(\phi, \theta) \quad (27.57)$$

One approach to understand the behavior of GANs is to study these underlying ODEs which when discretize results in the gradient descent updates above, rather than directly study the discrete updates. These ODEs can be used for stability analysis to study the behavior around an equilibrium entails finding the eigenvalues of the Jacobian of the game

$$J = \begin{bmatrix} -\nabla_\phi \nabla_\phi L_D(\phi, \theta) & -\nabla_\theta \nabla_\phi L_D(\phi, \theta) \\ -\nabla_\phi \nabla_\theta L_G(\phi, \theta) & -\nabla_\theta \nabla_\theta L_G(\phi, \theta) \end{bmatrix} \quad (27.58)$$

evaluated at a stationary point (i.e. where  $\nabla_\phi L_D(\phi, \theta) = 0$  and  $\nabla_\theta L_G(\phi, \theta) = 0$ ). If the eigenvalues of the Jacobian all have negative real parts, then the system is asymptotically stable around the equilibrium; if at least one eigenvalue has positive real part, the system is unstable around the equilibrium. For the DiracGAN, the Jacobian evaluated at the equilibrium  $\theta = \phi = 0$  is:

$$J = \begin{bmatrix} \nabla_\phi \nabla_\phi l(\theta\phi) + l(0) & \nabla_\theta \nabla_\phi l(\theta\phi) + l(0) \\ -\nabla_\phi \nabla_\theta l(\theta\phi) + l(0) & -\nabla_\theta \nabla_\theta l(\theta\phi) + l(0) \end{bmatrix} = \begin{bmatrix} 0 & l'(0) \\ -l'(0) & 0 \end{bmatrix} \quad (27.59)$$

where eigenvalues of this Jacobian are  $\lambda_{\pm} = \pm il'(0)$ . This is interesting, as the real parts of the eigenvalues are both 0; this result tells us that there is no asymptotic convergence to an equilibrium, but linear convergence could still occur. In this simple case we can reach the conclusion that convergence does not occur as we observe that there is a preserved quantity in this system, as  $\theta^2 + \phi^2$  does not change in time (Figure 27.9, left):

$$\frac{d(\theta^2 + \phi^2)}{dt} = 2\theta \frac{d\theta}{dt} + 2\phi \frac{d\phi}{dt} = -2\theta l'(\theta\phi)\phi + 2\phi l'(\theta\phi)\theta = 0.$$

Using stability analysis to understand the underlying continuous dynamics of GANs around an equilibrium has been used to show that explicit regularization can help convergence [NK17; Bal+18]. Alternatively, one can directly study the updates of simultaneous gradient descent shown in Equations 27.54 and 27.55. Under certain conditions [MNG17b] prove that GANs trained with simultaneous gradient descent reach a local Nash equilibrium [MNG17b]. Their approach relies on assessing the convergence of series of the form  $F^k(\mathbf{x})$  resulting from the repeated application of gradient descent update of the form  $F(\mathbf{x}) = \mathbf{x} + hG(\mathbf{x})$ , where  $h$  is the learning rate. Since the function  $F$  depends on the learning rate  $h$ , their convergence results depend on the size of the learning rate, which is not the case for continuous time approaches.

Both continuous and discrete approaches have been useful in understanding and improving GAN training; however, both approaches still leave a gap between our theoretical understanding and the most commonly used algorithms to train GANs in practice, such as alternating gradient descent or more complex optimizers used in practice, like Adam. Far from only providing different proof techniques, these approaches can reach different conclusions about the convergence of a GAN: we show an example in Figure 27.9, where we see that simultaneous gradient descent and the continuous dynamics behave differently when a large enough learning rate is used. In this case, the discretization error — the difference between the behavior of the continuous dynamics in Equations 27.56 and 27.57 and the gradient descent dynamics in Equations 27.54 and 27.55 — makes the analysis of gradient descent using continuous dynamics reach the wrong conclusion about DiracGAN [Ros+21]. This difference in behavior has been a motivator to train GANs with higher order numerical integrators such as RungeKutta4, which to more closely follow the underlying continuous system compared to gradient descent [Qin+20].

While optimization convergence analysis is an indispensable step in understanding GAN training and has led to significant practical improvements, it is worth noting that ensuring converge to an equilibrium does not ensure the model has learned a good fit of the data distribution. The loss landscape determined by the choice of  $L_D$  and  $L_G$ , as well as the parametrization of the discriminator and generator can lead to equilibria which do not capture the data distribution. The lack of distributional guarantees provided by game equilibria showcases the need to complement convergence analysis with work looking at the effect of gradient based learning in this game setting on the learned distribution.

40

## 27.4 Conditional GANs

We have thus far discussed how to use implicit generative models to learn a true unconditional distribution  $p^*(\mathbf{x})$  from which we only have samples. It is often useful, however, to be able to learn conditional distributions of the from  $p^*(\mathbf{x}|\mathbf{y})$ . This requires having paired data, where each input

47

$\mathbf{x}_n$  is paired with a corresponding set of covariates  $\mathbf{y}_n$ , such as a class label, or a set of attributes or words, so  $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n) : n = 1 : N\}$ , as in standard supervised learning. The conditioning variable can be discrete - like a class label - or continuous - such as an embedding encoding information about past experience. **Conditional generative models** are appealing since we can specify that we want the generated sample to be associated with conditioning information  $y$ , making them very amenable to real world applications - see Section 27.7.

To be able to learn implicit conditional distributions  $q_\theta(\mathbf{x}|\mathbf{y})$ , we require datasets that specify the conditioning information associated with data as well as adapt model architectures and loss functions to learn conditional distributions. In the GAN case, changing the loss function for the generative model can be done by changing the critic, since the critic is part of the loss function of the generator; it is important for the critic to provide learning signal accounting for conditioning information, by penalizing a generator which provides realistic samples but which ignore the provided conditioning.

If we do not change the form of the min-max game, but provide the conditioning information to the two players, a **conditional GAN** can be created from the original GAN game [MO14]:

$$\min_{\theta} \max_{\phi} \frac{1}{2} \mathbb{E}_{p(\mathbf{y})} \mathbb{E}_{p^*(\mathbf{x}|\mathbf{y})} [\log D_\phi(\mathbf{x}, \mathbf{y})] + \frac{1}{2} \mathbb{E}_{p(\mathbf{y})} \mathbb{E}_{q_\theta(\mathbf{x}|\mathbf{y})} [\log(1 - D_\phi(\mathbf{x}, \mathbf{y}))] \quad (27.60)$$

In the case of implicit latent variable models, the embedding information becomes an additional input to the generator, together with the latent variable  $\mathbf{z}$ :

$$\min_{\theta} \max_{\phi} \mathcal{L}(\theta, \phi) = \frac{1}{2} \mathbb{E}_{p(\mathbf{y})} \mathbb{E}_{p^*(\mathbf{x}|\mathbf{y})} [\log D_\phi(\mathbf{x}, \mathbf{y})] + \frac{1}{2} \mathbb{E}_{p(\mathbf{y})} \mathbb{E}_{q_\theta(\mathbf{x}|\mathbf{y})} [\log(1 - D_\phi(\mathcal{G}_\theta(\mathbf{z}, \mathbf{y}), \mathbf{y}))] \quad (27.61)$$

For discrete conditioning information such as labels, one can also add a new loss function, by training a critic which does not only learn to distinguish between real and fake data, but learns to classify both data and generated samples as pertaining to one of the  $K$  classes provided in the dataset [OOS17]:

$$\mathcal{L}_c(\theta, \phi) = \frac{1}{2} \mathbb{E}_{p(\mathbf{y})} \mathbb{E}_{p^*(\mathbf{x}|\mathbf{y})} [\log D_\phi(\mathbf{y}|\mathbf{x})] + \frac{1}{2} \mathbb{E}_{p(\mathbf{y})} \mathbb{E}_{q_\theta(\mathbf{x}|\mathbf{y})} [\log(D_\phi(\mathbf{y}|\mathbf{x}))] \quad (27.62)$$

Note that while we could have two critics, one unsupervised critic and one supervised which maximizes the equation above, in practice the same critic is used, to aid shaping the features used in both decision surfaces. Unlike the adversarial nature of the unsupervised game, it is in the interest of both players to maximize the classification loss  $\mathcal{L}_c$ . Thus together with the adversarial dynamics provided by  $\mathcal{L}$ , the two players are trained as follows:

$$\max_{\phi} \mathcal{L}(\theta, \phi) + \mathcal{L}_c(\theta, \phi) \quad \min_{\theta} \mathcal{L}(\theta, \phi) - \mathcal{L}_c(\theta, \phi) \quad (27.63)$$

In the case of conditional latent variable models, the latent variable controls the sample variability *inside* the mode specified by the conditioning information. In early conditional GANs, the conditioning information was provided as additional input to the discriminator and generator, for example by concatenating the conditioning information to the latent variable  $\mathbf{z}$  in the case of the generator; it has been since observed that it is important to provide the conditioning information at various layers of the model, both for the generator and the discriminator [DV+17; DSK16] or use a projection discriminator [MK18].

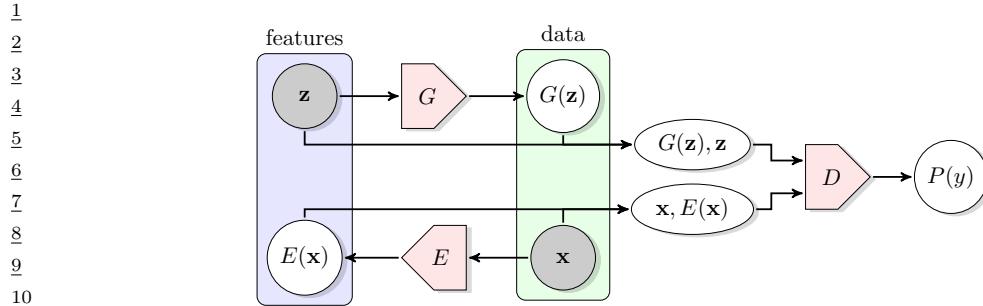


Figure 27.10: Learning an implicit posterior using an adversarial approach, as done in BiGAN. From Figure 1 of [DKD16]. Used with kind permission of Jeff Donahue.

## 27.5 Inference with GANs

Unlike other latent variable models such as Variational Autoencoders, GANs do not define an inference procedure associated with the generative model. To deploy the principles behind GANs to find a posterior distribution  $p(z|x)$ , multiple approaches have been taken, from combining GANs and Variational Autoencoders via hybrid methods [MNG17a; Sri+17; Lar+16; Mak+15b] to constructing inference methods catered to implicit variable models [Dum+16; DKD16; DS19]. An overview of these methods can be found in [Hus17b].

GAN based methods which perform inference and learn **implicit posterior distribution**  $p(z|x)$  introduce changes to the GAN algorithm to do so. An example of such a method is **BiGAN** (bidirectional GAN) [DKD16] or **ALI** (adversarially learned inference) [Dum+16], which trains an implicit parametrized encoder  $E_\theta$  to map input  $x$  to latent variables  $z$ . To ensure consistency between the encoder  $E_\theta$  and the generator  $G_\theta$ , an adversarial approach is introduced with a discriminator  $D_\phi$  learning to distinguish between pairs of data and latent samples:  $D_\phi$  learns to consider pairs  $(x, E_\theta(x))$  with  $x \sim p^*$  as real, while  $(G_\theta(z), z)$  with  $z \sim q(z)$  is considered fake. This approach, shown in Figure 27.10, ensures that the joint distributions are matched, and thus the marginal distribution  $q_\theta(x)$  given by  $G_\theta$  should learn  $p^*(x)$ , while the conditional distribution  $p_\theta(z|x)$  given by  $E_\theta$  should learn  $q_\theta(z|x) = \frac{q_\theta(x,z)}{q_\theta(x)} \propto q_\theta(x|z)q(z)$ . This joint GAN loss can be used both to train the generator  $G_\theta$  and the encoder  $E_\theta$ , without requiring a reconstruction loss common in other inference methods. While not using a reconstruction loss, this objective retains the property that under global optimality conditions the encoder and decoder are inverses of each other:  $E_\theta(G_\theta(z)) = z$  and  $G_\theta(E_\theta(x)) = x$ . (See also Section 22.2.7 for a discussion of how VAEs learn to ensure  $p^*(x)p_\theta(z|x)$  matches  $p(z)p_\theta(x|z)$  using an explicit model of the data.)

## 27.6 Neural architectures in GANs

We have so far discussed the learning principles, algorithms, and optimization methods that can be used to train implicit generative models parametrized by deep neural networks. We have not discussed, however, the importance of the choice of neural network architectures for the model and the critic, choices which have fueled the progress in GAN generation since their conception. We will look at a few case studies which show the importance of information about data modalities into

the critic and the generator (Section 27.6.1), employing the right inductive biases (Section 27.6.2), incorporating attention in GAN models (Section 27.6.3), progressive generation (Section 27.6.4), regularization (Section 27.6.5) and using large scale architectures (Section 27.6.6).

### 27.6.1 The importance of discriminator architectures

Since the discriminator or critic is rarely optimal – either due to the use of alternating gradient descent or the lack of capacity of the neural discriminator – GANs do not perform distance or divergence minimization in practice. Instead, the critic acts as part of a **learned loss function** for the model (the generator). Every time the critic is updated, the loss function for the generative model changes; this is in stark contrast with divergence minimization such maximum likelihood estimation, where the loss function stays the same throughout the training of the model. Just as learning features of data instead of handcrafting them is a reason for the success of deep learning methods, learning loss functions advanced the state of the art of generative modeling. Critics that take data modalities into account — such as convolutional critics for images and recurrent critics for sequential data such as text or audio — become part of data modality dependent loss functions. This in turn provides modality specific learning signal to the model, for example by penalizing blurry images and encouraging sharp edges, which is achieved due to the convolutional parametrization of the critic. Even within the same data modality changes to critic architectures and regularisation have been one of the main drivers in obtaining better GANs, since they affect the generator’s loss function, and thus also the *gradients of the generator* and have a strong effect on optimization.

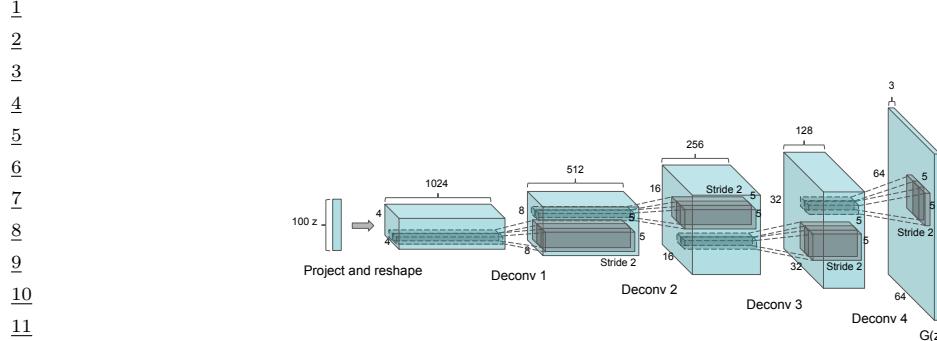
### 27.6.2 Architectural inductive biases

While the original GAN paper used convolutions only sparingly, Deep Convolutional GAN (**DCGAN**) [RMC15] performed an extensive study on what architectures are most useful for GAN training, resulting in a set of useful guidelines that led to a substantial boost in performance. Without changing the learning principles behind GANs, DCGAN was able to obtain better results on image data by using convolutional generators (Figure 27.11) and critics, using BatchNormalization for both the generator and critic, replacing pooling layers with strided convolutions, using ReLU activation networks in the generator and LeakyReLU activations in the discriminator. Many of these principles are still in use today, for larger architectures and with various loss functions. Since DCGAN, residual convolutional layers have become a key staple of both models and critics for image data [Gul+17], and recurrent architectures are used for sequence data such as text [SSG18b; Md+19].

### 27.6.3 Attention in GANs

Attention mechanisms are explained in detail in Section 16.2.7. In this section, we discuss how to use them for both the GAN generator and discriminator; this is called the Self Attention GAN or **SAGAN** model [Zha+19b]. The advantage of self attention is that it ensures that both discriminator and generator have access to a *global* view of other units of the same layer, unlike convolutional layers. This is illustrated in Figure 27.12, which visualizes the global span of attention: query points can attend to various other areas in the image.

The self-attention mechanism for convolutional features reshaped to  $\mathbf{h} \in \mathbb{R}^{C \times N}$  is defined by  $\mathbf{f} = W_f \mathbf{h}$ ,  $\mathbf{g} = W_g \mathbf{h}$ ,  $\mathbf{S} = \mathbf{f}^T \mathbf{g}$ , where  $W_f \in \mathbb{R}^{C' \times C}$ ,  $W_g \in \mathbb{R}^{C' \times C}$ , where  $C' \leq C$  is a hyperparameter.



*Figure 27.11: DCGAN convolutional generator. From Figure 1 of [RMC15]. Used with kind permission of Alec Radford.*

From  $\mathbf{S} \in \mathbb{R}^{N \times N}$ , a probability row matrix  $\beta$  is obtained by applying the softmax operator for each row, which is then used to *attend* to a linear transformation of the features  $\mathbf{o} = W_o(W_h \mathbf{h})\beta^T \in R^{C \times N}$ , using learned operators  $W_h \in \mathbb{R}^{C' \times C}, W_o \in \mathbb{R}^{C \times C'}$ . An output is then created by  $\mathbf{y} = \gamma \mathbf{o} + \mathbf{h}$ , where  $\gamma \in \mathbb{R}$  is a learned parameter.

Beyond providing global signal to the players, it is worth noting the flexibility of the self attention mechanism. The learned parameter  $\gamma$  ensures that the model can decide not to use the attention layer, and thus adding self attention does not restrict the set of possible models an architecture can learn. Moreover, self attention significantly increases the number of parameters of the model (each attention layer introduced 4 learned matrices  $\mathbf{W}_f, \mathbf{W}_g, \mathbf{W}_h, \mathbf{W}_o$ ), an approach that has been observed as a fruitful way to improve GAN training.

29

### 30 27.6.4 Progressive generation

31

One of the first successful approaches to generating higher resolution, color images from a GAN is via an *iterative* process, by first generating a lower dimensional sample, and then using that as conditioning information to generate a higher dimensional sample, and repeating the process until the desired resolution is reached. **LapGAN** [DCF+15] uses a Laplacian pyramid as the iterative building block, by first upsampling the lower dimensional samples using a simple upsampling operation, such as smoothed upsampling, and then using a conditional generator to produce a residual to be added to the upsampled version to produce the higher resolution sample. In turn, this higher resolution sample can then be provided to another LapGAN layer to produce another, even higher resolution sample, and so on - this process is shown in Figure 27.13. In LapGAN, a different generator and critic are trained for each iterative block of the model; in ProgressiveGAN [Kar+18] the lower resolution generator and critic are “grown”, by becoming part of the generator and critic used to learn to generate higher resolution samples. The higher resolution generator is obtained by adding new layers on top of the last layer of the lower resolution generator. A residual connection between an upscaled version of the lower dimensional sample and the output of the newly created higher resolution generator is added, which is annealed from 0 to 1 in training - transitioning from using the

47

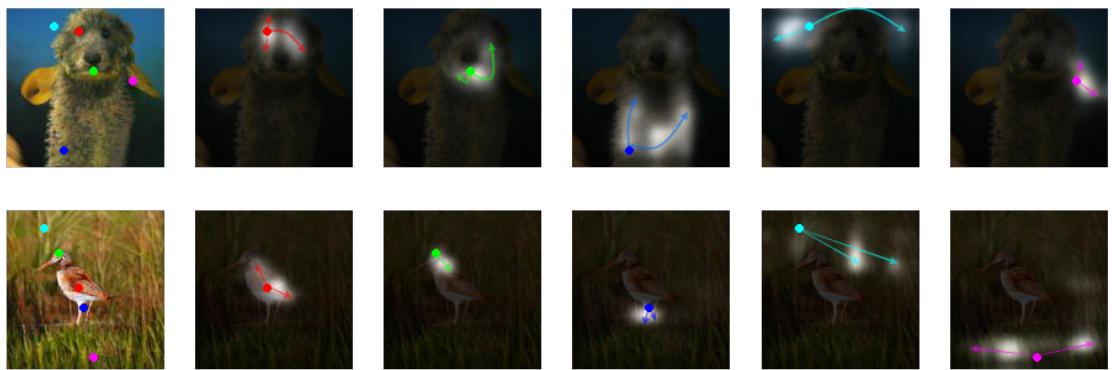


Figure 27.12: Attention queries used by a SAGAN model, showcasing the global span of attention. Each row first shows the input image and a set of color coded query locations in the image. The subsequent images show the attention maps corresponding to each query location in the first image, with the query color coded location being shown, and arrows from it to the attention map are used to highlight the most attended regions. From Figure 1 of [Zha+19b]. Used with kind permission of Han Zhang.

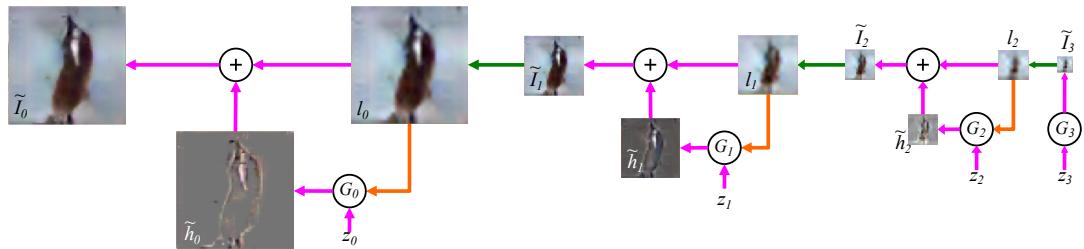


Figure 27.13: LapGAN generation algorithm: the generation process starts with a low dimension sample, which gets upsampled and residually added to the output of a generator at a higher resolution. The process gets repeated multiple times. From Figure 1 of [DCF+15]. Used with kind permission of Emily Denton.

upscaled version of the lower dimensional sample early in training, to only using the sample of the higher resolution generator at the end of training. A similar change is done to the discriminator, but the new layers are added before the layers of the higher of the lower level discriminator. Figure 27.14 shows the growing generator and discriminators in ProgressiveGAN training.

#### 27.6.5 Regularization

Regularizing both the discriminator and the generator has by now a long tradition in GAN training. Regularizing GANs can be justified from multiple perspectives: theoretically, as it has been shown to be tied to convergence analysis [MGN18b]; empirically, as it has been shown to help performance and stability in practice [RMC15; Miy+18c; Zha+19b; BDS18]; and intuitively, as it can be used to avoid overfitting in the discriminator and generator. Regularization approaches include adding noise to the

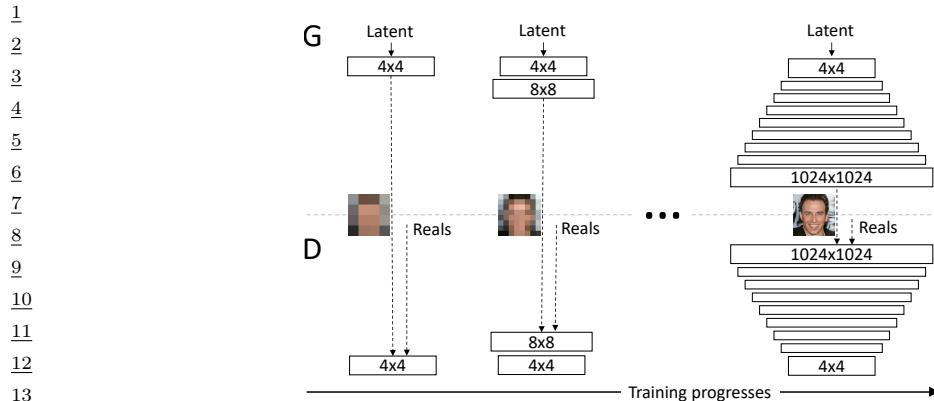


Figure 27.14: ProgressiveGAN training algorithm. The input to the discriminator at the bottom of the figure is either a generated image, or a real image (denotes as ‘Reals’ in the figure) at the corresponding resolution. From Figure 1 of [Kar+18]. Used with kind permission of Tero Karras.

discriminator input [AB17], adding noise to the discriminator and generator hidden features [ZML16], using BatchNorm for the two players [RMC15], adding dropout in the discriminator [RMC15], Spectral Normalization [Miy+18c; Zha+19b; BDS18], gradient penalties — penalizing the norm of the discriminator gradient with respect to its input  $\|\nabla_{\mathbf{x}} D_{\phi}(\mathbf{x})\|^2$  by adding a regularization term to the loss function [Arb+18; Fed+18; ACB17; Gul+17]. Often regularization methods help training regardless of the type of loss function used, and have been shown to have effects both on training performance as well as a stabilizer of the GAN game. However, improving stability and improving performance in GAN training can be at odds with each other, since too much regularization can make the models very stable, but reduce performance [BDS18].

### 27.6.6 Scaling up GAN models

By combining many of the architectural tricks discussed thus far — very large residual networks, self attention, spectral normalization both in the discriminator and the generator, BatchNormalization in the generator — one can train GANs to generating diverse, high quality data, as done with BigGAN [BDS18], StyleGAN [Kar+20c], and Alias-Free GAN [Kar+21]. Beyond combining carefully chosen architectures and regularization, creating large scale GANs also require changes in optimization, with large batch sizes being a key component. This furthers the view that the key components of the GAN game — the losses, the parametrization of the models, and optimization have to be viewed collectively rather than in isolation.

## 27.7 Applications

The ability to generate new plausible data enables a wide range of applications for GANs. This section will look at a set of applications that aim to demonstrate the breadth of GANs across different data modalities: images (Section 27.7.1), video (Section 27.7.2), audio (Section 27.7.3) and text (Section 27.7.4), and include applications such as imitation learning (Section 27.7.5), domain



Figure 27.15: Increasingly realistic synthetic faces generated by different kinds of GAN, specifically (from left to right): original GAN [Goo+14], DCGAN [RMC15], CoupledGAN [LT16], ProgressiveGAN [Kar+18], StyleGAN [KLA19]. Used with kind permission of Ian Goodfellow. An online demo, which randomly generates face images using StyleGAN, can be found at <https://thispersondoesnotexist.com>.

adaption (Section 27.7.6) and art (Section 27.7.7).

### 27.7.1 GANs for image generation

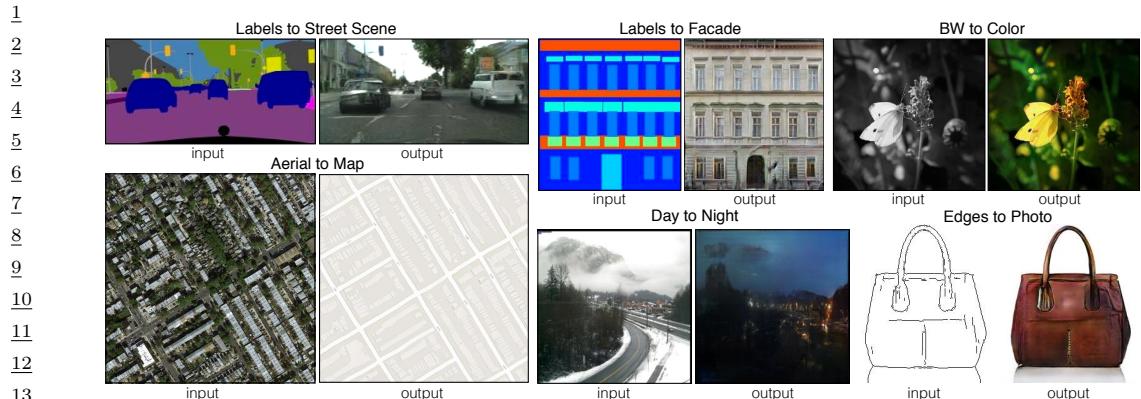
The most widely studied application area is in image generation. Image generation can take various forms, of which we cover the translation of one image to another using either paired or unpaired data sets. There are many other topics related to image GANs that we do not cover, and a more complete overview can be found in other sources, such as [Goo16] for the theory and [Bro19] for the practice. We show the progression of quality in sample generation of faces using GANs in Figure 27.15. There is also increasing need to consider the generation of images with regards to the potential risks they can have when used in other domains, which involve discussions of synthetic media and **deep fakes**, and sources for discussion include [Bru+18; Wit].

#### 27.7.1.1 Conditional image generation

Class-conditional image generation using GANs has become a very fruitful endeavor. BigGAN [BDS18] carries out class-conditional generation of ImageNet samples across a variety of categories, from dogs to cats and volcanoes and hamburgers. StyleGAN [KLA19] is able to generate high quality images of faces at high resolution by learning a conditioning style vector and the ProgressiveGAN architecture discussed in Section 27.6.4. By learning the conditioning vector they are able to generate samples which are interpolating between the styles of other samples, for example by preserving coarser style elements such as pose or face shape from one sample, and smaller scale style elements such as hair style from another; this provides fine grained control over the style of the generated images.

#### 27.7.1.2 Paired image-to-image generation

We have discussed in Section 27.4 how using paired data of the form  $(\mathbf{x}_n, \mathbf{y}_n)$  can be used to build conditional generative models of  $p(\mathbf{x}|\mathbf{y})$ . In some cases, the conditioning variable  $\mathbf{y}$  has the same size and shape as the output variable  $\mathbf{x}$ . The resulting model  $p_\theta(\mathbf{x}|\mathbf{y})$  can then be used to perform **image to image translation**, as illustrated in Figure 27.16, where  $\mathbf{y}$  is drawn from the **source**



*Figure 27.16: Example results on several image-to-image translation problems as generated by the pix2pix conditional GAN. From Figure 1 of [Iso+17]. Used with kind permission of Philip Isola.*

**domain**, and  $\mathbf{x}$  from the **target domain**. Collecting paired data of this form can be expensive, but in some cases, we can acquire it automatically. One such example is image colorization, where a paired dataset can easily be obtained by processing color images into grayscale images (see e.g., [Jas]).

A conditional GAN used for paired image-to-image translation was proposed in [Iso+17], and is known as the **pix2pix** model. It uses a U-net style architecture for the generator, as used for semantic segmentation tasks. However, they replace the batch normalization layers with instance normalization, as in neural style transfer.

For the discriminator, pix2pix uses a **patchGAN** model, that tries to classify local patches as being real or fake (as opposed to classifying the whole image). Since the patches are local, the discriminator is forced to focus on the style of the generated patches, and ensure they match the statistics of the target domain. A patch-level discriminator is also faster to train than a whole-image discriminator, and gives a denser feedback signal. This can produce results similar to Figure 27.16 (depending on the dataset).

33

### 27.7.1.3 Unpaired image-to-image generation

36 A major drawback of conditional GANs is the need to collect paired data. It is often much easier to collect **unpaired data** of the form  $\mathcal{D}_x = \{\mathbf{x}_n : n = 1 : N_x\}$  and  $\mathcal{D}_y = \{\mathbf{y}_n : n = 1 : N_y\}$ . For example,  $\mathcal{D}_x$  might be a set of daytime images, and  $\mathcal{D}_y$  a set of night-time images; it would be impossible to collect a paired dataset in which exactly the same scene is recorded during the day and night (except using a computer graphics engine, but then we wouldn't need to learn a generator).

41 We assume that the datasets  $\mathcal{D}_x$  and  $\mathcal{D}_y$  come from the marginal distributions  $p(\mathbf{x})$  and  $p(\mathbf{y})$  respectively. We would then like to fit a joint model of the form  $p(\mathbf{x}, \mathbf{y})$ , so that we can compute conditionals  $p(\mathbf{x}|\mathbf{y})$  and  $p(\mathbf{y}|\mathbf{x})$  and thus translate from one domain to another. This is called **unsupervised domain translation**.

45 In general, this is an ill-posed problem, since there are an infinite number of different joint distributions that are consistent with a set of marginals  $p(\mathbf{x})$  and  $p(\mathbf{y})$ . We can try, however, to learn

47

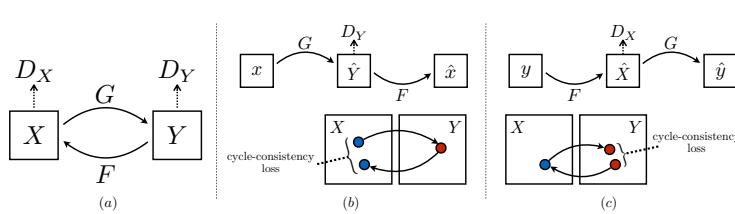


Figure 27.17: Illustration of the CycleGAN training scheme. (a) Illustration of the 4 functions that are trained. (b) Forward cycle consistency from  $\mathcal{X}$  back to  $\mathcal{X}$ . (c) Backwards cycle consistency from  $\mathcal{Y}$  back to  $\mathcal{Y}$ . From Figure 3 of [Zhu+17]. Used with kind permission of Jun-Yan Zhu.

a joint distribution such that samples from it satisfy additional constraints. For example, if  $G$  is a conditional generator that maps a sample from  $\mathcal{X}$  to  $\mathcal{Y}$ , and  $F$  maps a sample from  $\mathcal{Y}$  to  $\mathcal{X}$ , it is reasonable to require that these be inverses of each other, i.e.,  $F(G(\mathbf{x})) = \mathbf{x}$  and  $G(F(\mathbf{y})) = \mathbf{y}$ . This is called a **cycle consistency** loss [Zhu+17]. We can encourage  $G$  and  $F$  to satisfy this constraint by using a penalty term on the difference between the starting image and the image we get after going through this cycle:

$$\mathcal{L}_{\text{cycle}} = \mathbb{E}_{p(\mathbf{x})} \|F(G(\mathbf{x})) - \mathbf{x}\|_1 + \mathbb{E}_{p(\mathbf{y})} \|G(F(\mathbf{y})) - \mathbf{y}\|_1 \quad (27.64)$$

To ensure that the outputs of  $G$  are samples from  $p(\mathbf{y})$  and those of  $F$  are samples from  $p(\mathbf{x})$ , we use a standard GAN approach, introducing discriminators  $D_X$  and  $D_Y$ , which can be done using any choice of GAN loss  $\mathcal{L}_{\text{GAN}}$ ; as visualized in Figure 27.17. Finally, we can optionally check that applying the conditional generator to images from its own domain does not change them:

$$\mathcal{L}_{\text{identity}} = \mathbb{E}_{p(\mathbf{x})} \|\mathbf{x} - F(\mathbf{x})\|_1 + \mathbb{E}_{p(\mathbf{y})} \|\mathbf{y} - G(\mathbf{y})\|_1 \quad (27.65)$$

We can combine all three of these consistency losses to train the translation mappings  $F$  and  $G$ , using hyperparameters  $\lambda_1$  and  $\lambda_2$ :

$$\mathcal{L} = \mathcal{L}_{\text{GAN}} + \lambda_1 \mathcal{L}_{\text{cycle}} + \lambda_2 \mathcal{L}_{\text{identity}} \quad (27.66)$$

CycleGAN results on various datasets are shown in Figure 27.18. The bottom row shows how CycleGAN can be used for **style transfer**.

### 27.7.2 Video generation

The GAN framework can be expanded from individual images (frames) to videos; the techniques used to generate realistic images can also be applied to generate videos, with additional techniques required to ensure *spatio-temporal consistency*. Spatio-temporal consistency is obtained by ensuring that the discriminator has access to the real data and generated sequences in order, thus penalizing the generator when generating realistic individual frames without respecting temporal order [SMS17; Sai+20; CDS19; Tul+18]. Another discriminator can be employed to additionally ensure each frame is realistic [Tul+18; CDS19]. The generator itself needs to have a temporal element, which is often implemented through a recurrent component. As with images, the generation framework can be expanded to video-to-video translation [Ban+18; Wan+18], encompassing applications such as motion transfer [Cha+19a].

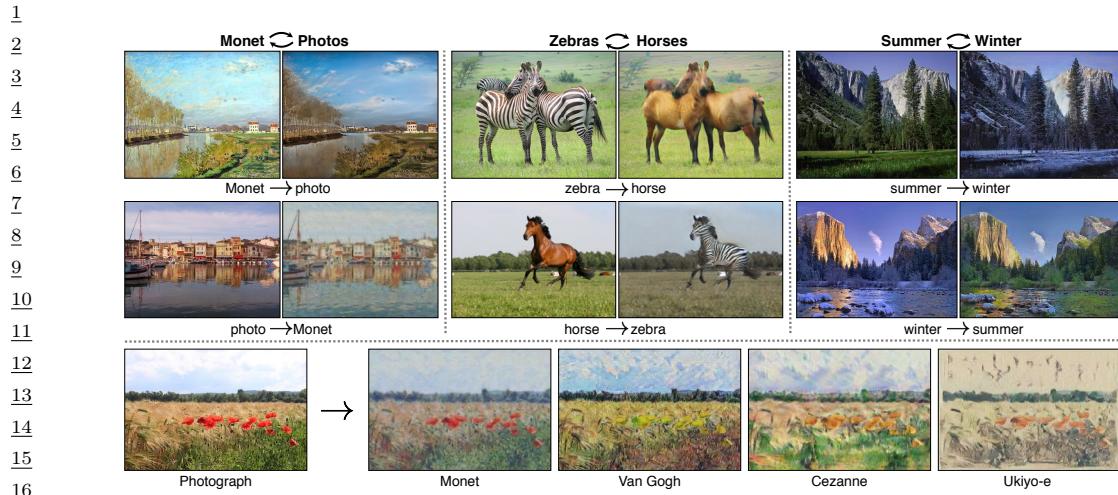


Figure 27.18: Some examples of unpaired image-to-image translation generated by the CycleGAN model. From Figure 1 of [Zhu+17]. Used with kind permission of Jun-Yan Zhu.

### 27.7.3 Audio generation

Generative models have been demonstrated in the tasks of generating audio waveforms, as well as for the task of text-to-speech (TTS) generation. Other types of generative models, such as autoregressive models, such as WaveNet [oor+16] and WaveRNN [Kal+18b] have been developed for these applications, although autoregressive models are difficult to parallelize over time since they predict each time step of the audio sequentially and can be computationally expensive and too slow to be used in practice. GANs provide an alternative approach for these tasks and other paths for addressing these concerns.

Many different GAN architectures have been developed for audio-only generation, including generation of single note recordings from instruments by GANSynth, a vocoder model that uses GANs to generate magnitude spectrograms from mel-spectrograms [Eng+18], in voice conversion using a modified CycleGAN discussed above [Kan+20], and the direct generation of raw audio in WaveGAN [DMP18].

Initial work on GANs for TTS was developed [Yan+17] whose approach is similar to conditional GANs for image generation (see Section 27.7.1.2), but uses 1d convolution instead of 2d. More recent GANs such as GAN-TTS [Biñ+19] use more advanced architectures and discriminators that operate at multiple frequency scales that have performance that now matches the best performing autoregressive models when assessed using mean opinion scores. In both the direct-audio generation, the ability of GANs to allow faster generation and different types of context is the advantage that makes them advantageous compared to other models.

### 27.7.4 Text generation

Similar to image and audio domains, there are several tasks for text data for which GAN-based approaches have been developed, including conditional text generation and text-style transfer. Text

1 data are often represented as discrete values, at either the character level or the word-level, indicating  
 2 membership within a set of a particular vocabulary size (alphabet size, or number of words). Due to  
 3 the discrete nature of text, GAN models trained on text are *explicit*, since they explicitly model the  
 4 probability distribution of the output, rather than modeling the sampling path. This is unlike most  
 5 GAN models of continuous data such as images that we have discussed in the chapter so far, though  
 6 explicit GANs of continuous data do exist [Die+19b].

7 The discrete nature of text is why maximum likelihood is one of the most common methods of  
 8 learning generative models of text. However, models trained with maximum likelihood are often  
 9 limited to autoregressive models, while like in the audio case, GANs make it possible to generate  
 10 text in a non-autoregressive manner, making other tasks possible, such as one-shot feedforward  
 11 generation [Gul+17].

12 The difficulty of generating discrete data such as text using GANs can be seen looking at their  
 13 loss function - examples in Equations (27.19), (27.21) and (27.28). GAN losses contain terms of  
 14 the form  $\mathbb{E}_{q_\theta(\mathbf{x})} f(\mathbf{x})$ , which we not only need to evaluate, but also backpropagate through, by  
 15 computing  $\nabla_{\theta} \mathbb{E}_{q_\theta(\mathbf{x})} f(\mathbf{x})$ . In the case of implicit distributions given by latent variable models, we  
 16 used the reparametrization trick to compute this gradient (Equation 27.49). In the discrete case, the  
 17 reparametrization trick is not available and we have to look for other ways to estimate the desired  
 18 gradient. One approach is to use the score function estimator, discussed in Section 6.6.3. However,  
 19 the score function estimator exhibits high gradient variance, which can destabilize training. One  
 20 common approach to avoid this issue is to pre-train the language model generator using maximum  
 21 likelihood, and then to fine-tune with a GAN loss which gets backpropagated into the generator  
 22 using the score-function estimator, as done by Sequence GAN [Yu+17], MaliGAN [Che+17], and  
 23 RankGAN [Lin+17]. While these methods spearheaded the use of GANs for text, they do not  
 24 address the inherent instabilities of score function estimation and thus have to limit the amount of  
 25 adversarial fine tuning to a small number of epochs and often use a small learning rate, keeping their  
 26 performance close to that of the maximum-likelihood solution [SSG18a; Cac+18].

27 An alternative to maximum likelihood pretraining is to use other approaches to stabilize the score  
 28 function estimator or to use continuous relaxations for backpropagation. ScratchGAN is a word-level  
 29 model that uses large batch sizes and discriminator regularization to stabilize score function training  
 30 (these techniques are the same that we have seen as stabilizers for training image GANs) [Md+19].  
 31 [Pre+17] completely avoid the score function estimator and develop a character level model without  
 32 pre-training, by using continuous relaxations and curriculum learning. These training approaches  
 33 can also benefit from other architectural advances, e.g., [NNP19] showed that language GANs can  
 34 benefit from complex architectures such as Relation Networks [San+17].

35 Finally, unsupervised text style transfer, mimicking image style transfer, have been proposed by  
 36 [She+17; Fu+17] using adversarial classifiers to decode to a different style/language, or like [Pra+18]  
 37 who trains different encoders, one per style, by combining the encoder of a pre-trained NMT and  
 38 style classifiers, among other approaches.

### 41 27.7.5 Imitation Learning

42 Imitation learning takes advantage of observations of expert demonstrations to learn action policies  
 43 and reward functions of unknown environments by minimizing some form of discrepancy between  
 44 learned and the expert behaviors. There are many approaches available, including behavioral  
 45 cloning [PPG91] that treats this problem as one of supervised learning, and inverse reinforcement  
 46

1 learning [NR00b]. GANs are appealing for imitation learning since they provide a way to avoid the  
2 difficulty of designing good discrepancy functions for behaviors, and instead learn these discrepancy  
3 functions using a discriminator between trajectories generated by a learned agent and observed  
4 demonstrations.  
5

6 This approach, known as Generative Adversarial Imitation Learning (GAIL) [HE16a] demonstrates  
7 the ability to use GANs for complex behaviors in high-dimensional environments. GAIL jointly  
8 learns a generator, which forms a stochastic policy, along with a discriminator that acts as a reward  
9 signal. Like we saw in the probabilistic development of GANs in the earlier sections, GAIL can  
10 also be generalized to multiple  $f$ -divergences, rather than the standard Jensen-Shannon divergence  
11 used as the standard loss in GANs. This has lead to a family of other GAIL variants that use  
12 other  $f$ -divergences [Ke+19a; Fin+16; Bro+20a], including  $f$ -GAIL that aims to also learn the  
13 best  $f$ -divergence to use [Zha+20e], as well as new analytical insight into the computation and  
14 generalization of such approaches [Che+20a].  
15

### 16 27.7.6 Domain Adaptation

17 An important task in machine learning is to correct for shifts in the data distribution over time,  
18 minimizing some measure of domain shift, as we discuss in Section 20.3.2. Like with the other  
19 applications, GANs are popular as ways of avoiding the choice of distance or degree of shift.  
20 Both the supervised and unsupervised approaches for image generation we reviewed earlier looked  
21 at pixel-level domain adaptation models that perform distribution alignment in raw pixel space,  
22 translating source data to the style of a target domain, as with pix2pix and CycleGAN. Extensions  
23 of these approaches for the general problem of domain adaptation seek to do this not only in the  
24 observed data space (e.g., with pixels), but also at the feature level. One general approach is  
25 domain-adversarial training of neural networks [Gan+16b] or adversarial discriminative domain  
26 adaptation (ADDA) [Tze+17]; The CyCADA approach of [Hof+18] extends CycleGAN by enforcing  
27 both structural and semantic consistency during adaptation using a cycle-consistency loss and  
28 semantics losses based on a particular visual recognition task. There are also many extensions that  
29 include class conditional information [Tsa+18; Lon+18] or adaptation when the modes to be matched  
30 have different frequencies in the source and target domains [BHC19].  
31

### 32 27.7.7 Design, Art and Creativity

33 Generative models, particularly of images, have added to approaches in the more general area of  
34 algorithmic art. The applications in image and audio generation with transfer, can also be considered  
35 aspects of artistic image generation. In these cases, the goal of training is not generalization, but to  
36 create appealing images across different types of visual aesthetics [Sar18]. One example takes style  
37 transfer GANs to create visual experiences, in which objects placed under a video are re-rendered  
38 using other visual styles in real time [AFG19]. The generation ability has been used to explore  
39 alternative designs and fabrics in fashion [Kat+19], and have now also become part of major drawing  
40 software to provide new tools to support designers [Ado]. And beyond images, creative and artistic  
41 expression using GANs include areas in music, voice, dance, and typography [AI 19].  
42

43

44

45

46

47

PART V

## Discovery



# 28 Discovery methods: an overview

## 28.1 Introduction

We have seen in Part III how to create probabilistic models that can make predictions about outputs given inputs, using supervised learning methods (conditional likelihood maximization). And we have seen in Part IV how to create probabilistic models that can generate outputs unconditionally, using unsupervised learning methods (unconditional likelihood maximization). However, in some settings, our goal is to try to *understand* a given dataset. That is, we want to *discover* something “interesting”, and possibly “actionable”. Prediction and generation are useful subroutines for discovery, but are not sufficient on their own. In particular, although neural networks often implicitly learn useful features from data, they are often hard to interpret, and the results can be unstable and sensitive to arbitrary details of the training protocol (e.g., SGD learning rates, or random seeds).

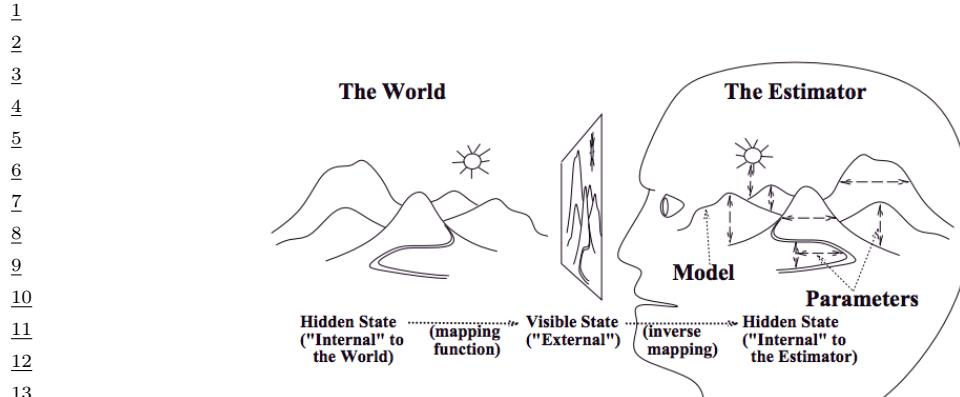
In this part of the book, we focus on learning models that create an interpretable representation of high dimensional data. A common approach is to use a **latent variable model**, in which we make the assumption that the observed data  $\mathbf{x}$  was caused by, or generated by, some underlying (often low dimensional) **latent factors**  $\mathbf{z}$ , which represents the “true” state of the world. Crucially, these latent variables are assumed to be meaningful to the end user of the model. (Thus evaluating such models will generally require domain expertise.)

For example, suppose we want to interpret an image  $\mathbf{x}$  in terms of an underlying 3d scene,  $\mathbf{z}$ , which is represented in terms of objects and surfaces. The **forwards mapping** from  $\mathbf{z}$  to  $\mathbf{x}$  is often many-to-one, i.e., different latent values, say  $\mathbf{z}$  and  $\mathbf{z}'$ , may give rise to the same observation  $\mathbf{x}$ , due to limitations of the sensor. (This is called **perceptual aliasing**.) Consequently the inverse mapping, from  $\mathbf{x}$  to  $\mathbf{z}$ , is ill-posed. In such cases, we need to impose a prior,  $p(\mathbf{z})$ , to make our estimate well-defined. In simple settings, we can use a point estimate, such as the MAP estimate

$$\hat{\mathbf{z}}(\mathbf{x}) = \operatorname{argmax}_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}) = \operatorname{argmax}_{\mathbf{z}} \log p(\mathbf{z}) + \log p(\mathbf{x}|\mathbf{z}) \quad (28.1)$$

In the context of computer vision, this approach is known as **vision as inverse graphics** or **analysis by synthesis** [KMY04; YK06; Doy+07; MC19]. See Figure 28.1 for an illustration.

This approach to inverse modeling is widely used in science and engineering, where  $\mathbf{z}$  represents the underlying state of the world which we want to estimate, and  $\mathbf{x}$  is just a noisy or partial manifestation of this true state. In some cases, we know both the prior  $p(\mathbf{z}|\boldsymbol{\theta})$  and the likelihood  $p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$ , and we just need to solve the inference problem for  $\mathbf{z}$ . But more commonly, the model parameters  $\boldsymbol{\theta}$  are also (partially) unknown, and need to be inferred from observable samples  $\mathcal{D} = \{\mathbf{x}_n : n = 1 : N\}$ . In some cases, the structure of the model itself is unknown and needs to be learned.



*Figure 28.1: Vision as inverse graphics. The agent (here represented by a human head) has to infer the scene  $\mathbf{z}$  given the image  $\mathbf{x}$  using an estimator. From Figure 1 of [Rao99]. Used with kind permission of Rajesh Rao.*

## 28.2 Overview of Part V

In Chapter 29, we discuss simple latent variable models where typically the observed data is a fixed-dimensional vector such as  $\mathbf{x} \in \mathbb{R}^D$ . In Chapter 30 and Chapter 31 we extend these models to work with sequences of correlated vectors,  $\mathbf{x} = \mathbf{x}_{1:T}$ , such as speech, video, genomics data, etc. It is straightforward to make parts of these model be nonlinear (“deep”), as we discuss. These models can also be extended to the spatio-temporal setting.

The models in Chapters 29 to 31 can all be interpreted as probabilistic graphical models with different kinds of CPDs. In Chapter 32, we discuss how to learn the structure of PGMs from data.

In Chapter 33, we discuss non-parametric Bayesian models, which allow us to represent uncertainty about many aspects of a model, such as the number of hidden states, the structure of the model, the form of a functional dependency, etc. Thus the complexity of the learned representation can grow dynamically, depending on the quantity and quality (informativeness) of the data. This is important when performing discovery tasks, and helps us maintain flexibility while still retaining interpretability.

In Chapter 34, we discuss representation learning using neural networks. This can be tackled using latent variable modeling, but there are also a variety of other estimation methods one can use. Finally, in Chapter 35, we discuss how to interpret the behavior of a learned (prediction) model (typically a neural network).

37

38

39

40

41

42

43

44

45

46

47

# 29 Latent variable models

## 29.1 Introduction

A **latent variable model (LVM)** is any probabilistic model in which some variables are always latent or hidden. A simple example is a mixture model (Section 29.2), which has the form  $p(\mathbf{x}) = \sum_k p(\mathbf{x}|z=k)p(z=k)$ , where  $z$  is an indicator variable that specifies which mixture component to use for generating  $\mathbf{x}$ . However, we can also use continuous latent variables, or a mixture of discrete and continuous. And we can also have multiple latent variables, which are interconnected in complex ways.

In this chapter, we discuss a very simple kind of LVM that has the following form:

$$\mathbf{z} \sim p(\mathbf{z}) \tag{29.1}$$

$$\mathbf{x}|\mathbf{z} \sim \text{Expfam}(\mathbf{x}|f(\mathbf{z})) \tag{29.2}$$

where  $f(\mathbf{z})$  is known as the **decoder**, and  $p(\mathbf{z})$  is some kind of prior. We assume that  $\mathbf{z}$  is a single “layer” of hidden random variables, corresponding to a set of “latent factors”. We will also mostly assume that the decoder  $f$  is a simple linear model. Thus the overall model is similar to a GLM (Section 15.1), except the input to the model is hidden.

We can create a large variety of different “classical” models by changing the form of the prior  $p(\mathbf{z})$  and/or the likelihood  $p(\mathbf{x}|\mathbf{z})$ , as we show in Table 29.1. We will give the details in the following sections. (Note that, although we are discussing generative models, our focus is on posterior inference of meaningful latents (discovery), rather than generating realistic samples of data.)

## 29.2 Mixture models

One way to create more complex probability models is to take a convex combination of simple distributions. This is called a **mixture model**. This has the form

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{k=1}^K \pi_k p_k(\mathbf{x}) \tag{29.3}$$

where  $p_k$  is the  $k$ 'th mixture component, and  $\pi_k$  are the mixture weights which satisfy  $0 \leq \pi_k \leq 1$  and  $\sum_{k=1}^K \pi_k = 1$ .

We can re-express this model as a hierarchical model, in which we introduce the discrete **latent variable**  $z \in \{1, \dots, K\}$ , which specifies which distribution to use for generating the output  $\mathbf{x}$ . The

Model	$p(\mathbf{z})$	$p(\mathbf{x} \mathbf{z})$	Section
FA/PCA	$\mathcal{N}(\mathbf{z} \mathbf{0}, \mathbf{I})$	$\mathcal{N}(\mathbf{x} \mathbf{W}\mathbf{z}, \boldsymbol{\Psi})$	Section 29.3.1
GMM	$\sum_c \text{Cat}(c \boldsymbol{\pi})\mathcal{N}(\mathbf{z} \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$	$\mathcal{N}(\mathbf{x} \mathbf{W}\mathbf{z}, \boldsymbol{\Psi})$	Section 29.2.4
MixFA	$\text{Cat}(c \boldsymbol{\pi})\mathcal{N}(\mathbf{z} \mathbf{0}, \mathbf{I})$	$\mathcal{N}(\mathbf{x} \mathbf{W}_c\mathbf{z} + \boldsymbol{\mu}_c, \boldsymbol{\Psi}_c)$	Section 29.4.3
NMF	$\prod_k \text{Ga}(z_k \alpha_k, \beta_k)$	$\prod_d \text{Poi}(x_d \exp(\mathbf{w}_d^\top \mathbf{z}))$	Section 29.5.1
Simplex FA (mPCA)	$\text{Dir}(\mathbf{z} \boldsymbol{\alpha})$	$\prod_d \text{Cat}(x_d \mathbf{W}_d\mathbf{z})$	Section 29.5.2
LDA	$\text{Dir}(\mathbf{z} \boldsymbol{\alpha})$	$\prod_d \text{Cat}(x_d \mathbf{W}\mathbf{z})$	Supplementary
ICA	$\prod_d \text{Lap}(z_d \lambda)$	$\prod_d \delta(x_d - \mathbf{w}_d^\top \mathbf{z})$	Section 29.6
Sparse coding	$\prod_k \text{Lap}(z_k \lambda)$	$\prod_d \mathcal{N}(x_d \mathbf{w}_d^\top \mathbf{z}, \sigma^2)$	Section 29.6.5

Table 29.1: Some popular “shallow” latent factor models. Abbreviations: FA = factor analysis, PCA = principal components analysis, GMM = Gaussian mixture model, NMF = non-negative matrix factorization, mPCA = multinomial PCA, LDA = latent Dirichlet allocation, ICA = independent components analysis.  $k = 1 : L$  ranges over latent dimensions,  $d = 1 : D$  ranges over observed dimensions. (For ICA, we have the constraint that  $L = D$ .)

16

17

prior on this latent variable is  $p(z = k) = \pi_k$ , and the conditional is  $p(\mathbf{x}|z = k) = p_k(\mathbf{x}) = p(\mathbf{x}|\boldsymbol{\theta}_k)$ . That is, we define the following joint model:

20

$$p(z|\boldsymbol{\theta}) = \text{Cat}(z|\boldsymbol{\pi}) \quad (29.4)$$

$$p(\mathbf{x}|z = k, \boldsymbol{\theta}) = p(\mathbf{x}|\boldsymbol{\theta}_k) \quad (29.5)$$

The “generative story” for the data is that we first sample a specific component  $z$ , and then we generate the observations  $\mathbf{x}$  using the parameters chosen according to the value of  $z$ . By marginalizing out  $z$ , we recover Equation (29.3):

27

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{k=1}^K p(z = k|\boldsymbol{\theta})p(\mathbf{x}|z = k, \boldsymbol{\theta}) = \sum_{k=1}^K \pi_k p(\mathbf{x}|\boldsymbol{\theta}_k) \quad (29.6)$$

We can create different kinds of mixture model by varying the base distribution  $p_k$ , as we illustrate below.

33

### 29.2.1 Gaussian mixture models (GMMs)

A Gaussian mixture model or GMM, also called a **mixture of Gaussians (MoG)**, is defined as follows:

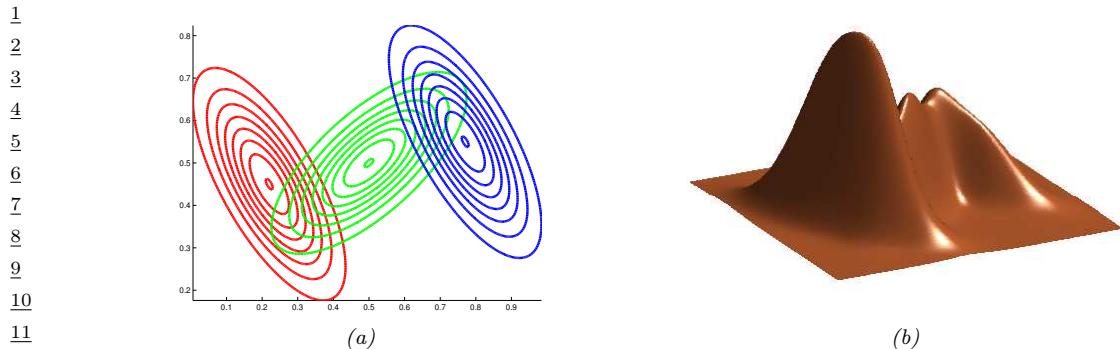
38

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (29.7)$$

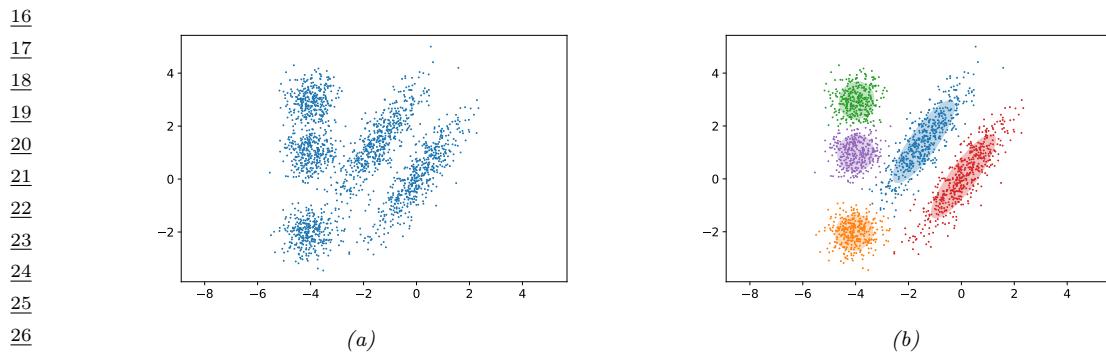
In Figure 29.1 we show the density defined by a mixture of 3 Gaussians in 2d. Each mixture component is represented by a different set of elliptical contours. If we let the number of mixture components grow sufficiently large, a GMM can approximate any smooth distribution over  $\mathbb{R}^D$ .

GMMs are often used for unsupervised **clustering** of real-valued data samples  $\mathbf{x}_n \in \mathbb{R}^D$ . This works in two stages. First we fit the model e.g., by computing the MLE  $\hat{\boldsymbol{\theta}} = \text{argmax} \log p(\mathcal{D}|\boldsymbol{\theta})$ , where

47



*Figure 29.1: A mixture of 3 Gaussians in 2d. (a) We show the contours of constant probability for each component in the mixture. (b) A surface plot of the overall density. Adapted from Figure 2.23 of [Bis06]. Generated by [gmm\\_plot\\_demo.py](#).*



*Figure 29.2: (a) Some data in 2d. (b) A possible clustering using  $K = 3$  clusters computed using a GMM. Generated by [gmm\\_2d.py](#).*

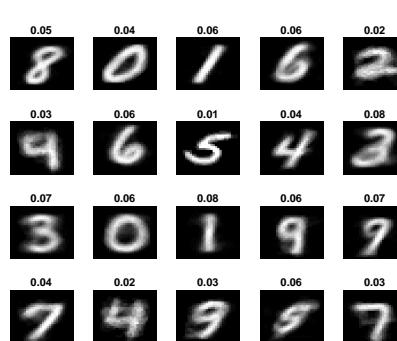
$\mathcal{D} = \{\mathbf{x}_n : n = 1 : N\}$  (e.g., using EM or SGD). Then we associate each data point  $\mathbf{x}_n$  with a discrete latent or hidden variable  $z_n \in \{1, \dots, K\}$  which specifies the identity of the mixture component or cluster which was used to generate  $\mathbf{x}_n$ . These latent identities are unknown, but we can compute a posterior over them using Bayes rule:

$$r_{nk} \triangleq p(z_n = k | \mathbf{x}_n, \boldsymbol{\theta}) = \frac{p(z_n = k | \boldsymbol{\theta}) p(\mathbf{x}_n | z_n = k, \boldsymbol{\theta})}{\sum_{k'=1}^K p(z_n = k' | \boldsymbol{\theta}) p(\mathbf{x}_n | z_n = k', \boldsymbol{\theta})} \quad (29.8)$$

The quantity  $r_{nk}$  is called the **responsibility** of cluster  $k$  for data point  $n$ . Given the responsibilities, we can compute the most probable cluster assignment as follows:

$$\hat{z}_n = \arg \max_k r_{nk} = \arg \max_k [\log p(\mathbf{x}_n | z_n = k, \boldsymbol{\theta}) + \log p(z_n = k | \boldsymbol{\theta})] \quad (29.9)$$

This is known as **hard clustering**. (If we use the responsibilities to fractionally assign each data point to different clusters, it is called **soft clustering**.) See Figure 29.2 for an example.



*Figure 29.3: We fit a mixture of 20 Bernoullis to the binarized MNIST digit data. We visualize the estimated cluster means  $\hat{\mu}_k$ . The numbers on top of each image represent the estimated mixing weights  $\hat{\pi}_k$ . No labels were used when training the model. Generated by `mix_ber_em_mnist.py`.*

If we have a uniform prior over  $z_n$ , and we use spherical Gaussians with  $\Sigma_k = \mathbf{I}$ , the hard clustering problem reduces to

$$z_n = \operatorname{argmin}_k \|\mathbf{x}_n - \hat{\mu}_k\|_2^2 \quad (29.10)$$

In other words, we assign each data point to its closest centroid, as measured by Euclidean distance. This is the basis of the K-means clustering algorithm (see the prequel to this book).

### 29.2.2 Bernoulli mixture models

If the data is binary valued, we can use a **Bernoulli mixture model** (BMM), also called a **mixture of Bernoullis**, where each mixture component has the following form:

$$p(\mathbf{x}|z=k, \boldsymbol{\theta}) = \prod_{d=1}^D \text{Ber}(y_d|\mu_{dk}) = \prod_{d=1}^D \mu_{dk}^{y_d} (1-\mu_{dk})^{1-y_d} \quad (29.11)$$

Here  $\mu_{dk}$  is the probability that bit  $d$  turns on in cluster  $k$ .

For example, consider fitting a mixture of Bernoullis using  $K = 20$  components to the MNIST dataset. The resulting parameters for each mixture component (i.e.,  $\mu_k$  and  $\pi_k$ ) are shown in Figure 29.3. We see that the model has “discovered” a representation of each type of digit. (Some digits are represented multiple times, since the model does not know the “true” number of classes. See Section 3.7.1 for information on how to choose the number  $K$  of mixture components.)

### 29.2.3 Gaussian scale mixtures

A **Gaussian scale mixture** of GSM [AM74; Wes87] is like an “infinite” mixture of Gaussians, each with a different scale (variance). More precisely, let  $x = \epsilon z$ , where  $z \sim \mathcal{N}(0, \sigma_0^2)$  and  $\epsilon \sim p(\epsilon)$ . We can

1 think of this as **multiplicative noise** being applied to the Gaussian rv  $z$ . We have  $x|\epsilon \sim \mathcal{N}(0, \epsilon^2 \sigma_0^2)$ .  
2 Marginalizing out the scale  $\epsilon$  gives  
3

4

$$\underline{5} \quad p(x) = \int \mathcal{N}(x|0, \sigma_0^2 \epsilon^2) p(\epsilon^2) d\epsilon \quad (29.12)$$

6

7 By changing the prior  $p(\epsilon)$ , we can create various interesting distributions. We give some examples  
8 below.  
9

10 The main advantage of this approach is that it is often computationally more convenient to  
11 work with the **expanded parameterization**, in which we explicitly include the scale term  $\epsilon$ , since,  
12 conditional on that, the distribution is Gaussian. We use this formulation in Section 6.7.5, where we  
13 discuss robust regression.  
14

#### 15 29.2.3.1 Student $t$ distribution as GSM

16 We can represent the Student distribution as a GSM as follows:  
17

18

$$\underline{19} \quad \mathcal{T}(y|0, \sigma^2, \nu) = \int_0^\infty \mathcal{N}(y|0, z\sigma^2) \text{IG}(z|\frac{\nu}{2}, \frac{\nu}{2}) dz = \int_0^\infty \mathcal{N}(y|0, z\sigma^2) \chi^{-2}(z|\nu, 1) dz \quad (29.13)$$

20

21 where IG is the inverse Gamma distribution (Section 2.2.2.8). Thus we can think of the Student as an  
22 infinite superposition of Gaussians of different widths; marginalizing this out induces a distribution  
23 with wider tails than a Gaussian with the same variance. This result also explains why the Student  
24 distribution approaches a Gaussian as the dof gets large, since when  $\nu = \infty$ , the inverse Gamma  
25 distribution becomes a delta function.  
26

#### 27 29.2.3.2 Laplace distribution as GSM

28 Similarly one can show that the Laplace distribution is an infinite weighted sum of Gaussians, where  
29 the precision comes from a Gamma distribution:  
30

31

$$\underline{32} \quad \text{Lap}(w|0, \lambda) = \int \mathcal{N}(w|0, \tau^2) \text{Ga}(\tau^2|1, \frac{\lambda^2}{2}) d\tau^2 \quad (29.14)$$

33

#### 34 29.2.3.3 Spike and slab distribution

35 Suppose  $\epsilon \sim \text{Ber}(\pi)$ . Note that  $p(\epsilon^2) = p(\epsilon)$ , since  $\epsilon \in \{0, 1\}$ . In this case we have  
36

37

$$\underline{38} \quad x = \sum_{\epsilon \in \{0, 1\}} \mathcal{N}(x|0, \sigma_0^2 \epsilon) p(\epsilon) = \pi \mathcal{N}(x|0, \sigma_0^2) + (1 - \pi) \delta_0(x) \quad (29.15)$$

39

40 This is known as the **spike and slab** distribution, since the  $\delta_0(x)$  is a “spike” at 0, and the  $\mathcal{N}(x|0, \sigma_0^2)$   
41 acts like a uniform “slab” for large enough  $\sigma_0$ . This distribution is useful in sparse modeling.  
42

#### 43 29.2.3.4 Horseshoe distribution

44 Suppose  $\epsilon \sim \mathcal{C}_+(1)$ , which is the half-Cauchy distribution (see Section 2.2.2.4). Then the induced  
45 distribution  $p(x)$  is called the **horseshoe distribution** [CPS10]. This has a spike at 0, like the  
46

47



Figure 29.4: Example of recovering a clean image (right) from a corrupted version (left) using MAP estimation with a GMM patch prior and Gaussian likelihood. First row: image denoising. Second row: image deblurring. Third row: image inpainting. From [RW15] and [ZW11]. Used with kind permission of Dan Rosenbaum and Daniel Zoran.

Student and Laplace distributions, but has heavy tails that do not asymptote to zero. This makes it useful as a sparsity promoting prior, that “kills off” small parameters, but does not overregularize large parameters.

#### 29.2.4 Using GMMs as a prior for inverse imaging problems

In this section, we consider using GMMs as a blackbox density model to regularize the inversion of a many-to-one mapping. Specifically, we consider the problem of inferring a “clean” image  $\mathbf{x}$  from a corrupted version  $\mathbf{y}$ . We use a linear-Gaussian forwards model of the form

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{W}\mathbf{x}, \sigma^2\mathbf{I}) \quad (29.16)$$

where  $\sigma^2$  is the variance of the measurement noise. The form of the matrix  $\mathbf{W}$  depends on the nature of the corruption, which we assume is known, for simplicity. Here are some common examples of different kinds of corruption we can model in our approach:

- If the corruption is due to additive noise (as in Figure 29.4a), we can set  $\mathbf{W} = \mathbf{I}$ . The resulting MAP estimate can be used for **image denoising**, as in Figure 29.4b.
- If the corruption is due to blurring (as in Figure 29.4c), we can set  $\mathbf{W}$  to be a fixed convolutional kernel [KF09b]. The resulting MAP estimate can be used for **image deblurring**, as in Figure 29.4d.
- If the corruption is due to occlusion (as in Figure 29.4e), we can set  $\mathbf{W}$  to be a diagonal matrix, with 0s in the locations corresponding to the occluders. The resulting MAP estimate can be used for **image inpainting**, as in Figure 29.4f.
- If the corruption is due to downsampling, we can set  $\mathbf{W}$  to a convolutional kernel. The resulting MAP estimate can be used for **image super-resolution**.

Thus we see that the linear-Gaussian likelihood model is surprisingly flexible. Given the model, our goal is to invert it, by computing the MAP estimate  $\hat{\mathbf{x}} = \text{argmax } p(\mathbf{x}|\mathbf{y})$ . However, the problem of inverting this model is ill-posed, since there are many possible latent images  $\mathbf{x}$  that map to the same observed image  $\mathbf{y}$ . Therefore we need to use a prior to regularize the inversion process.

In [ZW11], they propose to partition the image into patches, and to use a GMM prior of the form  $p(\mathbf{x}_i) = \sum_k p(c_i = k)\mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$  for each patch  $i$ . They use  $K = 200$  mixture components, and they fit the GMM on a dataset of 2M clean image patches.

To compute the MAP mixture component,  $c_i^*$ , we can marginalize out  $\mathbf{x}_i$  and use Equation (2.62) to compute the marginal likelihood

$$c_i^* = \underset{c}{\text{argmax}} \, p(c)p(\mathbf{y}_i|c) = \underset{c}{\text{argmax}} \, p(c)\mathcal{N}(\mathbf{y}_i|\mathbf{W}\boldsymbol{\mu}_c, \sigma^2\mathbf{I} + \mathbf{W}\boldsymbol{\Sigma}_c\mathbf{W}^\top) \quad (29.17)$$

We can then approximate the MAP for the latent patch  $\mathbf{x}_i$  by using the approximation

$$p(\mathbf{x}_i|\mathbf{y}_i) \approx p(\mathbf{x}_i|\mathbf{y}_i, c_i^*) \propto \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_{c_i^*}, \boldsymbol{\Sigma}_{c_i^*})\mathcal{N}(\mathbf{y}_i|\mathbf{W}\mathbf{x}_i, \sigma^2\mathbf{I}) \quad (29.18)$$

If we know  $c_i^*$ , we can compute the above using Bayes rule for Gaussians in Equation (2.59).

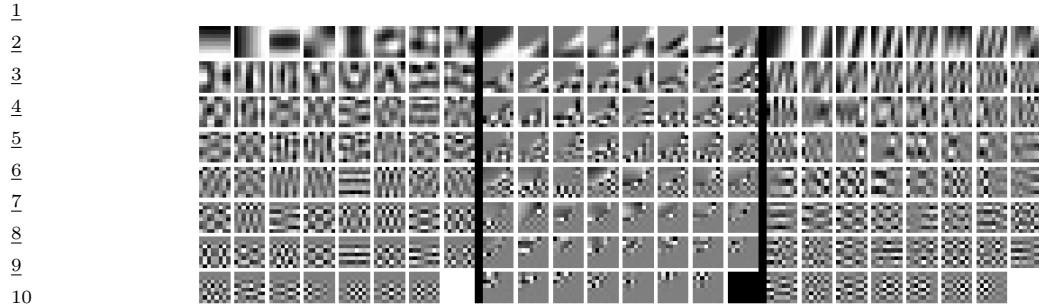
To apply this method to full images, [ZW11] optimize the following objective

$$E(\mathbf{x}|\mathbf{y}) = \frac{1}{2\sigma^2} \|\mathbf{W}\mathbf{x} - \mathbf{y}\|^2 - \text{EPPL}(\mathbf{x}) \quad (29.19)$$

where **EPPL** is the “expected patch log likelihood”, given by

$$\text{EPPL}(\mathbf{x}) = \sum_i \log p(\mathbf{P}_i \mathbf{x}) \quad (29.20)$$

where  $\mathbf{x}_i = \mathbf{P}_i \mathbf{x}$  is the  $i$ 'th patch computed by projection matrix  $\mathbf{P}_i$ . Since these patches overlap, this is not a valid likelihood, since it overcounts the pixels. Nevertheless, optimizing this objective (using a method called “half quadratic splitting”) works well empirically. See Figure 29.4 for some examples of this process in action.



11 *Figure 29.5: Illustration of the parameters learned by a GMM applied to image patches. Each of the 3 panels*  
12 *corresponds to a different mixture component  $k$ . Within each panel, we show the eigenvectors (reshaped as*  
13 *images) of the covariance matrix  $\Sigma_k$  in decreasing order of eigenvalue. We see various kinds of patterns,*  
14 *including ones that look like the ones learned from PCA (see Figure 29.25), but also ones that look like edges*  
15 *and texture. From Figure 6 of [ZW11]. Used with kind permission of Daniel Zoran.*

1617

18 A more principled solution to the overlapping patch problem is to use a multiscale model, as  
19 proposed in [PE16]. Another approach, proposed in [FW21], uses Gibbs sampling to combine samples  
20 from overlapping patches. This approach has the additional advantage of computing posterior samples  
21 from  $p(\mathbf{x}|\mathbf{y})$ , which can look much better than the posterior mean or mode computed by optimization  
22 methods. (For example, if the corruption process removes the color from the latent image  $\mathbf{x}$  to  
23 create a gray scale  $\mathbf{y}$ , then the posterior MAP estimate of  $\mathbf{x}$  will also be a gray scale image, whereas  
24 posterior samples will be color images.) See also Section 29.4.3 where we show how to extend the  
25 GMM model to a mixture of low rank Gaussians, which lets us directly model images instead of  
26 image patches.  
27

28

#### 29 29.2.4.1 Why does the method work?

30 To understand why such a simple model of image patches works so well, note that the log prior for a  
31 single latent image patch  $\mathbf{x}_i$  using mixture component  $k$  can be written as follows:  
32

$$\log p(\mathbf{x}_i|c_n = k) = \log \mathcal{N}(\mathbf{x}_i|\mathbf{0}, \Sigma_k) = -\mathbf{x}_i^\top \Sigma_k^{-1} \mathbf{x}_i + a_k \quad (29.21)$$

34

35 where  $a_k$  is a constant that depends on  $k$  but is independent of  $\mathbf{x}_i$ . Let  $\Sigma_k = \mathbf{V}\Lambda_k\mathbf{V}^\top$  be an  
36 eigendecomposition of  $\Sigma_k$ , where  $\lambda_{k,d}$  is the  $d$ 'th eigenvalue of  $\Sigma_k$ , and  $\mathbf{v}_{k,d}$  is the  $d$ 'th eigenvector.  
37 Then we can rewrite the above as follows:

38

$$\log p(\mathbf{x}_i|c_n = k) = -\sum_{d=1}^D \frac{1}{\lambda_{k,d}} (\mathbf{v}_{k,d}^\top \mathbf{x}_i)^2 + a_k \quad (29.22)$$

41

42 Thus we see that the eigenvectors are acting like templates. Each mixture component has a different  
43 set of templates, each with their own weight (eigenvalue), as illustrated in Figure 29.5. By mixing  
44 these together, we get a powerful model for the statistics of natural image patches. (See [ZW12] for  
45 more analysis of why this simple model works so well, based on the “dead leaves” model of image  
46 formation.)  
47

---

#### 29.2.4.2 Speeding up inference using discriminative models

Although simple and effective, computing  $f(\mathbf{y}) = \operatorname{argmax}_{\mathbf{x}} p(\mathbf{x}|\mathbf{y})$  for each image patch can be slow if the image is large. However, every time we solve this problem, we can store the result, and build up a dataset of  $(\mathbf{y}, f(\mathbf{y}))$  pairs. We can then train an amortized inference network (Section 10.3.7) to learn this  $\mathbf{y} \rightarrow f(\mathbf{y})$  mapping, to speed up future inferences, as proposed in [RW15]. (See also [Par+19] for further speedup tricks.)

An alternative approach is to dispense with the generative model, and to train on an artificially created dataset of the form  $(\mathbf{y}, \mathbf{x})$ , where  $\mathbf{x}$  is a clean natural image, and  $\mathbf{y} = C(\mathbf{x})$  is an artificial corruption of it. We can then train a discriminative model  $\hat{f}(\mathbf{y})$  directly from  $(\mathbf{y}, \mathbf{x})$  pairs. This technique works very well (see e.g., [Luc+18]), but is limited by the form of corruptions  $C$  it is trained on. This means we need to train a different network for every linear operator  $\mathbf{W}$ , and sometimes even for every different noise level  $\sigma^2$ .

#### 29.2.4.3 Blind inverse problems

In the discussion above, we assumed the forward model had the form  $p(\mathbf{y}|\mathbf{x}, \theta) = \mathcal{N}(\mathbf{y}|\mathbf{W}\mathbf{x}, \sigma^2\mathbf{I})$ , where  $\mathbf{W}$  is known. If  $\mathbf{W}$  is not known, then computing  $p(\mathbf{x}|\mathbf{y})$  is known as a **blind inverse problem**.

Such problems are much harder to solve. One approach is to estimate the parameters of the forwards model,  $\mathbf{W}$ , and the latent image,  $\mathbf{x}$ , using an EM-like method from a set of images coming from the same likelihood function. That is, we alternate between estimating  $\hat{\mathbf{x}} = \operatorname{argmax}_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}, \hat{\mathbf{W}})$  in the E step, and estimating  $\hat{\mathbf{W}} = \operatorname{argmax}_{\mathbf{W}} p(\mathbf{y}|\hat{\mathbf{x}}, \mathbf{W})$  in the M step. Some encouraging results of this approach are shown in [Ani+18]. (They use a GAN prior for  $p(\mathbf{x})$  rather than a GMM.)

In cases where we get two independent noisy samples,  $\mathbf{y}_1$  and  $\mathbf{y}_2$ , generated from the same underlying image  $\mathbf{x}$ , then we can avoid having to explicitly learn an image prior  $p(\mathbf{x})$ , and can instead directly learn an estimator for the posterior mode,  $f(\mathbf{y}) = \operatorname{argmax}_{\mathbf{x}} p(\mathbf{x}|\mathbf{y})$ , without needing access to the latent image  $\mathbf{x}$ , by exploiting a form of cycle consistency; see [XC19] for details.

### 29.3 Factor analysis

In this section, we discuss a simple latent factor model in which the prior  $p(\mathbf{z})$  is Gaussian, and the likelihood  $p(\mathbf{x}|\mathbf{z})$  is also Gaussian, using a linear decoder  $f(\mathbf{z})$  for the mean. This family includes many important special cases, such as PCA, as we discuss below.

#### 29.3.1 Vanilla factor analysis

Factor analysis corresponds to the following linear-Gaussian latent variable generative model:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \tag{29.23}$$

$$p(\mathbf{x}|\mathbf{z}, \theta) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \boldsymbol{\Psi}) \tag{29.24}$$

where  $\mathbf{W}$  is a  $D \times L$  matrix, known as the **factor loading matrix**, and  $\boldsymbol{\Psi}$  is a diagonal  $D \times D$  covariance matrix.

---

1    **29.3.1.1 FA as a Gaussian with low-rank plus diagonal covariance**

3    FA can be thought of as a low-rank version of a Gaussian distribution. To see this, note that the  
4    induced marginal distribution  $p(\mathbf{x}|\boldsymbol{\theta})$  is a Gaussian (see Equation (2.62) for the derivation):  
5

6    
$$p(\mathbf{x}|\boldsymbol{\theta}) = \int \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \boldsymbol{\Psi}) \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) d\mathbf{z} \quad (29.25)$$
  
7

8    
$$= \mathcal{N}(\mathbf{x}|\mathbf{W}\boldsymbol{\mu}_0 + \boldsymbol{\mu}, \boldsymbol{\Psi} + \mathbf{W}\boldsymbol{\Sigma}_0\mathbf{W}^T) \quad (29.26)$$
  
9

10   The first and second moments can be derived as follows:  
11

11    
$$\mathbb{E}[\mathbf{x}] = \mathbf{W}\boldsymbol{\mu}_0 + \boldsymbol{\mu} \quad (29.27)$$
  
12    
$$\text{Cov}[\mathbf{x}] = \mathbf{W}\text{Cov}[\mathbf{z}]\mathbf{W}^T + \boldsymbol{\Psi} = \mathbf{W}\boldsymbol{\Sigma}_0\mathbf{W}^T + \boldsymbol{\Psi}$$
  
13

14   From this, we see that we can set  $\boldsymbol{\mu}_0 = \mathbf{0}$  without loss of generality, since we can always absorb  
15    $\mathbf{W}\boldsymbol{\mu}_0$  into  $\boldsymbol{\mu}$ . Similarly, we can set  $\boldsymbol{\Sigma}_0 = \mathbf{I}$  without loss of generality, since we can always absorb a  
16   correlated prior by using a new weight matrix,  $\tilde{\mathbf{W}} = \mathbf{W}\boldsymbol{\Sigma}_0^{-\frac{1}{2}}$ , since then

17    
$$\text{Cov}[\mathbf{x}] = \mathbf{W}\boldsymbol{\Sigma}_0\mathbf{W}^T + \boldsymbol{\Psi} = \tilde{\mathbf{W}}\tilde{\mathbf{W}}^T + \boldsymbol{\Psi} \quad (29.28)$$
  
18

19   Finally, we see that we should restrict  $\boldsymbol{\Psi}$  to be diagonal, otherwise we could set  $\tilde{\mathbf{W}} = \mathbf{0}$ , thus ignoring  
20   the latent factors, while still being able to model any covariance. After these simplifications we have  
21   the final model:

22    
$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}) \quad (29.29)$$
  
23

24    
$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \boldsymbol{\Psi}) \quad (29.30)$$
  
25

25    
$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^T + \boldsymbol{\Psi}) \quad (29.31)$$
  
26

27   For example, suppose where  $L = 1$ ,  $D = 2$  and  $\boldsymbol{\Psi} = \sigma^2\mathbf{I}$ . We illustrate the generative process in  
28   this case in Figure 29.6. We can think of this as taking an isotropic Gaussian “spray can”, representing  
29   the likelihood  $p(\mathbf{x}|\mathbf{z})$ , and “sliding it along” the 1d line defined by  $\mathbf{w}\mathbf{z} + \boldsymbol{\mu}$  as we vary the 1d latent  
30   prior  $\mathbf{z}$ . This induces an elongated (and hence correlated) Gaussian in 2d. That is, the induced  
31   distribution has the form  $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{w}\mathbf{w}^T + \sigma^2\mathbf{I})$ .

32   In general, FA approximates the covariance matrix of the visible vector using a low-rank decompo-  
33   sition:

34    
$$\mathbf{C} = \text{Cov}[\mathbf{x}] = \mathbf{W}\mathbf{W}^T + \boldsymbol{\Psi} \quad (29.32)$$
  
35

36   This only uses  $O(LD)$  parameters, which allows a flexible compromise between a full covariance  
37   Gaussian, with  $O(D^2)$  parameters, and a diagonal covariance, with  $O(D)$  parameters.  
38

39    **29.3.1.2 Computing the posterior**

40   We can compute the posterior over the latent codes,  $p(\mathbf{z}|\mathbf{x})$ , using Bayes rule for Gaussians. In  
41   particular, from Equation (2.59), we have  
42

43    
$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_{z|x}, \boldsymbol{\Sigma}_{z|x}) \quad (29.33)$$
  
44

44    
$$\boldsymbol{\Sigma}_{z|x}^{-1} = \mathbf{I} + \mathbf{W}^T\boldsymbol{\Psi}^{-1}\mathbf{W} \quad (29.34)$$
  
45

46    
$$\boldsymbol{\mu}_{z|x} = \boldsymbol{\Sigma}_{z|x}[\mathbf{W}^T\boldsymbol{\Psi}^{-1}(\mathbf{x} - \boldsymbol{\mu})] \quad (29.35)$$
  
47

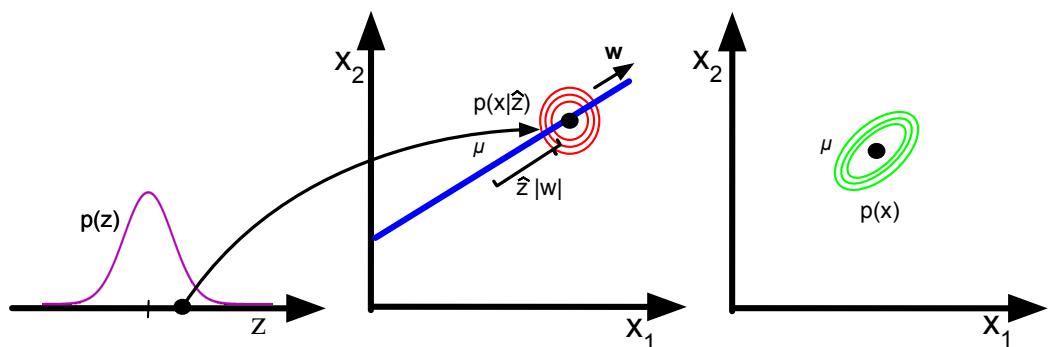


Figure 29.6: Illustration of the FA generative process, where we have  $L = 1$  latent dimension generating  $D = 2$  observed dimensions; we assume  $\Psi = \sigma^2 \mathbf{I}$ . The latent factor has value  $z \in \mathbb{R}$ , sampled from  $p(z)$ ; this gets mapped to a 2d offset  $\delta = zw$ , where  $w \in \mathbb{R}^2$ , which gets added to  $\mu$  to define a Gaussian  $p(\mathbf{x}|z) = \mathcal{N}(\mathbf{x}|\mu + \delta, \sigma^2 \mathbf{I})$ . By integrating over  $z$ , we “slide” this circular Gaussian “spray can” along the principal component axis  $w$ , which induces elliptical Gaussian contours in  $\mathbf{x}$  space centered on  $\mu$ . Adapted from Figure 12.9 of [Bis06].

This can also be extended to handle missing data (if we make the missing at random assumption — see Section 22.3.5 for discussion). In particular, let us partition  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ ,  $\mathbf{W} = [\mathbf{W}_1, \mathbf{W}_2]$ , and  $\mu = [\mu_1, \mu_2]$ , and suppose  $\mathbf{x}_2$  is missing (unknown). From Section 2.3.6, we have

$$p(z|\mathbf{x}_1) = \mathcal{N}(z|\mu_{z|1}, \Sigma_{z|1}) \quad (29.36)$$

$$\Sigma_{z|1}^{-1} = \mathbf{I} + \mathbf{W}_1^\top \Sigma_{11}^{-1} \mathbf{W}_1 \quad (29.37)$$

$$\mu_{z|1} = \Sigma_{z|1} [\mathbf{W}_1^\top \Sigma_{11}^{-1} (\mathbf{x}_1 - \mu_1)] \quad (29.38)$$

where  $\Sigma_{11}$  is the top left block of  $\Psi$ .

### 29.3.1.3 Computing the likelihood

In this section, we discuss how to efficiently compute the log (marginal) likelihood, which is given by

$$\log p(\mathbf{x}|\mu, \mathbf{C}) = -\frac{1}{2} [D \log(2\pi) + \log \det(\mathbf{C}) + \tilde{\mathbf{x}}^\top \mathbf{C}^{-1} \tilde{\mathbf{x}}] \quad (29.39)$$

where  $\tilde{\mathbf{x}} = \mathbf{x} - \mu$ , and  $\mathbf{C} = \mathbf{W}\mathbf{W}^\top + \Psi$ . We can avoid inverting the  $D \times D$  matrix  $\mathbf{C}$  by using the matrix inversion lemma:

$$\mathbf{C}^{-1} = (\mathbf{W}\mathbf{W}^\top + \Psi)^{-1} \quad (29.40)$$

$$= \Psi^{-1} - \Psi^{-1} \mathbf{W} (\mathbf{I} + \mathbf{W}^\top \Psi^{-1} \mathbf{W})^{-1} \mathbf{W}^\top \Psi^{-1} \quad (29.41)$$

$$= \Psi^{-1} - \Psi^{-1} \mathbf{W} \mathbf{L}^{-1} \mathbf{W}^\top \Psi^{-1} \quad (29.42)$$

where  $\mathbf{L} = \mathbf{I} + \mathbf{W}^\top \Psi^{-1} \mathbf{W}$  is  $L \times L$ . The Mahalanobis distance is then given by

$$\tilde{\mathbf{x}}^\top \mathbf{C}^{-1} \tilde{\mathbf{x}} = \tilde{\mathbf{x}}^\top [\Psi^{-1} \tilde{\mathbf{x}} - \Psi^{-1} \mathbf{W} \mathbf{L}^{-1} (\mathbf{W}^\top \Psi^{-1} \tilde{\mathbf{x}})] \quad (29.43)$$

1 which takes  $O(L^3 + LD)$  to compute. From the matrix determinant lemma, the log determinant is  
2 given by  
3

4  $\log \det(\mathbf{C}) = \log \det(\mathbf{L}) + \log \det(\boldsymbol{\Psi})$  (29.44)  
5

6 which takes  $O(L^3 + D)$  to compute.  
7

### 8 29.3.1.4 Unidentifiability of the parameters 9

10 The parameters of a FA model are unidentifiable. To see this, consider a model with weights  $\mathbf{W}$  and  
11 observation covariance  $\boldsymbol{\Psi}$ . We have

12  $\text{Cov}[\mathbf{x}] = \mathbf{W}\mathbb{E}[zz^\top]\mathbf{W}^\top + \mathbb{E}[\epsilon\epsilon^\top] = \mathbf{WW}^\top + \boldsymbol{\Psi}$  (29.45)  
13

14 Now consider a different model with weights  $\tilde{\mathbf{W}} = \mathbf{WR}$ , where  $\mathbf{R}$  is an arbitrary orthogonal rotation  
15 matrix, satisfying  $\mathbf{RR}^\top = \mathbf{I}$ . This has the same likelihood, since  
16

17  $\text{Cov}[\mathbf{x}] = \tilde{\mathbf{W}}\mathbb{E}[zz^\top]\tilde{\mathbf{W}}^\top + \mathbb{E}[\epsilon\epsilon^\top] = \mathbf{WRR}^\top\mathbf{W}^\top + \boldsymbol{\Psi} = \mathbf{WW}^\top + \boldsymbol{\Psi}$  (29.46)  
18

19 Geometrically, multiplying  $\mathbf{W}$  by an orthogonal matrix is like rotating  $\mathbf{z}$  before generating  $\mathbf{x}$ ; but  
20 since  $\mathbf{z}$  is drawn from an isotropic Gaussian, this makes no difference to the likelihood. Consequently,  
21 we cannot uniquely identify  $\mathbf{W}$ , and therefore cannot uniquely identify the latent factors, either.  
22 This is called the “**factor rotations problem**” (see e.g., [Dar80]).

23 To break this symmetry, several solutions can be used, as we discuss below.

- 24 • **Forcing  $\mathbf{W}$  to have orthonormal columns.** Perhaps the simplest solution to the identifiability  
25 problem is to force  $\mathbf{W}$  to have orthonormal columns. This is the approach adopted by PCA.  
26 The resulting posterior estimate will then be unique, up to permutation of the latent dimensions.  
27 (In PCA, this ordering ambiguity is resolved by sorting the dimensions in order of decreasing  
28 eigenvalues of  $\mathbf{W}$ .)
- 29 • **Forcing  $\mathbf{W}$  to be lower triangular.** One way to resolve permutation unidentifiability, which  
30 is popular in the Bayesian community (e.g., [LW04]), is to ensure that the first visible feature is  
31 only generated by the first latent factor, the second visible feature is only generated by the first  
32 two latent factors, and so on. For example, if  $L = 3$  and  $D = 4$ , the corresponding factor loading  
33 matrix is given by

34 
$$\mathbf{W} = \begin{pmatrix} w_{11} & 0 & 0 \\ w_{21} & w_{22} & 0 \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{pmatrix}$$
 (29.47)  
35  
36  
37  
38

39 We also require that  $w_{kk} > 0$  for  $k = 1 : L$ . The total number of parameters in this constrained  
40 matrix is  $D + DL - L(L-1)/2$ , which is equal to the number of uniquely identifiable parameters in  
41 FA.<sup>1</sup> The disadvantage of this method is that the first  $L$  visible variables, known as the **founder**  
42 **variables**, affect the interpretation of the latent factors, and so must be chosen carefully.  
43

44 1. We get  $D$  parameters for  $\boldsymbol{\Psi}$  and  $DL$  for  $\mathbf{W}$ , but we need to remove  $L(L-1)/2$  degrees of freedom coming from  $\mathbf{R}$ ,  
45 since that is the dimensionality of the space of orthogonal matrices of size  $L \times L$ . To see this, note that there are  $L-1$   
46 free parameters in  $\mathbf{R}$  in the first column (since the column vector must be normalized to unit length), there are  $L-2$   
47 free parameters in the second column (which must be orthogonal to the first), and so on.

- **Sparsity promoting priors on the weights.** Instead of pre-specifying which entries in  $\mathbf{W}$  are zero, we can encourage the entries to be zero, using  $\ell_1$  regularization [ZHT06], ARD [Bis99; AB08], or spike-and-slab priors [Rat+09]. This is called sparse factor analysis. This does not necessarily ensure a unique MAP estimate, but it does encourage interpretable solutions.
- **Choosing an informative rotation matrix.** There are a variety of heuristic methods that try to find rotation matrices  $\mathbf{R}$  which can be used to modify  $\mathbf{W}$  (and hence the latent factors) so as to try to increase the interpretability, typically by encouraging them to be (approximately) sparse. One popular method is known as **varimax** [Kai58].
- **Use of non-Gaussian priors for the latent factors.** If we replace the prior on the latent variables,  $p(\mathbf{z})$ , with a non-Gaussian distribution, we can sometimes uniquely identify  $\mathbf{W}$ , as well as the latent factors. See e.g., [KKH20] for details.

### 29.3.2 Probabilistic PCA

In this section, we consider a special case of the factor analysis model in which  $\mathbf{W}$  has orthonormal columns,  $\Psi = \sigma^2\mathbf{I}$  and  $\mu = \mathbf{0}$ . This model is called **probabilistic principal components analysis (PPCA)** [TB99], or **sensible PCA** [Row97]. The marginal distribution on the visible variables has the form

$$p(\mathbf{x}|\theta) = \int \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z}, \sigma^2\mathbf{I})\mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})d\mathbf{z} = \mathcal{N}(\mathbf{x}|\mu, \mathbf{C}) \quad (29.48)$$

where

$$\mathbf{C} = \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I} \quad (29.49)$$

#### 29.3.2.1 Connection to PCA

The log likelihood for PPCA is given by

$$\log p(\mathbf{X}|\mu, \mathbf{W}, \sigma^2) = -\frac{ND}{2} \log(2\pi) - \frac{N}{2} \log |\mathbf{C}| - \frac{1}{2} \sum_{n=1}^N (\mathbf{x}_n - \mu)^\top \mathbf{C}^{-1} (\mathbf{x}_n - \mu) \quad (29.50)$$

The MLE for  $\mu$  is  $\bar{\mathbf{x}}$ . Plugging in gives

$$\log p(\mathbf{X}|\mu, \mathbf{W}, \sigma^2) = -\frac{N}{2} [D \log(2\pi) + \log |\mathbf{C}| + \text{tr}(\mathbf{C}^{-1} \mathbf{S})] \quad (29.51)$$

where  $\mathbf{S} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^\top$  is the empirical covariance matrix.

In [TB99; Row97] they show that the maximum of this objective must satisfy

$$\mathbf{W} = \mathbf{U}_L (\Lambda_L - \sigma^2 \mathbf{I})^{\frac{1}{2}} \mathbf{R} \quad (29.52)$$

where  $\mathbf{U}_L$  is a  $D \times L$  matrix whose columns are given by the  $L$  eigenvectors of  $\mathbf{S}$  with largest eigenvalues,  $\Lambda_L$  is the  $L \times L$  diagonal matrix of corresponding eigenvalues, and  $\mathbf{R}$  is an arbitrary  $L \times L$  orthogonal matrix, which (WLOG) we can take to be  $\mathbf{R} = \mathbf{I}$ .

If we plug in the MLE for  $\mathbf{W}$ , we find the covariance for the predictive distribution to be

$$\mathbf{C} = \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I} = \mathbf{U}_L (\Lambda_L - \sigma^2 \mathbf{I}) \mathbf{U}_L^\top + \sigma^2 \mathbf{I} \quad (29.53)$$

In the noise-free limit, where  $\sigma^2 = 0$ , we see that  $\mathbf{W}_{\text{mle}} = \mathbf{U}_L \Lambda_L^{\frac{1}{2}}$  and  $\mathbf{C} = \mathbf{WW}^\top$ , as in PCA.  
The MLE for the observation variance is

$$\sigma^2 = \frac{1}{D-L} \sum_{i=L+1}^D \lambda_i \quad (29.54)$$

which is the average distortion associated with the discarded dimensions. If  $L = D$ , then the estimated noise is 0, since the model collapses to  $\mathbf{z} = \mathbf{x}$ .

### 29.3.2.2 Computing the posterior

To use PPCA as an alternative to PCA, we need to compute the posterior mean  $\mathbb{E}[\mathbf{z}|\mathbf{x}]$ , which is the equivalent of the PCA encoder model. Using the factor analysis results from Section 29.3.1.2, we have

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\sigma^{-2}\Sigma\mathbf{W}^\top(\mathbf{x} - \boldsymbol{\mu}), \Sigma) \quad (29.55)$$

where

$$\Sigma^{-1} = \mathbf{I} + \sigma^{-2}\mathbf{W}^\top\mathbf{W} = \frac{1}{\sigma^2}(\sigma^2\mathbf{I} + \mathbf{W}^\top\mathbf{W}) = \frac{1}{\sigma^2}\mathbf{M} \quad (29.56)$$

To compute this efficiently, we can use the matrix inversion lemma to write

$$(\mathbf{WW}^\top + \sigma^2\mathbf{I})^{-1} = (\mathbf{I} - \mathbf{W}(\mathbf{W}^\top\mathbf{W} + \sigma^2\mathbf{I})^{-1}\mathbf{W}^\top)/\sigma^2 \quad (29.57)$$

If we define

$$\mathbf{M} = (\mathbf{W}^\top\mathbf{W} + \sigma^2\mathbf{I}) \quad (29.58)$$

the posterior covariance simplifies to

$$\Sigma = \sigma^2\mathbf{M} \quad (29.59)$$

Hence

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mathbf{M}^{-1}\mathbf{W}^\top(\mathbf{x} - \boldsymbol{\mu}), \sigma^2\mathbf{M}^{-1}) \quad (29.60)$$

In the  $\sigma^2 = 0$  limit, we have  $\mathbf{M} = \mathbf{W}^\top\mathbf{W}$  and so

$$\mathbb{E}[\mathbf{z}|\mathbf{x}] = (\mathbf{W}^\top\mathbf{W})^{-1}\mathbf{W}^\top(\mathbf{x} - \bar{\mathbf{x}}) \quad (29.61)$$

This is the orthogonal projection of the data into the latent space, as in standard PCA.

### 29.3.2.3 Computing the likelihood

To compute  $p(\mathbf{x}|\boldsymbol{\mu}, \mathbf{C})$  efficiently, we need to evaluate  $\mathbf{C}^{-1}$  and  $\log|\mathbf{C}|$  efficiently. To do this, we can use the matrix inversion lemma to write

$$\mathbf{C}^{-1} = (\mathbf{WW}^\top + \sigma^2\mathbf{I})^{-1} = (\mathbf{I} - \mathbf{W}(\mathbf{W}^\top\mathbf{W} + \sigma^2\mathbf{I})^{-1}\mathbf{W}^\top)/\sigma^2 = \sigma^{-2} [\mathbf{I} - \mathbf{WM}^{-1}\mathbf{W}^\top] \quad (29.62)$$

where  $\mathbf{M}$  is defined in Equation (29.58). When we plug in the MLE for  $\mathbf{W}$  from Equation (29.52) (using  $\mathbf{R} = \mathbf{I}$ ) we find

$$\mathbf{C}^{-1} = \sigma^{-2} [\mathbf{I} - \mathbf{U}_L(\Lambda_L - \sigma^2 \mathbf{I})\Lambda_L^{-1}\mathbf{U}_L^\top] = \sigma^{-2}(\mathbf{I} - \mathbf{U}_L\mathbf{K}\mathbf{U}_L^\top) \quad (29.63)$$

where  $\mathbf{K} = \text{diag}(k_d)$  and  $k_d = 1 - \frac{\sigma_d^2}{\lambda_d}$ . Similarly one can show

$$\log |\mathbf{C}| = \log |\mathbf{W}\mathbf{W}^\top + \sigma^2 \mathbf{I}| = (D - L) \log \sigma^2 + \sum_{j=1}^L \log \lambda_j \quad (29.64)$$

$$= D \log \sigma^2 - \sum_{d=1}^L \log(1 - k_d) \quad (29.65)$$

#### 29.3.2.4 EM for (P)PCA

In Section 29.3.2.1, we showed how to fit the (P)PCA model using eigenvector method. we can also use EM, which has some advantages which we discuss in Section 29.3.2.5. This relies on the probabilistic formulation of PCA. However the algorithm continues to work in the zero noise limit,  $\sigma^2 = 0$ , as shown by [Row97].

In particular, let  $\tilde{\mathbf{Z}} = \mathbf{Z}^\top$  be a  $L \times N$  matrix storing the posterior means (low-dimensional representations) along its columns. Similarly, let  $\tilde{\mathbf{X}} = \mathbf{X}^\top$  store the original data along its columns. From Equation (29.60), when  $\sigma^2 = 0$ , we have

$$\tilde{\mathbf{Z}} = (\mathbf{W}^T \mathbf{W})^{-1} \mathbf{W}^T \tilde{\mathbf{X}} \quad (29.66)$$

This constitutes the E step. Notice that this is just an orthogonal projection of the data.

From Equation 29.110, the M step is given by

$$\hat{\mathbf{W}} = \left[ \sum_n \mathbf{x}_n \mathbb{E}[\mathbf{z}_n]^T \right] \left[ \sum_n \mathbb{E}[\mathbf{z}_n] \mathbb{E}[\mathbf{z}_n]^T \right]^{-1} \quad (29.67)$$

where we exploited the fact that  $\Sigma = \text{Cov}[\mathbf{z}|\mathbf{x}, \theta] = 0\mathbf{I}$  when  $\sigma^2 = 0$ .

It is worth comparing this expression to the MLE for multi-output linear regression, which has the form  $\mathbf{W} = (\sum_i \mathbf{y}_i \mathbf{x}_i^T)(\sum_i \mathbf{x}_i \mathbf{x}_i^T)^{-1}$ . Thus we see that the M step is like linear regression where we replace the observed inputs by the expected values of the latent variables.

In summary, here is the entire algorithm:

$$\tilde{\mathbf{Z}} = (\mathbf{W}^T \mathbf{W})^{-1} \mathbf{W}^T \tilde{\mathbf{X}} \text{ (E step)} \quad (29.68)$$

$$\mathbf{W} = \tilde{\mathbf{X}} \tilde{\mathbf{Z}}^T (\tilde{\mathbf{Z}} \tilde{\mathbf{Z}}^T)^{-1} \text{ (M step)} \quad (29.69)$$

[TB99] showed that the only stable fixed point of the EM algorithm is the globally optimal solution. That is, the EM algorithm converges to a solution where  $\mathbf{W}$  spans the same linear subspace as that defined by the first  $L$  eigenvectors. However, if we want  $\mathbf{W}$  to be orthogonal, and to contain the eigenvectors in descending order of eigenvalue, we have to orthogonalize the resulting matrix (which can be done quite cheaply). Alternatively, we can modify EM to give the principal basis directly [AO03].

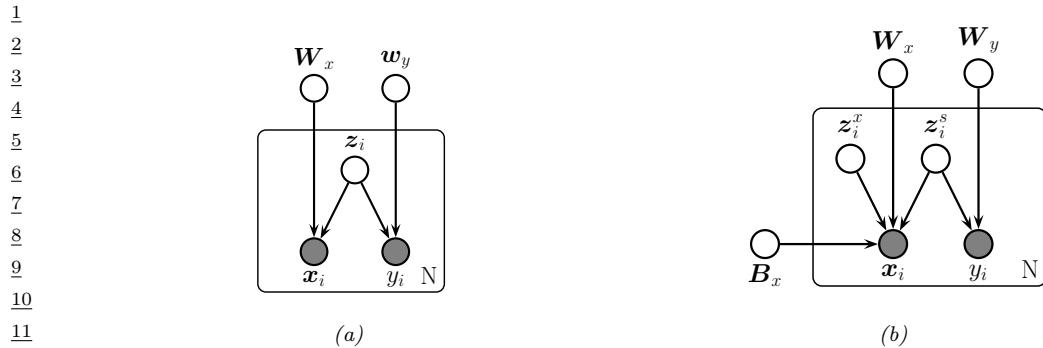


Figure 29.7: Gaussian latent factor models for paired data. (a) Supervised PCA. (b) Partial least squares.

### 29.3.2.5 EM vs eigenvector methods

EM for PCA has the following advantages over eigenvector methods:

- EM can be faster. In particular, assuming  $N, D \gg L$ , the dominant cost of EM is the projection operation in the E step, so the overall time is  $O(TLND)$ , where  $T$  is the number of iterations. [Row97] showed experimentally that the number of iterations is usually very small (the mean was 3.6), regardless of  $N$  or  $D$ . (This results depends on the ratio of eigenvalues of the empirical covariance matrix.) This is much faster than the  $O(\min(ND^2, DN^2))$  time required by straightforward eigenvector methods, although more sophisticated eigenvector methods, such as the Lanczos algorithm, have running times comparable to EM.
- EM can be implemented in an online fashion, i.e., we can update our estimate of  $\mathbf{W}$  as the data streams in.
- EM can handle missing data in a simple way (see e.g., [IR10; DJ15]). (See Section 22.3.5 for more discussion of missing data.)
- EM can be extended to handle mixtures of PPCA/ FA models (see Section 29.4).
- EM can be modified to variational EM or to variational Bayes EM to fit more complex models (see e.g., Section 29.3.4).

### 29.3.3 Factor analysis models for paired data

In this section, we discuss linear-Gaussian factor analysis models when we have two kinds of observed variables,  $\mathbf{x} \in \mathbb{R}^{D_x}$  and  $\mathbf{y} \in \mathbb{R}^{D_y}$ , which are paired. These often correspond to different sensors or modalities (e.g., images and sound). We follow the presentation of [Vir10].

---

### 29.3.3.1 Supervised PCA

If we have two observed signals, we can model the joint  $p(\mathbf{x}, \mathbf{y})$  using a shared low-dimensional representation using the following linear Gaussian model:

$$p(\mathbf{z}_n) = \mathcal{N}(\mathbf{z}_n | \mathbf{0}, \mathbf{I}_L) \quad (29.70)$$

$$p(\mathbf{x}_n | \mathbf{z}_n, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_n | \mathbf{W}_x \mathbf{z}_n, \sigma_x^2 \mathbf{I}_{D_x}) \quad (29.71)$$

$$p(\mathbf{y}_n | \mathbf{z}_n, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{y}_n | \mathbf{W}_y \mathbf{z}_n, \sigma_y^2 \mathbf{I}_{D_y}) \quad (29.72)$$

This is illustrated as a graphical model in Figure 29.7a. The intuition is that  $\mathbf{z}_n$  is a shared latent subspace, that captures features that  $\mathbf{x}_n$  and  $\mathbf{y}_n$  have in common. The variance terms  $\sigma_x$  and  $\sigma_y$  control how much emphasis the model puts on the two different signals.

The above model is called **supervised PCA** [Yu+06]. If we put a prior on the parameters  $\boldsymbol{\theta} = (\mathbf{W}_x, \mathbf{W}_y, \sigma_x, \sigma_y)$ , it is called **Bayesian factor regression** model of [Wes03].

We can marginalize out  $\mathbf{z}_n$  to get  $p(\mathbf{y}_n | \mathbf{x}_n, \boldsymbol{\theta})$ . If  $\mathbf{y}_n$  is a scalar, this becomes

$$p(\mathbf{y}_n | \mathbf{x}_n, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{y}_n | \mathbf{x}_n^\top \mathbf{v}, \mathbf{w}_y^\top \mathbf{C} \mathbf{w}_y + \sigma_y^2) \quad (29.73)$$

$$\mathbf{C} = (\mathbf{I} + \sigma_x^{-2} \mathbf{W}_x^\top \mathbf{W}_x)^{-1} \quad (29.74)$$

$$\mathbf{v} = \sigma_x^{-2} \mathbf{W}_x \mathbf{C} \mathbf{w}_y \quad (29.75)$$

To apply this to the classification setting, we can replace the Gaussian  $p(\mathbf{y}|\mathbf{z})$  with a logistic regression model:

$$p(\mathbf{y}_n | \mathbf{z}_n, \boldsymbol{\theta}) = \text{Ber}(\mathbf{y}_n | \sigma(\mathbf{w}_y^\top \mathbf{z}_n)) \quad (29.76)$$

In this case, we can no longer compute the marginal posterior predictive  $p(\mathbf{y}_n | \mathbf{x}_n, \boldsymbol{\theta})$  in closed form, but we use can techniques similar to exponential family PCA (see [Guo09] for details).

The above model is completely symmetric in  $\mathbf{x}$  and  $\mathbf{y}$ . If our goal is to predict  $\mathbf{y}$  from  $\mathbf{x}$  via the latent bottleneck  $\mathbf{z}$ , then we might want to upweight the likelihood term for  $\mathbf{y}$ , as proposed in [Ris+08]. This gives

$$p(\mathbf{X}, \mathbf{Y}, \mathbf{Z} | \boldsymbol{\theta}) = p(\mathbf{Y} | \mathbf{Z}, \mathbf{W}_y) p(\mathbf{X} | \mathbf{Z}, \mathbf{W}_x)^\alpha p(\mathbf{Z}) \quad (29.77)$$

where  $\alpha \leq 1$  controls the relative importance of modeling the two sources. The value of  $\alpha$  can be chosen by cross-validation.

### 29.3.3.2 Partial least squares

We now consider an asymmetric or more “discriminative” form of supervised PCA. The key idea is to allow some of the (co)variance in the input features to be explained by its own subspace,  $\mathbf{z}_i^x$ , and to let the rest of the subspace,  $\mathbf{z}_i^s$ , be shared between input and output. The model has the form

$$p(\mathbf{z}_i) = \mathcal{N}(\mathbf{z}_i^s | \mathbf{0}, \mathbf{I}_{L_s}) \mathcal{N}(\mathbf{z}_i^x | \mathbf{0}, \mathbf{I}_{L_x}) \quad (29.78)$$

$$p(\mathbf{y}_i | \mathbf{z}_i) = \mathcal{N}(\mathbf{W}_y \mathbf{z}_i^s + \boldsymbol{\mu}_y, \sigma^2 \mathbf{I}_{D_y}) \quad (29.79)$$

$$p(\mathbf{x}_i | \mathbf{z}_i) = \mathcal{N}(\mathbf{W}_x \mathbf{z}_i^s + \mathbf{B}_x \mathbf{z}_i^x + \boldsymbol{\mu}_x, \sigma^2 \mathbf{I}_{D_x}) \quad (29.80)$$

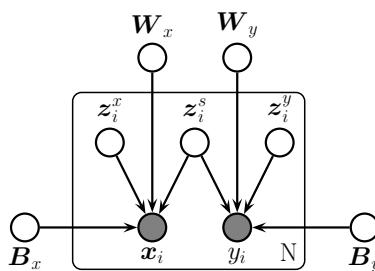


Figure 29.8: Canonical correlation analysis as a PGM.

See Figure 29.7b. The corresponding induced distribution on the visible variables has the form

$$p(\mathbf{v}_i|\boldsymbol{\theta}) = \int \mathcal{N}(\mathbf{v}_i|\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \sigma^2\mathbf{I})\mathcal{N}(\mathbf{z}_i|\mathbf{0}, \mathbf{I})d\mathbf{z}_i = \mathcal{N}(\mathbf{v}_i|\boldsymbol{\mu}, \mathbf{WW}^T + \sigma^2\mathbf{I}) \quad (29.81)$$

where  $\mathbf{v}_i = (\mathbf{x}_i; \mathbf{y}_i)$ ,  $\boldsymbol{\mu} = (\boldsymbol{\mu}_y; \boldsymbol{\mu}_x)$  and

$$\mathbf{W} = \begin{pmatrix} \mathbf{W}_y & \mathbf{0} \\ \mathbf{W}_x & \mathbf{B}_x \end{pmatrix} \quad (29.82)$$

$$\mathbf{WW}^T = \begin{pmatrix} \mathbf{W}_y \mathbf{W}_y^T & \mathbf{W}_x \mathbf{W}_x^T \\ \mathbf{W}_x \mathbf{W}_x^T & \mathbf{W}_x \mathbf{W}_x^T + \mathbf{B}_x \mathbf{B}_x^T \end{pmatrix} \quad (29.83)$$

We should choose  $L$  large enough so that the shared subspace does not capture covariate-specific variation.

MLE in this model is equivalent to the technique of **partial least squares (PLS)** [Gus01; Nou+02; Sun+09]. This model can be also be generalized to discrete data using the exponential family [Vir10].

### 29.3.3.3 Canonical correlation analysis

We now consider a symmetric unsupervised version of PLS, in which we allow each view to have its own ‘‘private’’ subspace, but there is also a shared subspace. If we have two observed variables,  $\mathbf{x}_i$  and  $\mathbf{y}_i$ , then we have three latent variables,  $\mathbf{z}_i^s \in \mathbb{R}^{L_s}$  which is shared,  $\mathbf{z}_i^x \in \mathbb{R}^{L_x}$  and  $\mathbf{z}_i^y \in \mathbb{R}^{L_y}$  which are private. We can write the model as follows [BJ05]:

$$p(\mathbf{z}_i) = \mathcal{N}(\mathbf{z}_i^s|\mathbf{0}, \mathbf{I}_{L_s})\mathcal{N}(\mathbf{z}_i^x|\mathbf{0}, \mathbf{I}_{L_x})\mathcal{N}(\mathbf{z}_i^y|\mathbf{0}, \mathbf{I}_{L_y}) \quad (29.84)$$

$$p(\mathbf{x}_i|\mathbf{z}_i) = \mathcal{N}(\mathbf{x}_i|\mathbf{B}_x \mathbf{z}_i^x + \mathbf{W}_x \mathbf{z}_i^s + \boldsymbol{\mu}_x, \sigma^2 \mathbf{I}_{D_x}) \quad (29.85)$$

$$p(\mathbf{y}_i|\mathbf{z}_i) = \mathcal{N}(\mathbf{y}_i|\mathbf{B}_y \mathbf{z}_i^y + \mathbf{W}_y \mathbf{z}_i^s + \boldsymbol{\mu}_y, \sigma^2 \mathbf{I}_{D_y}) \quad (29.86)$$

See Figure 29.8 The corresponding observed joint distribution has the form

$$p(\mathbf{v}_i|\boldsymbol{\theta}) = \int \mathcal{N}(\mathbf{v}_i|\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \sigma^2\mathbf{I})\mathcal{N}(\mathbf{z}_i|\mathbf{0}, \mathbf{I})d\mathbf{z}_i = \mathcal{N}(\mathbf{v}_i|\boldsymbol{\mu}, \mathbf{WW}^T + \sigma^2\mathbf{I}_D) \quad (29.87)$$

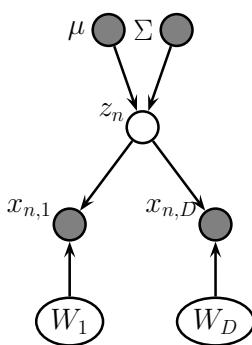


Figure 29.9: Exponential family PCA model as a PGM-D.

where

$$\mathbf{W} = \begin{pmatrix} \mathbf{W}_x & \mathbf{B}_x & \mathbf{0} \\ \mathbf{W}_y & \mathbf{0} & \mathbf{B}_y \end{pmatrix} \quad (29.88)$$

$$\mathbf{WW}^T = \begin{pmatrix} \mathbf{W}_x \mathbf{W}_x^T + \mathbf{B}_x \mathbf{B}_x^T & \mathbf{W}_x \mathbf{W}_y^T \\ \mathbf{W}_y \mathbf{W}_y^T & \mathbf{W}_y \mathbf{W}_y^T + \mathbf{B}_y \mathbf{B}_y^T \end{pmatrix} \quad (29.89)$$

[BJ05] showed that MLE for this model is equivalent to a classical statistical method known as **canonical correlation analysis** or **CCA** [Hot36]. However, the PGM perspective allows us to easily generalize to multiple kinds of observations (this is known as **generalized CCA** [Hor61]) or to nonlinear models (this is known as **deep CCA** [WLL16; SNM16]), or exponential family CCA [KVK10]. See [Uur+17] for further discussion of CCA and its extensions.

### 29.3.4 Factor analysis with exponential family likelihoods

So far we have assumed the observed data is real-valued, so  $\mathbf{x}_n \in \mathbb{R}^D$ . If we want to model other kinds of data (e.g., binary or categorical), we can simply replace the Gaussian output distribution with a suitable member of the exponential family, where the natural parameters are given by a linear function of  $\mathbf{z}_n$ . That is, we use

$$p(\mathbf{x}_n | \mathbf{z}_n) = \exp(\mathcal{T}(\mathbf{x})^\top \boldsymbol{\theta} + h(\mathbf{x}) - g(\boldsymbol{\theta})) \quad (29.90)$$

where the  $N \times D$  matrix of natural parameters is assumed to be given by the low rank decomposition  $\boldsymbol{\Theta} = \mathbf{Z}\mathbf{W}$ , where  $\mathbf{Z}$  is  $N \times L$  and  $\mathbf{W}$  is  $L \times D$ . The resulting model is called **exponential family factor analysis**

Unlike the linear-Gaussian FA, we cannot compute the exact posterior  $p(\mathbf{z}_n | \mathbf{x}_n, \mathbf{W})$  due to the lack of conjugacy between the expfam likelihood and the Gaussian prior. Furthermore, we cannot compute the exact marginal likelihood either, which prevents us from finding the optimal MLE.

[CDS02] proposed a coordinate ascent method for a deterministic variant of this model, known as **exponential family PCA**. This alternates between computing a point estimate of  $\mathbf{z}_n$  and  $\mathbf{W}$ . This

1 can be regarded as a degenerate version of variational EM, where the E step uses a delta function  
2 posterior for  $\mathbf{z}_n$ . [GS08] present an improved algorithm that finds the global optimum, and [Ude+16]  
3 presents an extension called **generalized low rank models**, that covers many different kinds of  
4 loss function.  
5

6 However, it is often preferable to use a probabilistic version of the model, rather than computing  
7 point estimates of the latent factors. In this case, we must represent the posterior use a non-degenerate  
8 distribution to avoid overfitting, since the number of latent variables is proportional to the number  
9 of data cases [WCS08]. Fortunately, we can use a non-degenerate posterior, such as a Gaussian, by  
10 optimizing the variational lower bound. We give some examples of this below.

11

### 12 29.3.4.1 Example: binary PCA

13 Consider a factored Bernoulli likelihood:  
14

$$\underline{15} \quad p(\mathbf{x}|\mathbf{z}) = \prod_d \text{Ber}(x_d|\sigma(\mathbf{w}_d^\top \mathbf{z})) \quad (29.91)$$

16 Suppose we observe  $N = 150$  bit vectors of length  $D = 6$ . Each example is generated by choosing one  
17 of three binary prototype vectors, and then by flipping bits at random. See Figure 29.10(a) for the  
18 data. We can fit this using the variational EM algorithm (see [Tip98] for details). We use  $L = 2$  latent  
19 dimensions to allow us to visualize the latent space. In Figure 29.10(b), we plot  $\mathbb{E}[\mathbf{z}_n|\mathbf{x}_n, \hat{\mathbf{W}}]$ . We see  
20 that the projected points group into three distinct clusters, as is to be expected. In Figure 29.10(c),  
21 we plot the reconstructed version of the data, which is computed as follows:  
22

$$\underline{23} \quad p(\hat{x}_{nd} = 1|\mathbf{x}_n) = \int d\mathbf{z}_n p(\mathbf{z}_n|\mathbf{x}_n)p(\hat{x}_{nd}|\mathbf{z}_n) \quad (29.92)$$

24 If we threshold these probabilities at 0.5 (corresponding to a MAP estimate), we get the “denoised”  
25 version of the data in Figure 29.10(d).  
26

### 27 29.3.4.2 Example: categorical PCA

28 We can generalize the model in Section 29.3.4.1 to handle categorical data by using the following  
29 likelihood:

$$\underline{30} \quad p(\mathbf{x}|\mathbf{z}) = \prod_d \text{Cat}(x_d|\sigma(\mathbf{W}_d \mathbf{z})) \quad (29.93)$$

31 We call this **categorical PCA** (**CatPCA**). A variational EM algorithm for fitting this is described  
32 in [Kha+10].  
33

### 34 29.3.5 Factor analysis with DNN likelihoods

35 The FA model assumes the observed data can be modeled as arising from a linear mapping from a  
36 low-dimensional set of Gaussian factors. One way to relax this assumption is to let the mapping  
37 from  $\mathbf{z}$  to  $\mathbf{x}$  be a nonlinear model, such as a neural network. That is, the likelihood becomes  
38

$$\underline{39} \quad p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|f(\mathbf{w}; \theta), \sigma^2 \mathbf{I}) \quad (29.94)$$

40

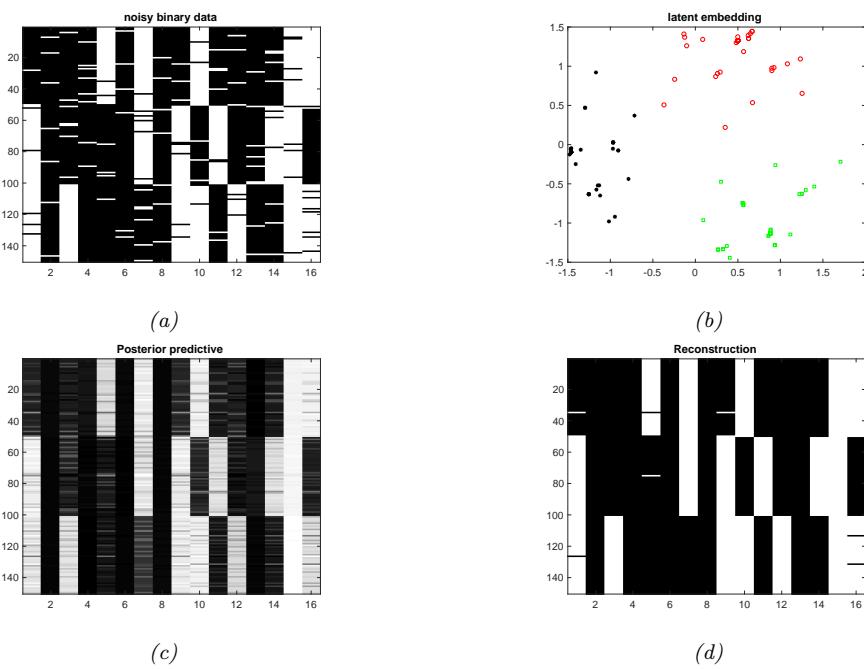


Figure 29.10: (a) 150 synthetic 16 dimensional bit vectors. (b) The 2d embedding learned by binary PCA, fit using variational EM. We have color coded points by the identity of the true “prototype” that generated them. (c) Predicted probability of being on. (d) Thresholded predictions. Generated by [binary\\_fa\\_demo.py](#).

We call this “**DNN factor analysis**”. (We can of course replace the Gaussian likelihood with other distributions, such as categorical.) Unfortunately we can no longer compute the posterior or the MLE exactly, so we need to use approximate methods. In Chapter 22, we discuss variational autoencoders, which fits this model using amortized variational inference. However, it is also possible to fit the same model using other inference methods, such as MCMC (see e.g., [Hof17]).

### 29.3.6 Factor analysis with GP likelihoods (GP-LVM)

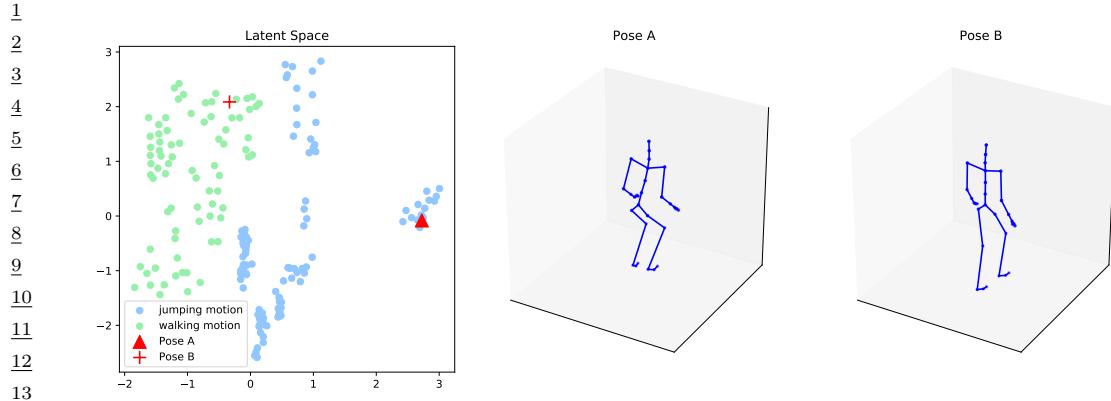
Another way to make a nonlinear version of FA is to replace  $f$  with a Gaussian process (Chapter 18). This is known as a **GP-LVM**, which stands for “Gaussian process latent variable model” [Law05]. (This is closely related to an approach known as **kernel PCA**, discussed in [Mur22, Sec 20.4.6].)

To explain the method in more detail, we start with PPCA (Section 29.3.2). Recall that the PPCA model is as follows:

$$p(\mathbf{z}_i) = \mathcal{N}(\mathbf{z}_i | \mathbf{0}, \mathbf{I}) \quad (29.95)$$

$$p(\mathbf{x}_i | \mathbf{z}_i, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_i | \mathbf{W}\mathbf{z}_i, \sigma^2 \mathbf{I}) \quad (29.96)$$

We can fit this model by maximum likelihood, by integrating out the  $\mathbf{z}_i$  and maximizing  $\mathbf{W}$  (and



*Figure 29.11: Illustration of a 2d embedding of human motion-capture data using a GP-LVM. We show two poses and their corresponding embeddings. Generated by [gplvm\\_mocap.ipynb](#). Used with kind permission of Aditya Ravuri.*

$\sigma^2$ ). The objective is given by

$$p(\mathbf{X}|\mathbf{W}, \sigma^2) = (2\pi)^{-DN/2} |\mathbf{C}|^{-N/2} \exp\left(-\frac{1}{2}\text{tr}(\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X})\right) \quad (29.97)$$

where  $\mathbf{C} = \mathbf{WW}^\top + \sigma^2\mathbf{I}$ . As we showed in Section 29.3.2, the MLE for  $\mathbf{W}$  can be computed in terms of the eigenvectors of  $\mathbf{X}^\top\mathbf{X}$ .

Now we consider the dual problem, whereby we maximize  $\mathbf{Z}$  and integrate out  $\mathbf{W}$ . We will use a prior of the form  $p(\mathbf{W}) = \prod_j \mathcal{N}(\mathbf{w}_j|\mathbf{0}, \mathbf{I})$ . The corresponding likelihood becomes

$$p(\mathbf{X}|\mathbf{Z}, \sigma^2) = \prod_{d=1}^D \mathcal{N}(\mathbf{X}_{:,d}|\mathbf{0}, \mathbf{ZZ}^\top + \sigma^2\mathbf{I}) \quad (29.98)$$

$$= (2\pi)^{-DN/2} |\mathbf{K}_z|^{-D/2} \exp\left(-\frac{1}{2}\text{tr}(\mathbf{K}_z^{-1}\mathbf{XX}^\top)\right) \quad (29.99)$$

where  $\mathbf{K}_z = \mathbf{K} + \sigma^2\mathbf{I}$ , and  $\mathbf{K} = \mathbf{ZZ}^\top$ . The MLE for  $\mathbf{Z}$  can be computed in terms of the eigenvectors of  $\mathbf{K}_z$ , and gives the same results as PPCA (see [Law05] for the details).

The advantage of the dual formulation is that we can use a more general kernel for  $\mathbf{K}$  instead of  $\mathbf{K} = \mathbf{ZZ}^\top$ . The MLE for  $\mathbf{Z}$  is no longer be available via eigenvalue methods, but can be computed using gradient-based optimization.

In Figure 29.11, we illustrate the model (with an ARD kernel) applied to some **motion capture** data, from the CMU mocap database at <http://mocap.cs.cmu.edu/>. Each person has 41 markers, whose motion in 3d is tracked using 12 infrared cameras. Each data point corresponds to a different body pose. When projected to 2d, we see that similar poses are clustered nearby.

47

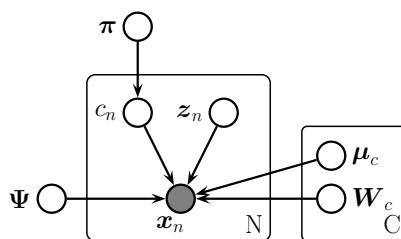


Figure 29.12: Mixture of factor analyzers as a PGM.

## 29.4 Mixture of factor analysers

The factor analysis model (Section 29.3.1) assumes the observed data can be modeled as arising from a linear mapping from a low-dimensional set of Gaussian factors. One way to relax this assumption is to assume the model is only locally linear, so the overall model becomes a (weighted) combination of FA models; this is called a **mixture of factor analysers**. The overall model for the data is a mixture of linear manifolds, which can be used to approximate an overall curved manifold. (Another way to think of this model is a mixture of Gaussians, where each mixture component has a low-rank covariance matrix.)

### 29.4.1 Model definition

More precisely, let latent indicator  $c_n \in \{1, \dots, K\}$ , specifying which subspace (cluster) we should use to generate the data. If  $c_n = k$ , we sample  $\mathbf{z}_n$  from a Gaussian prior and pass it through the  $\mathbf{W}_k$  matrix and add noise, where  $\mathbf{W}_k$  maps from the  $L$ -dimensional subspace to the  $D$ -dimensional visible space.<sup>2</sup> More precisely, the model is as follows:

$$p(\mathbf{x}_n | \mathbf{z}_n, c_n = k, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k + \mathbf{W}_k \mathbf{z}_n, \boldsymbol{\Psi}_k) \quad (29.100)$$

$$p(\mathbf{z}_n | \boldsymbol{\theta}) = \mathcal{N}(\mathbf{z}_n | \mathbf{0}, \mathbf{I}) \quad (29.101)$$

$$p(c_n | \boldsymbol{\theta}) = \text{Cat}(c_n | \boldsymbol{\pi}) \quad (29.102)$$

This is called a **mixture of factor analysers** (MFA) [GH96b]. The corresponding distribution in the visible space is given by

$$p(\mathbf{x} | \boldsymbol{\theta}) = \sum_k p(c = k) \int p(\mathbf{z} | c) p(\mathbf{x} | \mathbf{z}, c) d\mathbf{z} = \sum_k \pi_k \int \mathcal{N}(\mathbf{z} | \boldsymbol{\mu}_k, \mathbf{I}) \mathcal{N}(\mathbf{x} | \mathbf{W}_k \mathbf{z}, \boldsymbol{\Psi}_k) d\mathbf{z} \quad (29.103)$$

In the special case that  $\boldsymbol{\Psi}_k = \sigma^2 \mathbf{I}$ , we get a mixture of PPCA models (although it is difficult to ensure orthogonality of the  $\mathbf{W}_k$  in this case). See Figure 29.13 for an example of the method applied to some 2d data.

<sup>2</sup> If we allow  $\mathbf{z}_n$  to depend on  $c_n$ , we can let each subspace have a different dimensionality, as suggested in [KS15].

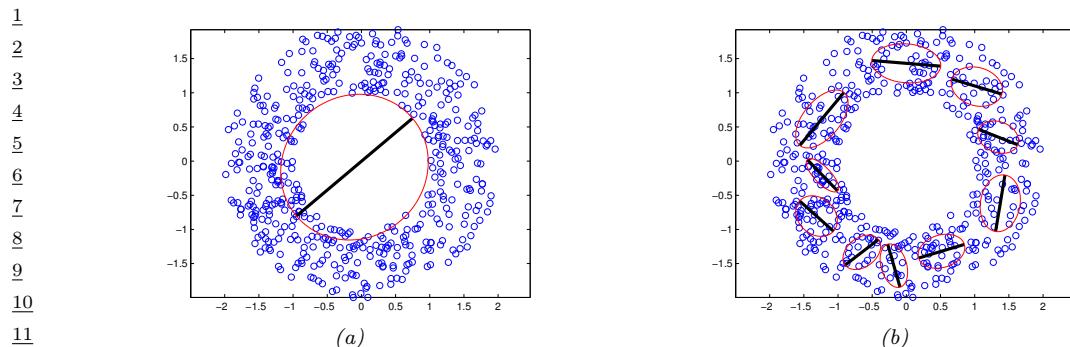


Figure 29.13: Mixture of PPCA models fit to a 2d dataset, using  $L = 1$  latent dimensions. (a)  $K = 1$  mixture components. (b)  $K = 10$  mixture components. Generated by `mixPpcaDemo.py`.

<sup>18</sup> We can think of this as a low-rank version of a mixture of Gaussians. In particular, this model  
<sup>19</sup> needs  $O(KLD)$  parameters instead of the  $O(KD^2)$  parameters needed for a mixture of full covariance  
<sup>20</sup> Gaussians. This can reduce overfitting.

### 29.4.2 Model fitting

There are several ways to fit an MFA model, some of which we discuss below.

### 27 29.4.2.1 Model fitting using EM

<sup>29</sup> We can fit this model using EM (see [GH96b] for the derivation). In the E step, we compute the  
<sup>30</sup> posterior responsibility of cluster  $c$  for data point  $n$  using

$$r_{nc} \triangleq p(c_n = c | \mathbf{x}_n, \boldsymbol{\theta}) \propto \pi_c \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_c, \mathbf{W}_c \mathbf{W}_c^\top + \boldsymbol{\Psi}) \quad (29.104)$$

<sup>34</sup> The conditional posterior for  $z_n$  is given by

$$p(\mathbf{z}_n | \mathbf{x}_n, c_n = c, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{z}_n | \mathbf{m}_{nc}, \boldsymbol{\Sigma}_{nc}) \quad (29.105)$$

$$\boldsymbol{\Sigma}_{\text{reg}} \triangleq (\mathbf{I} + \mathbf{W}^T \boldsymbol{\Psi}^{-1} \mathbf{W})^{-1} \quad (29,106)$$

$$m_{\cdot \cdot} \triangleq \Sigma_{\cdot \cdot}(\mathbf{W}^T \boldsymbol{\Psi}^{-1}(x_{\cdot \cdot} - \mu_{\cdot \cdot})) \quad (29, 107)$$

For the M step, we define  $\tilde{\mathbf{W}}_c \equiv (\mathbf{W}_c, \boldsymbol{\mu}_c)$ , and  $\tilde{\mathbf{z}} \equiv (\mathbf{z}, 1)$ . Also, define

$$\mathbf{b}_{nc} \triangleq \mathbb{E}[\tilde{\mathbf{z}} | \mathbf{x}_n, c_n = c] = [\mathbf{m}_{nc}; 1] \quad (29.108)$$

$$\Omega_{nc} \triangleq \mathbb{E} \left[ \tilde{\mathbf{z}} \tilde{\mathbf{z}}^\top | \mathbf{x}_n, c_n = c \right] = \begin{pmatrix} \mathbb{E} [\mathbf{z} \mathbf{z}^\top | \mathbf{x}_n, c_n = c] & \mathbb{E} [\mathbf{z} | \mathbf{x}_n, c_n = c] \\ \mathbb{E} [\mathbf{z} | \mathbf{x}_n, c_n = c]^\top & 1 \end{pmatrix} \quad (29.109)$$

1 Then the M step is as follows:  
 2

$$\hat{\mathbf{W}}_c = \left[ \sum_n r_{nc} \mathbf{x}_n \mathbf{b}_c^\top \right] \left[ \sum_n r_{nc} \boldsymbol{\Omega}_{nc} \right]^{-1} \quad (29.110)$$

$$\hat{\Psi} = \frac{1}{N} \text{diag} \left\{ \sum_{nc} r_{nc} (\mathbf{x}_n - \hat{\mathbf{W}}_c \mathbf{b}_{nc}) \mathbf{x}_n^\top \right\} \quad (29.111)$$

$$\hat{\pi}_c = \frac{1}{N} \sum_n r_{nc} \quad (29.112)$$

### 29.4.2.2 Model fitting using SGD

We can also fit mixture models using SGD, as shown in [RW18]. This idea can be combined with an inference network (see Section 10.3.7) to efficiently approximate the posterior over the latent variables. [Zon+18] use this approach to jointly learn a GMM applied to a deep autoencoder to provide a nonlinear extension of MFA; they show good results on anomaly detection.s

### 29.4.2.3 Model selection

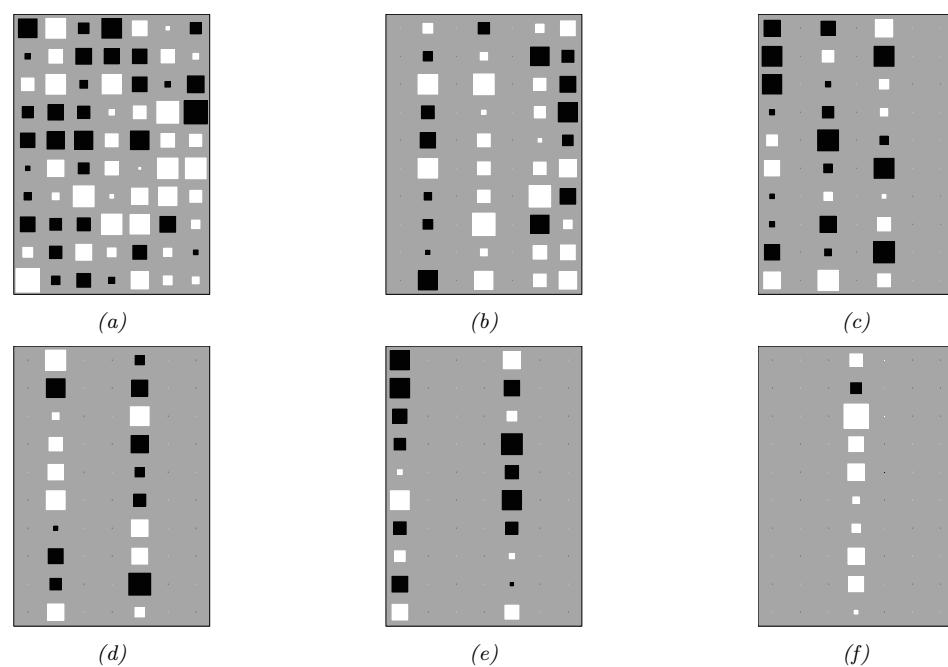
To choose the number of mixture components  $C$ , and the number of latent dimensions  $L$ , we can use discrete search combined with objectives such as the marginal likelihood or validation likelihood. However, we can also use numerical optimization methods to optimize  $L$ , which can be faster. We initially assume that  $C$  is known. To estimate  $L$ , we set the model to its maximal size, and then use a technique called automatic relevance determination or ARD to automatically prune out irrelevant weights (see Section 15.2.7). This can be implemented using variational Bayes EM (Section 10.2.5); for details, see [Bis99; GB00].

Figure 29.14 illustrates this approach applied to a mixture of FA models fit to a small synthetic dataset. The figures visualize the weight matrices for each cluster, using **Hinton diagrams**, where the size of the square is proportional to the value of the entry in the matrix. We see that many of them are sparse. Figure 29.15 shows that the degree of sparsity depends on the amount of training data, in accord with the Bayesian Occam's razor. In particular, when the sample size is small, the method automatically prefers simpler models, but as the sample size gets sufficiently large, the method converges on the “correct” solution, which is one with 6 subspaces of dimensionality 1, 2, 2, 3, 4 and 7.

Although the ARD method can estimate the number of latent dimensions  $L$ , it still needs to perform discrete search over the number of mixture components  $C$ . This is done using “birth” and “death” moves [GB00]. An alternative approach is to perform stochastic sampling in the space of models. Traditional approaches, such as [LW04], are based on reversible jump MCMC, and also use birth and death moves. However, this can be slow and difficult to implement. More recent approaches use non-parametric priors, combined with Gibbs sampling, see e.g., [PC09].

### 29.4.3 MixFA for image generation

In this section, we use the MFA model as a generative model for images, following [RW18]. This is equivalent to using a mixture of Gaussians, where each mixture component has a low-rank covariance



<sup>24</sup> Figure 29.14: Illustration of estimating the effective dimensionalities in a mixture of factor analysers using  
<sup>25</sup> variational Bayes EM with an ARD prior. Black are negative values, white are positive, gray is 0. The blank  
<sup>26</sup> columns have been forced to 0 via the ARD mechanism, reducing the effective dimensionality. The data was  
<sup>27</sup> generated from 6 clusters with intrinsic dimensionalities of 7, 4, 3, 2, 2, 1, which the method has successfully  
<sup>28</sup> estimated. From Figure 4.4 of [Bea03]. Used with kind permission of Matt Beal.

number of points per cluster	intrinsic dimensionalities					
	1	7	4	3	2	2
8		2			1	
8	1			2		
16	1		4			2
32	1	6	3	3	2	2
64	1	7	4	3	2	2
128	1	7	4	3	2	2

*Figure 29.15: We show the estimated number of clusters, and their estimated dimensionalities, as a function of sample size. The ARD algorithm found two different solutions when  $N = 8$ . Note that more clusters, with larger effective dimensionalities, are discovered as the sample sizes increases. From Table 4.1 of [Bea03]. Used with kind permission of Matt Beal.*



Figure 29.16: Random samples from the MixFA model fit to CelebA. Generated by `mix_PPDA_celeba.ipynb`. Adapted from Figure 4 of [RW18]. Used with kind permission of Yair Weiss

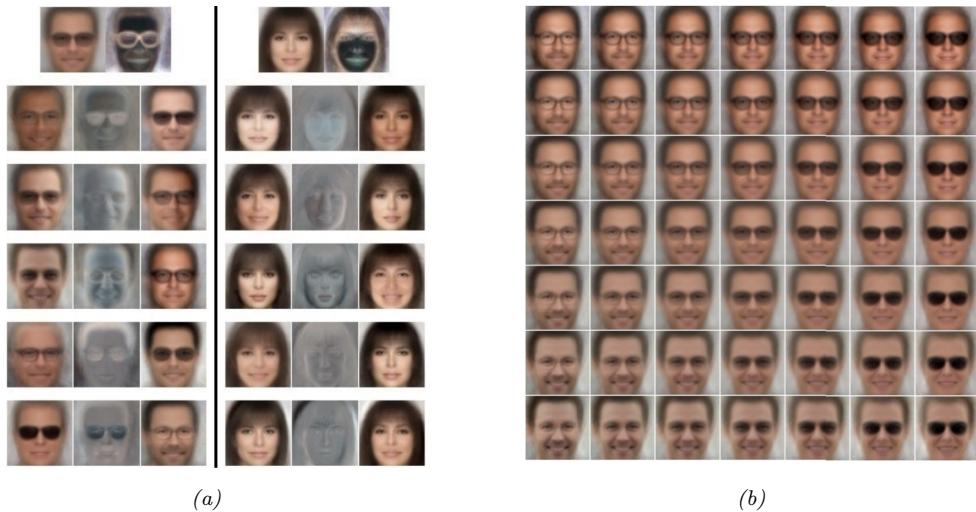


Figure 29.17: (a) Visualization of the parameters learned by the MFA model. The top row shows the mean  $\mu_k$  and noise variance  $\Psi_k$ , reshaped from 12,288-dimensional vectors to  $64 \times 64 \times 3$  images, for two mixture components  $k$ . The next 5 rows show the first 5 (of 10) basis functions (columns of  $\mathbf{W}_k$ ) as images. On row  $i$ , left column, we show  $\mu_k - \mathbf{W}_k[:, i]$ ; in the middle, we show  $0.5 + \mathbf{W}_k[:, i]$ , and on the right we show  $\mu_k + \mathbf{W}_k[:, i]$ . (b) Images generated by computing  $\mu_k + z_1 \mathbf{W}_k[:, i] + z_2 \mathbf{W}_k[:, j]$ , for some component  $k$  and dimensions  $i, j$ , where  $(z_1, z_2)$  are drawn from the grid  $[-1 : 1, -1 : 1]$ , so the central image is just  $\mu_k$ . From Figure 6 of [RW18]. Used with kind permission of Yair Weiss

matrix. Surprisingly, the results are competitive with deep generative models such as those in Part IV, despite the fact that no neural networks are used in the model.

In [RW18], they fit the MFA model to the CelebA dataset, which is a dataset of faces of celebrities (movie stars). They use  $K = 300$  components, each of latent dimension  $L = 10$ ; the observed data has dimension  $D = 64 \times 64 \times 3 = 12,288$ . They fit the model using SGD, using the methods from Section 29.3.1.3 to efficiently compute the log likelihood, despite the high dimensionality. The  $\mu_k$  parameters are initialized using K-means clustering, and the  $\mathbf{W}_k$  parameters are initialized using factor analysis for each component separately. Then the model is fine-tuned end-to-end.



14 *Figure 29.18: Samples from the 100 CelebA images with lowest likelihood under the MFA model. Generated*  
 15 *by mix\_PPcA\_celeba.ipynb. Adapted from Figure 7a of [RW18]. Used with kind permission of Yair Weiss*



30 *Figure 29.19: Illustration of image imputation using an MFA. Left column shows 4 original images. Subsequent*  
 31 *pairs of columns show an occluded input, and a predicted output. Generated by mix\_PPcA\_celeba.ipynb.*  
 32 *Adapted from Figure 7b of [RW18]. Used with kind permission of Yair Weiss*

33  
34  
35  
36 Figure 29.16 shows some images generated from the fitted model. The results are surprisingly good  
 37 for such a simple locally linear model. The reason the method works is similar to the discussion  
 38 in Section 29.2.4.1: essentially the  $\mathbf{W}_k$  matrix learns a set of  $L$ -dimensional basis functions for the  
 39 subset of face images that get mapped to cluster  $k$ . See Figure 29.17 for an illustration.

40 There are several advantages to this model compared to VAEs and GANs. First, [RW18], showed  
 41 that this MixFA model captures more of the modes of the data distribution than more sophisticated  
 42 generative models, such as VAEs (Section 22.2) and GANs (Chapter 27). Second, we can compute  
 43 the exact likelihood  $p(\mathbf{x})$ , so we can compute outliers or unusual images. This is illustrated in  
 44 Figure 29.18.

45 Third, we can perform image imputation from partially observed images given arbitrary missingness  
 46 patterns. To see this, let us partition  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ , where  $\mathbf{x}_1$  (of size  $D_1$ ) is observed and  $\mathbf{x}_2$  (of size  
 47

2  $D_2 = D - D_1$ ) is missing. We can compute the most probable cluster using

3

$$\underline{4} \quad k^* = \operatorname{argmax}_{k=1}^K p(c=k) p(\mathbf{x}_1 | c=k) \quad (29.113)$$

5

6 where

7

$$\underline{8} \quad \log p(\mathbf{x}_1 | \boldsymbol{\mu}_k, \mathbf{C}_k) = -\frac{1}{2} \left[ D_1 \log(2\pi) + \log \det(\mathbf{C}_{k,11}) + \tilde{\mathbf{x}}_1^\top \mathbf{C}_{k,11}^{-1} \tilde{\mathbf{x}}_1 \right] \quad (29.114)$$

9

10 where  $\mathbf{C}_{k,11}$  is the top left  $D_1 \times D_1$  block of  $\mathbf{W}_k \mathbf{W}_k^\top + \boldsymbol{\Psi}_k$ , and  $\tilde{\mathbf{x}}_1 = \mathbf{x}_1 - \boldsymbol{\mu}_k[1:D_1]$ . Once we know  
11 which discrete mixture component to use, we can compute the Gaussian posterior  $p(\mathbf{z}|\mathbf{x}_1, k^*)$  using  
12 Equation (29.36). Let  $\hat{\mathbf{z}} = \mathbb{E}[\mathbf{z}|\mathbf{x}_1, k^*]$ . Given this, we can compute the predicted output for the full  
13 image:

14

$$\underline{15} \quad \hat{\mathbf{x}} = \mathbf{W}_{k^*} \hat{\mathbf{z}} + \boldsymbol{\mu}_{k^*} \quad (29.115)$$

16

17 We then use the estimate  $\mathbf{x}' = [\mathbf{x}_1, \hat{\mathbf{x}}_2]$ , so the observed pixels are not changed. This is an example of  
18 **image imputation**, and is illustrated in Figure 29.19. Note that we can condition on an arbitrary  
19 subset of pixels, and fill in the rest, whereas some other models (e.g., autoregressive models) can only  
20 predict the bottom right given the the top left (since they assume a generative model which works in  
21 raster-scan order).

## 29.5 LVMs with non-Gaussian priors

25 In this section, we discuss (linear) latent factor models with non-Gaussian priors. See Table 29.1 for  
26 a summary of the models we will discuss.

### 29.5.1 Non-negative matrix factorization (NMF)

30 Suppose that we use a gamma distribution for the latents:  $p(\mathbf{z}) = \prod_k \text{Ga}(z_{1,k} | \alpha_k, \beta_k)$ . This results  
31 in a sparse, non-negative hidden representation, which can help interpretability. This is particularly  
32 useful when the data is also sparse and non-negative, such as word counts. In this case, it makes  
33 sense to use a Poisson likelihood:  $p(\mathbf{x}|\mathbf{z}) = \prod_{d=1}^D \text{Poi}(x_d | \mathbf{w}_d^\top \mathbf{z})$ . The overall model has the form

34

$$\underline{35} \quad p(\mathbf{z}, \mathbf{x}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \left[ \prod_k \text{Ga}(z_{1,k} | \alpha_k, \beta_k) \right] \left[ \prod_{d=1}^D \text{Poi}(x_d | \mathbf{w}_d^\top \mathbf{z}) \right] \quad (29.116)$$

36

37 The resulting model is called the **GaP** (Gamma-Poisson) model [Can04]. See Figure 29.20a for the  
38 graphical model.

40 The parameters  $\alpha_k$  and  $\beta_k$  control the sparsity of the latent representation  $\mathbf{z}_n$ . If we set  $\alpha_k = \beta_k = 0$ ,  
41 and compute the MLE for  $\mathbf{W}$ , we recover **non-negative matrix factorization (NMF)** [PT94;  
42 LS99; LS01], as shown in [BJ06].

43 Figure 29.21 illustrates the result of applying NMF to a dataset of image patches of faces, where  
44 the data correspond to non-negative pixel intensities. We see that the learned basis functions are  
45 small localized **parts** of faces. Also, the coefficient vector  $\mathbf{z}$  is sparse and positive. For PCA, the  
46 coefficient vector has negative values, and the resulting basis functions are global, not local. For  
47

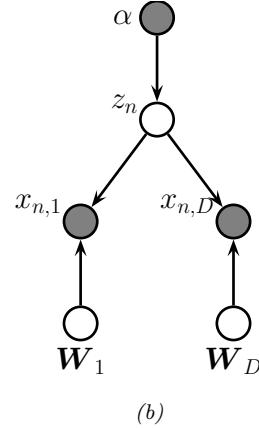
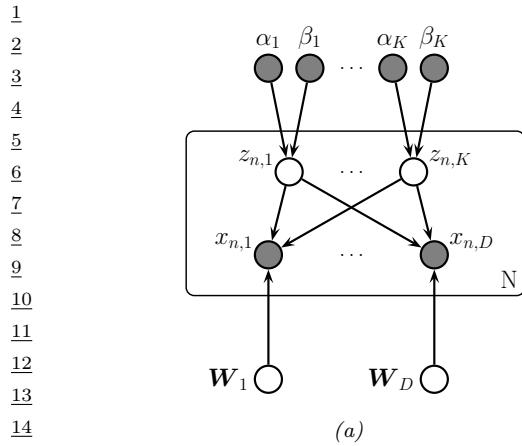


Figure 29.20: (a) Gaussian-Poisson (GAP) model as a PGM-D. Here  $z_{nk} \in \mathbb{R}^+$  and  $x_{n,d} \in \mathbb{Z}_{\geq 0}$ . (b) Simplex FA model as a PGM-D. Here  $z_n \in \mathbb{S}_K$  and  $x_{n,d} \in \{1, \dots, V\}$ .

vector quantization (i.e., GMM model),  $\mathbf{z}$  is a one-hot vector, with a single mixture component turned on; the resulting weight vectors correspond to entire image prototypes. The reconstruction quality is similar in each case, but the nature of the learned latent representation is quite different.

### 29.5.2 Multinomial PCA

Suppose we use a Dirichlet prior for the latents,  $p(\mathbf{z}) = \text{Dir}(\mathbf{z}|\boldsymbol{\alpha})$ , so  $\mathbf{z} \in \mathbb{S}_K$ , which is the  $K$ -dimensional probability simplex. As in Section 29.5.1, the vector  $\mathbf{z}$  will be sparse and non-negative, but in addition it will satisfy the constraint  $\sum_{k=1}^K z_k = 1$ , so the components are not independent. Now suppose our data is categorical,  $x_d \in \{1, \dots, V\}$ , so our likelihood has the form  $p(\mathbf{x}|\mathbf{z}) = \prod_d \text{Cat}(x_d|\mathbf{W}_d \mathbf{z})$ . The overall model is therefore

$$p(\mathbf{z}, \mathbf{x}) = \text{Dir}(\mathbf{z}|\boldsymbol{\alpha}) \prod_{d=1}^D \text{Cat}(x_d|\mathbf{W}_d \mathbf{z}) \quad (29.117)$$

See Figure 29.20b for the PGM-D. This model (or small variants of it) has multiple names: **user rating profile model** [Mar03], **admixture model** [PSD00], **mixed membership model** [EFL04], **multinomial PCA (mPCA)** [BJ06], or **simplex factor analysis (sFA)** [BD11].

#### 29.5.2.1 Example: roll call data

Let us consider the example from [BJ06], who applied this model to analyze some **roll call** data from the US Senate in 2003. Specifically, the data has the form  $x_{n,d} \in \{+1, -1, 0\}$  for  $n = 1 : 100$  and  $d = 1 : 459$ , where  $x_{nd}$  is the vote of the  $n$ 'th senator on the  $d$ 'th bill, where +1 means in favor, -1 means against, and 0 means not voting. In addition, we have the overall outcome, which we denote by  $x_{101,d} \in \{+1, -1\}$ , where +1 means the bill was passed, and -1 means it was rejected.

We fit the mPCA model to this data using 5 latent factors using variational EM. Figure 29.22 plots  $\mathbb{E}[z_{nk}|\mathbf{x}_n] \in [0, 1]$ , which is the degree to which senator  $n$  belongs to latent component or “bloc”

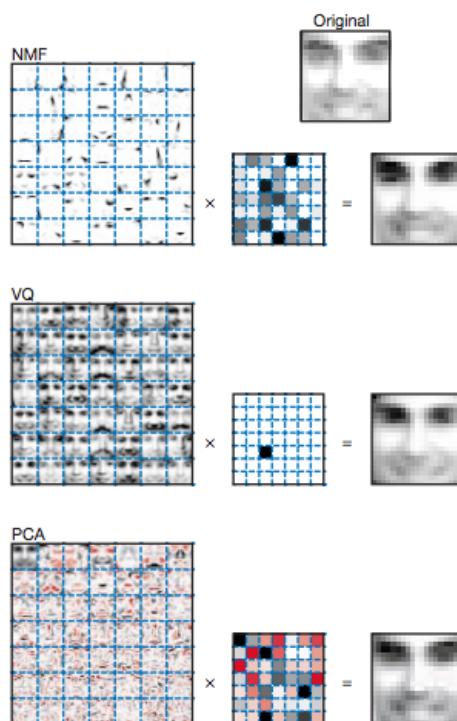


Figure 29.21: Illustrating the difference between Non-negative Matrix Factorization (NMF), Vector Quantization (VQ), and Principal Components Analysis (PCA). Left column: Filters (columns of  $\mathbf{W}$ ) learned from a set of 2429 faces images, each of size  $19 \times 19$ . There are 49 basis functions in total, shown in a  $7 \times 7$  montage; each filter is reshaped to a  $19 \times 19$  image for display purposes. (For PCA, negative weights are red, positive weights are black.) Middle column: The 49 latent factors  $\mathbf{z}$  when the model is applied to the original face image shown at the top. Right column: reconstructed face image. From Figure 1 of [LS99].

k. We see that component 5 is the Democratic majority, and block 2 is the Republican majority. See [BJ06] for further details.

### 29.5.2.2 Advantage of Dirichlet prior over Gaussian prior

The main advantage of using a Dirichlet prior compared to a Gaussian prior is that the latent factors are more interpretable. To see this, note that the mean parameters for  $d$ 'th output distribution have the form  $\boldsymbol{\mu}_{nd} = \mathbf{W}^d \mathbf{z}_n$ , and hence

$$p(x_{nd} = v | \mathbf{z}_n) = \sum_k z_{nk} w_{kv}^d \quad (29.118)$$

Thus the latent variables define the mean parameters. By contrast, the CatPCA model in Section 29.3.4.2 uses a Gaussian prior, so  $\mathbf{W}^d \mathbf{z}_n$  can be negative; consequently it must pass this vector

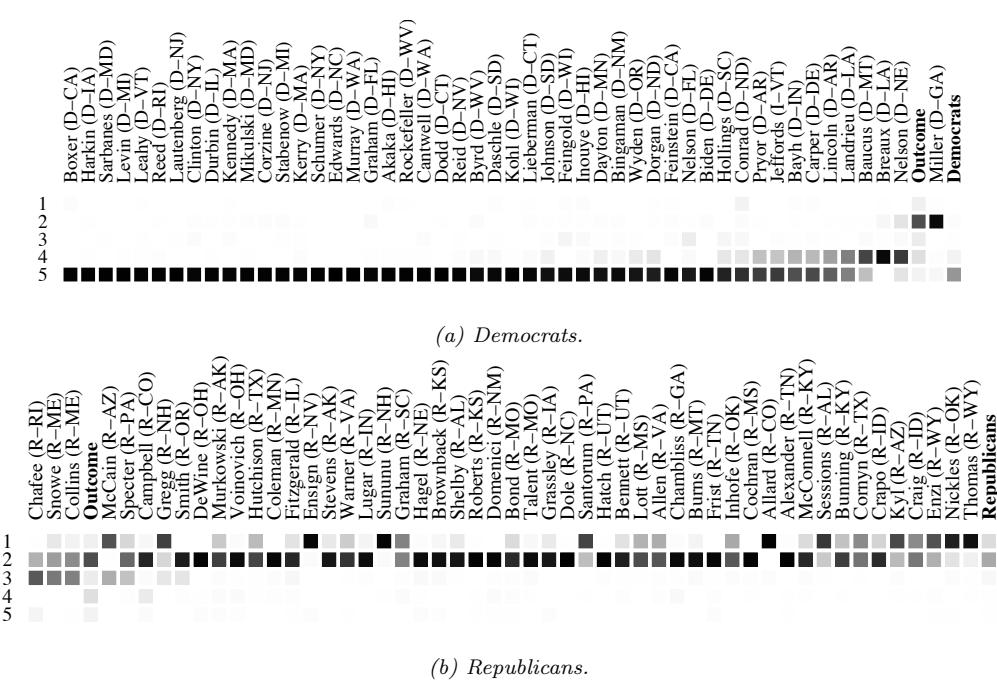


Figure 29.22: The simplex factor analysis model applied to some roll call data from the US Senate collected in 2003. The senators have been sorted from left to right using the binary PCA method of [Lee06]. See text for details. From Figures 8–9 of [BJ06]. Used with kind permission of Wray Buntine.

through a softmax, to convert from natural parameters to mean parameters; this makes  $\mathbf{z}_n$  harder to interpret.

### 29.5.2.3 Connection to mixture models

If  $\mathbf{z}_n$  were a one-hot vector, the mPCA model would be equivalent to selecting a single column from  $\mathbf{W}_d$  corresponding to the discrete hidden state:

$$p(x_{nd} = v | z_n) = \sum_{k=1}^K \mathbb{I}(z_n = k) w_{kv}^d \quad (29.119)$$

This is equivalent to a finite mixture of categorical distributions (c.f., Section 29.2.2), and corresponds to the assumption that  $\mathbf{x}$  is generated by a single cluster. However, the mPCA model does not require that  $\mathbf{z}_n$  be one-hot, and instead allows  $\mathbf{z}_n$  to partially belong to multiple clusters. For this reason, this model is also known as an **admixture mixture** or **mixed membership model** [EFL04].

### 29.5.3 Latent Dirichlet Allocation (LDA)

In this section, we discuss a simple extension of the multinomial PCA model of Section 29.5.2 in which  $\mathbf{x}$  is a variable-length sequence of tokens, and the weights are tied across “time”. Thus the

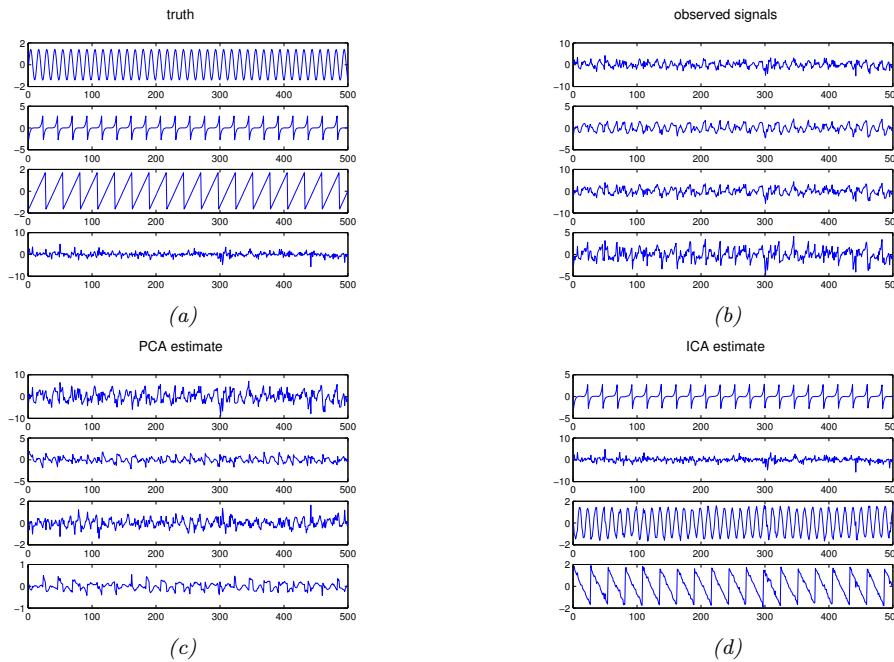


Figure 29.23: Illustration of ICA applied to 500 iid samples of a 4d source signal. (a) Latent signals. (b) Observations. (c) PCA estimate. (d) ICA estimate. This matches the true sources, up to permutation of the dimension indices. Generated by `ica_demo.py`.

model can be defined as follows:

$$p(\mathbf{z}, \mathbf{x}) = \text{Dir}(\mathbf{z}|\boldsymbol{\alpha}) \prod_{t=1}^T \text{Cat}(x_t|\mathbf{W}\mathbf{z}) \quad (29.120)$$

This is known as **latent Dirichlet allocation** or **LDA** [BNJ03a; Ble12; BGHM17]. Due to lack of space, we discuss this model in more detail in the supplementary material.

## 29.6 Independent components analysis (ICA)

Consider the following situation. You are in a crowded room and many people are speaking. Your ears essentially act as two microphones, which are listening to a linear combination of the different speech signals in the room. Your goal is to deconvolve the mixed signals into their constituent parts. This is known as the **cocktail party problem**, or the **blind source separation (BSS)** problem, where “blind” means we know “nothing” about the source of the signals. Besides the obvious applications to acoustic signal processing, this problem also arises when analysing EEG and MEG signals, financial data, and any other dataset (not necessarily temporal) where latent sources or factors get mixed together in a linear way. See Figure 29.23 for an example.

1 **29.6.1 Noiseless ICA model**

3 We can formalize the problem as follows. Let  $\mathbf{x}_n \in \mathbb{R}^D$  be the vector of observed responses, at “time”  
4  $n$ , where  $D$  is the number of sensors / microphones. Let  $\mathbf{z}_n \in \mathbb{R}^D$  be the hidden vector of source  
5 signals at time  $n$ , of the same dimensionality as the observed signal. We assume that  
6

7 
$$\mathbf{x}_n = \mathbf{A}\mathbf{z}_n \tag{29.121}$$
  
8

9 where  $\mathbf{A}$  is an invertible  $D \times D$  matrix known as the **mixing matrix** or the **generative weights**.  
10 The prior has the form  $p(\mathbf{z}_n) = \prod_{j=1}^D p_j(z_j)$ . Typically we assume this is a sparse prior, so only a  
11 subset of the signals are active at any one time (see Section 29.6.2 for further discussion of priors for  
12 this model). This model is called **independent components analysis** or **ICA**, since we assume  
13 that each observation  $\mathbf{x}_n$  is a linear combination of independent components represented by sources  
14  $\mathbf{z}_n$ , i.e,

15 
$$x_{nj} = \sum_i A_{ij} z_{nj} \tag{29.122}$$
  
16

18 Our goal is to infer the source signals,  $p(\mathbf{z}_n | \mathbf{x}_n, \mathbf{A})$ . Since the model is noiseless, we have  
19

20 
$$p(\mathbf{z}_n | \mathbf{x}_n, \mathbf{A}) = \delta(\mathbf{z}_n - \mathbf{B}\mathbf{x}_n) \tag{29.123}$$
  
21

22 where  $\mathbf{B} = \mathbf{A}^{-1}$  are the **recognition weights**. (We discuss how to estimate these weights in  
23 Section 29.6.3.)  
24

25 **29.6.2 The need for non-Gaussian priors**

26 Since  $\mathbf{x} = \mathbf{A}\mathbf{z}$ , we have  $\mathbb{E}[\mathbf{x}] = \mathbf{A}\mathbb{E}[\mathbf{z}]$  and  $\text{Cov}[\mathbf{x}] = \text{Cov}[\mathbf{A}\mathbf{z}] = \mathbf{A}\text{Cov}[\mathbf{z}]\mathbf{A}^\top$ . Without loss of  
27 generality, we can assume  $\mathbb{E}[\mathbf{z}] = \mathbf{0}$ , since we can always center the data. Similarly, we can assume  
28  $\text{Cov}[\mathbf{z}] = \mathbf{I}$ , since  $\mathbf{A}\mathbf{A}^\top$  can capture any correlation in  $\mathbf{x}$ . Thus  $\mathbf{z}$  is a set of  $D$  unit variance,  
29 uncorrelated variables, as in factor analysis (Section 29.3.1).  
30

31 However, this is not sufficient to uniquely identify  $\mathbf{A}$  and hence  $\mathbf{z}$ , as we explained in Section 29.3.1.4.  
32 So we need to go beyond an uncorrelated prior and enforce an independent, and non-Gaussian, prior.

33 To illustrate this, suppose we have two independent sources with uniform distributions, as shown  
34 in Figure 29.24(a). Now suppose we have the following mixing matrix  
35

36 
$$\mathbf{A} = \begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \tag{29.124}$$
  
37

38 Then we observe the data shown in Figure 29.24(b) (assuming no noise). The full-rank PCA model  
39 (where  $K = D$ ) is equivalent to ICA, except it uses a factored Gaussian prior for  $\mathbf{z}$ . The result of  
40 using PCA is shown in Figure 29.24(c). This corresponds to a **whitening** or **sphering** of the data,  
41 in which  $\text{Cov}[\mathbf{z}] = \mathbf{I}$ . To uniquely recover the sources, we need to perform an additional rotation.  
42 The trouble is, there is no information in the symmetric Gaussian posterior to tell us which angle to  
43 rotate by. In a sense, PCA solves “half” of the problem, since it identifies the linear subspace; all  
44 that ICA has to do is then to identify the appropriate rotation. To do this, ICA uses an independent,  
45 but non-Gaussian, prior. The result is shown in Figure 29.24(d). This shows that ICA can recover  
46 the source variables, up to a permutation of the indices and possible sign change.  
47

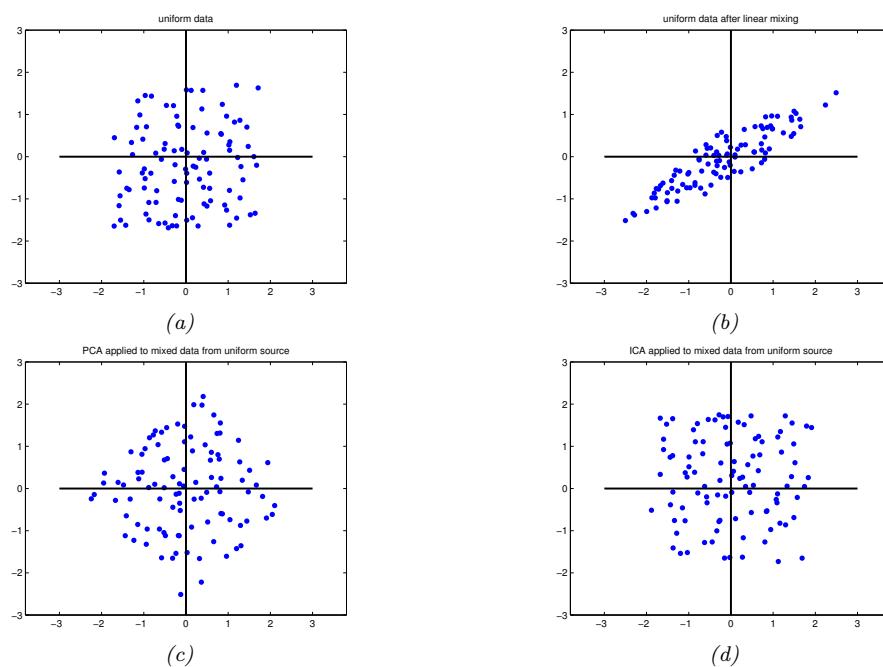


Figure 29.24: Illustration of ICA and PCA applied to 100 iid samples of a 2d source signal with a uniform distribution. (a) Latent signals. (b) Observations. (c) PCA estimate. (d) ICA estimate. Generated by [ica\\_demo\\_uniform.py](#).

We typically use a prior which is a super-Gaussian distribution, meaning it has heavy tails; this helps with identifiability. One option is to use a Laplace prior. For mean zero and variance 1, this has a log pdf given by

$$\log p(z) = -\sqrt{2}|z| - \log(\sqrt{2}) \quad (29.125)$$

However, since the Laplace prior is not differentiable at the origin, in ICA it is more common to use the logistic distribution, discussed in Section 15.4.1. The corresponding log pdf, for the case where the mean is zero and the variance is 1, is given by the following:

$$\log p(z) = -2 \log \cosh\left(\frac{\pi}{2\sqrt{3}}z\right) - \log \frac{4\sqrt{3}}{\pi} \quad (29.126)$$

### 29.6.3 Maximum likelihood estimation

Since  $\mathbf{x} = \mathbf{A}\mathbf{z}$ , from the change of variables formula, the density of the observed data is given by

$$p_x(\mathbf{x}) = p_z(\mathbf{z}) |\det(\mathbf{A}^{-1})| = p_z(\mathbf{B}\mathbf{x}) |\det(\mathbf{B})| \quad (29.127)$$

1 where  $\mathbf{B} = \mathbf{A}^{-1}$ . To maximize this, we can simplify the problem as follows. Let  $\Sigma = \text{Cov}[\mathbf{x}]$ , and  
2 define some nonsingular matrix  $\mathbf{V}$  such that  $\mathbf{B} = \mathbf{A}^{-1} = \mathbf{V}\Sigma^{-\frac{1}{2}}$ . Then  
3

4

$$\underline{5} \quad \mathbf{z} = \mathbf{B}\mathbf{x} = \mathbf{V}\Sigma^{-\frac{1}{2}}\mathbf{x} \quad (29.128)$$

6

7 Since we assumed  $\text{Cov}[\mathbf{z}] = \mathbf{I}$ , we have  
8

9

$$\underline{10} \quad \text{Cov}[\mathbf{z}] = \mathbf{V}\Sigma^{-\frac{1}{2}}\Sigma\Sigma^{\frac{1}{2}}\mathbf{V}^T = \mathbf{V}\mathbf{V}^T = \mathbf{I} \quad (29.129)$$

11

11 Hence  $\mathbf{V}$  is orthogonal. So if we sphere the data, by computing  $\tilde{\mathbf{x}} = \Sigma^{-1/2}\mathbf{x}$ , we will have  $\mathbf{B} = \mathbf{V}$ .  
12 So now our goal simplifies to estimating an orthogonal  $\mathbf{V}$  from the whitened data.<sup>3</sup>

13 With this new notation, we can write the likelihood as  
14

15

$$\underline{16} \quad p_x(\mathbf{x}) = p_z(\mathbf{V}\mathbf{x})|\det(\mathbf{V})| \quad (29.130)$$

17

18 Thus the average negative log likelihood is given by  
19

20

$$\underline{21} \quad \text{NLL}(\mathbf{V}) = -\frac{1}{N} \log p(\mathbf{X}|\mathbf{V}) = -\log |\det(\mathbf{V})| - \frac{1}{N} \sum_{j=1}^L \sum_{n=1}^N \log p_j(\mathbf{v}_j^T \mathbf{x}_n) \quad (29.131)$$

22

23 where  $\mathbf{v}_j$  is the  $j$ 'th row of  $\mathbf{V}$ . Since we are constraining  $\mathbf{V}$  to be orthogonal, the  $\log |\det(\mathbf{V})|$  term  
24 is a constant, so we can drop it. We can also replace the sum over  $n$  with an expectation wrt the  
25 empirical distribution to get the following objective  
26

27

$$\underline{28} \quad \text{NLL}(\mathbf{V}) = \sum_j \mathbb{E}[G_j(z_j)] \quad (29.132)$$

29

30 where  $z_j = \mathbf{v}_j^T \mathbf{x}$  and  $G_j(z) \triangleq -\log p_j(z)$ . We want to minimize this (nonconvex) objective subject to  
31 the constraint that  $\mathbf{V}$  is an orthonormal matrix.  
32

33 It is straightforward to derive a (projected) gradient descent algorithm to fit this model; however,  
34 it is rather slow. One can also derive a faster algorithm that follows the natural gradient; see e.g.,  
35 [Mac03, ch 34] for details. However, the most popular method is to use an approximate Newton  
36 method, known as **fast ICA** [HO00]. This was used to produce Figure 29.23.

37

### 38 29.6.4 Alternatives to MLE

39

40 In this section, we discuss various alternatives estimators for ICA that have been proposed over the  
41 years. We will show that they are equivalent to MLE. However, they bring interesting perspectives  
42 to the problem.  
43

44 45 46 3. Traditionally in the ICA literature the stated goal is to estimate the orthogonal matrix  $\mathbf{W}$ , but this notation  
conflicts with our use of  $\mathbf{W}$  as generative weights in the factor analysis model of Section 29.3.1. So we use the letter  $\mathbf{V}$  instead.

47

1 **29.6.4.1 Maximizing non-Gaussianity**

3 An early approach to ICA was to find a matrix  $\mathbf{V}$  such that the distribution  $\mathbf{z} = \mathbf{V}\mathbf{x}$  is as far from  
4 Gaussian as possible. (There is a related approach in statistics called **projection pursuit** [FT74].)  
5 One measure of non-Gaussianity is kurtosis, but this can be sensitive to outliers. Another measure is  
6 the **negentropy**, defined as  
7

$$\text{negentropy}(z) \triangleq \mathbb{H}(\mathcal{N}(\mu, \sigma^2)) - \mathbb{H}(z) \quad (29.133)$$

10 where  $\mu = \mathbb{E}[z]$  and  $\sigma^2 = \mathbb{V}[z]$ . Since the Gaussian is the maximum entropy distribution, this  
11 measure is always non-negative and becomes large for distributions that are highly non-Gaussian.

12 We can define our objective as maximizing

$$J(\mathbf{V}) = \sum_j \text{negentropy}(z_j) = \sum_j \mathbb{H}(\mathcal{N}(\mu_j, \sigma_j^2)) - \mathbb{H}(z_j) \quad (29.134)$$

17 where  $\mathbf{z} = \mathbf{V}\mathbf{x}$ . Since we assume  $\mathbb{E}[\mathbf{z}] = \mathbf{0}$  and  $\text{Cov}[\mathbf{z}] = \mathbf{I}$ , the first term is a constant. Hence

$$J(\mathbf{V}) = \sum_j -\mathbb{H}(z_j) + \text{const} = \sum_j \mathbb{E}[\log p(z_j)] + \text{const} \quad (29.135)$$

21 which we see is equal (up to a sign change, and irrelevant constants) to the log-likelihood in  
22 Equation (29.132).

24 **29.6.4.2 Minimizing total correlation**

26 In Section 5.3.5.1, we show that the total correlation of  $\mathbf{z}$  is given by

$$\text{TC}(\mathbf{z}) = \sum_j \mathbb{H}(z_j) - \mathbb{H}(\mathbf{z}) = D_{\text{KL}} \left( p(\mathbf{z}) \middle\| \prod_j p_k(z_j) \right) \quad (29.136)$$

31 This is zero iff the components of  $\mathbf{z}$  are all mutually independent. In Section 22.3.2.2, we show that  
32 minimizing this results in a representation that is **disentangled**.

34 Now since  $\mathbf{z} = \mathbf{V}\mathbf{x}$ , we have

$$\text{TC}(\mathbf{z}) = \sum_j \mathbb{H}(z_j) - \mathbb{H}(\mathbf{V}\mathbf{x}) \quad (29.137)$$

38 Since we constrain  $\mathbf{V}$  to be orthogonal, we can drop the last term, since  $\mathbb{H}(\mathbf{V}\mathbf{x}) = \mathbb{H}(\mathbf{x}) = \text{const}$ , since  
39 multiplying by  $\mathbf{V}$  does not change the shape of the distribution. Hence we have  $\text{TC}(\mathbf{z}) = \sum_k \mathbb{H}(z_k)$ .  
40 Minimizing this is equivalent to maximizing the negentropy, which is equivalent to maximum  
41 likelihood.

43 **29.6.4.3 Maximizing mutual information (InfoMax)**

45 Let  $z_j = \phi(\mathbf{v}_j^\top \mathbf{x}) + \epsilon$  be the noisy output of an encoder, where  $\phi$  is some nonlinear scalar function,  
46 and  $\epsilon \sim \mathcal{N}(0, 1)$ . It seems reasonable to try to maximize the information flow through this system, a  
47

principle known as **infomax** [Lin88b; BS95]. That is, we want to maximize the mutual information between  $\mathbf{z}$  (the internal neural representation) and  $\mathbf{x}$  (the observed input signal). We have  $I(\mathbf{x}; \mathbf{z}) = H(\mathbf{z}) - H(\mathbf{z}|\mathbf{x})$ , where the latter term is constant if we assume the noise has constant variance. One can show that we can approximate the former term as follows

$$\mathbb{H}(\mathbf{z}) = \sum_j \mathbb{E} [\log \phi'(\mathbf{v}_j^\top \mathbf{x})] + \log |\det(\mathbf{V})| \quad (29.138)$$

where, as usual, we can drop the last term if  $\mathbf{V}$  is orthogonal. If we define  $\phi(z)$  to be a cdf, then  $\phi'(z)$  is its pdf, and the above expression is equivalent to the log likelihood. In particular, if we use a logistic nonlinearity,  $\phi(z) = \sigma(z)$ , then the corresponding pdf is the logistic distribution, and  $\log \phi'(z) = \log \cosh(z)$ , which matches Equation (29.126) (ignoring irrelevant constants). Thus we see that infomax is equivalent to maximum likelihood.

14

### 15 29.6.5 Sparse coding

16 In this section, we consider an extension of ICA to the case where we allow for observation noise (using a Gaussian likelihood), and we allow for a non-square mixing matrix  $\mathbf{W}$ . We also use a Laplace prior for  $\mathbf{z}$ . The resulting model is as follows:

$$\begin{aligned} p(\mathbf{z}, \mathbf{x}) &= p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \left[ \prod_k \text{Lap}(z_k|\lambda) \right] \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z}, \sigma^2\mathbf{I}) \end{aligned} \quad (29.139)$$

24 Thus each observation  $\mathbf{x}$  is approximated by a sparse combination of columns of  $\mathbf{W}$ , known as **basis**  
25 **functions**; the sparse vector of weights is given by  $\mathbf{z}$ . (This can be thought of as a form of sparse  
26 factor analysis, except the sparsity is in the latent code  $\mathbf{z}$ , not the weight matrix  $\mathbf{W}$ .)

27 Not all basis functions will be active for any given observation, due to the sparsity penalty.  
28 Hence we can allow for more latent factors  $K$  than observations  $D$ . This is called **overcomplete**  
29 **representation**.

30 If we have a batch of  $N$  examples, stored in the rows of  $\mathbf{X}$ , the log joint becomes

$$\log p(\mathbf{X}, \mathbf{Z}|\mathbf{W}) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{x}_n - \mathbf{W}\mathbf{z}_n\|_2^2 + \lambda \|\mathbf{z}_n\|_1 = \frac{1}{2} \|\mathbf{X} - \mathbf{W}\mathbf{Z}\|_F^2 + \lambda \|\mathbf{Z}\|_{1,1} \quad (29.140)$$

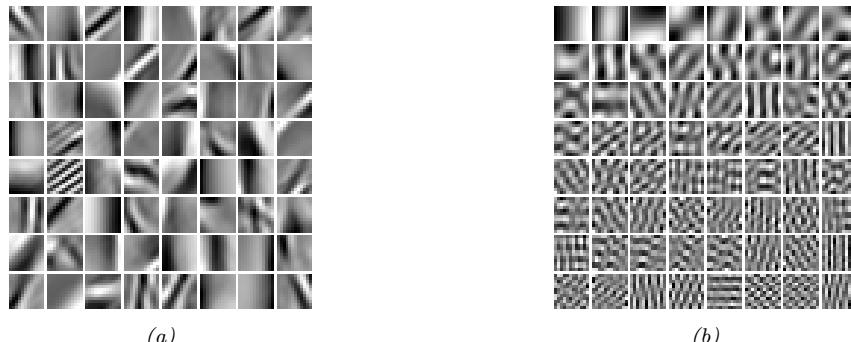
34 The MAP inference problem consists of estimating  $\mathbf{Z}$  for a fixed  $\mathbf{W}$ ; this is known as **sparse coding**,  
35 and can be solved using standard algorithms for sparse linear regression (see Section 15.2.5).<sup>4</sup>

36 The learning problem consists of estimating  $\mathbf{W}$ , marginalizing out  $\mathbf{Z}$ . This is called **dictionary**  
37 **learning**. Since this is computationally difficult, it is common to jointly optimize  $\mathbf{W}$  and  $\mathbf{Z}$  (thus  
38 “maxing out”  $\mathbf{Z}$  instead of marginalizing it out). We can do this by applying alternating optimization  
39 to Equation (29.140): estimating  $\mathbf{Z}$  given  $\mathbf{W}$  is a sparse linear regression problem, and estimating  $\mathbf{W}$   
40 given  $\mathbf{Z}$  is a simple least squares problem. (For faster algorithms, see [Mai+10].)

42 Figure 29.25(a) illustrates the results of dictionary learning when applied to a dataset of natural  
43 image patches. (Each patch is first centered and normalized to unit norm.) We see that the method

44 4. Solving an  $\ell_1$  optimization problem for each data example can be slow. However, it is possible to train a neural  
45 network to approximate the outcome of this process; this is known as **predictive sparse decomposition** [KRL08;  
46 GL10].

47



*Figure 29.25: Illustration of the filters learned by various methods when applied to natural image patches. (a) Sparse coding. (b) PCA. Generated by [sparse\\_dict\\_demo.ipynb](#).*

has learned bar and edge detectors that are similar to the simple cells in the primary visual cortex of the mammalian brain [OF96]. By contrast, PCA results in sinusoidal gratings, as shown in Figure 29.25(b).<sup>5</sup>

### 29.6.6 Nonlinear ICA

There are various ways to extend ICA to the nonlinear case. The resulting methods are similar to variational autoencoders (Chapter 22). For details, see e.g., [KKH20].

<sup>5</sup>. The reason PCA discovers sinusoidal grating patterns is because it is trying to model the covariance of the data, which, in the case of image patches, is translation invariant. This means  $\text{Cov}[I(x, y), I(x', y')] = f[(x - x')^2 + (y - y')^2]$  for some function  $f$ , where  $I(x, y)$  is the image intensity at location  $(x, y)$ . One can show (see e.g., [HHH09, p125]) that the eigenvectors of a matrix of this kind are always sinusoids of different phases, i.e., PCA discovers a **Fourier basis**.



# 30 Hidden Markov models

## 30.1 Introduction

In Section 2.8, we discussed Markov models. However, the assumption that we only need to know the current observation,  $\mathbf{y}_t$ , to predict the future,  $\mathbf{y}_{t+\tau}$ , without needing to know the past,  $\mathbf{y}_{1:t-1}$ , is a very limiting assumption. To allow the model to have “infinite” memory, we need to use a **hidden variable**  $\mathbf{z}_t$ , which summarizes *all* the past history,  $\mathbf{y}_{1:t}$ .

We now discuss one model of this kind, known as a **hidden Markov model** or **HMM**. This corresponds to a joint distribution of the following form:

$$p(\mathbf{y}_{1:T}, \mathbf{z}_{1:T}) = \left[ p(z_1) \prod_{t=2}^T p(z_t | z_{t-1}) \right] \left[ \prod_{t=1}^T p(\mathbf{y}_t | z_t) \right] \quad (30.1)$$

We see that  $\mathbf{z}_{1:T}$  corresponds to a first-order Markov chain in a latent state space, and at each time step, the model generates an observation  $\mathbf{y}_t$ . Thus we can think of an HMM as a **partially observed Markov process**. This is an example of a more general family of models known as state-space models, discussed in Chapter 31.

Figure 30.1a shows an HMM as a graphical model, where we unroll the model for 3 time steps. Figure 30.12 shows the corresponding plate version, where the parameter nodes are shown explicitly. The plates can capture how the parameters are shared across samples  $n$  (representing parameter tying), but it cannot capture the repetitive structure of the temporal “backbone” of the model.

## 30.2 HMMs: parameterization

### 30.2.1 Transition model

In an HMM, the hidden state  $z_t$  is discrete, so we can define the **transition model** as follows:

$$p(z_t = j | z_{t-1} = i) = A_{ij} \quad (30.2)$$

Here the  $i$ ’th row corresponds to the outgoing distribution from state  $i$ . This is a **row stochastic matrix**, meaning each row sums to one. We can visualize the non-zero entries in the transition matrix by creating a state transition diagram, as shown in Figure 2.16.

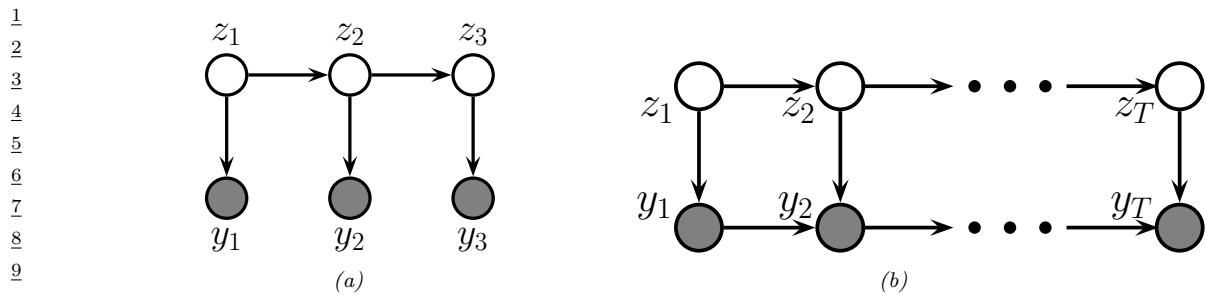


Figure 30.1: (a) An HMM represented as a PGM-D unrolled for 3 time steps. (b) An auto-regressive HMM of order 1.

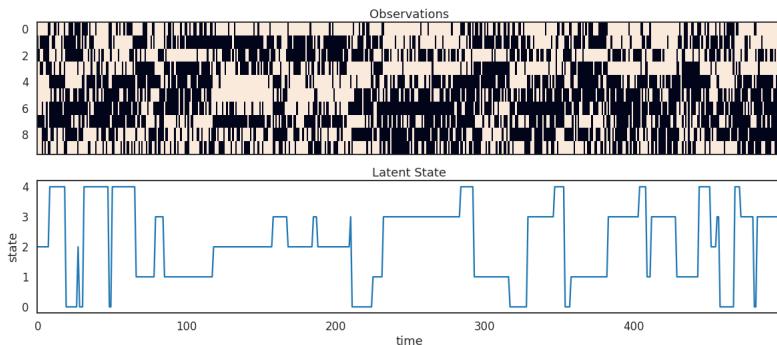


Figure 30.2: Some samples from an HMM with 5 Bernoulli observables. Generated by [bernoulli\\_hmm\\_example.ipynb](#).

### 30.2.2 Observation model

The term  $p(y_t|z_t = j)$  is the **observation model**. The form of this distribution depends on the type of data, for example, whether it is discrete or continuous. We give some examples below.

#### 30.2.2.1 Categorical likelihood

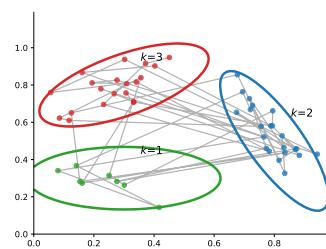
If  $y_t$  is discrete, it is common to represent  $p(y_t|z_t)$  by a set of categorical distributions, with one distribution per hidden state. In particular, we can define

$$p(y_t = k|z_t = j) = B_{jk} \quad (30.3)$$

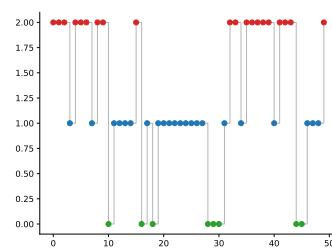
which is the probability of emitting symbol  $k$  from state  $j$ . Here  $\mathbf{B}$  is a row stochastic matrix, similar to  $\mathbf{A}$ . See Section 8.1.2 for an example.

If we have  $D$  discrete observations per time step, we can use a factorial model of the form

$$p(\mathbf{y}_t|z_t = j) = \prod_{d=1}^D \text{Cat}(y_{td}|\mathbf{B}_{d,j,:}) \quad (30.4)$$



(a)



(b)

Figure 30.3: (a) Some 2d data sampled from a 3 state HMM. Each state emits from a 2d Gaussian. (b) The hidden state sequence. Adapted from Figure 13.8 of [Bis06]. Generated by `hmm_lillypad.py`.

In the special case of binary observations, this becomes

$$p(\mathbf{y}_t | z_t = j) = \prod_{d=1}^D \text{Ber}(y_{td} | B_{d,j}) \quad (30.5)$$

In Figure 30.2, we give an example of an HMM with 5 hidden states and 10 Bernoulli observables.

### 30.2.2.2 Poisson likelihood

In Section 30.3.1, we give a worked example of an HMM which models a timeseries of integer counts using a Poisson observation model.

### 30.2.2.3 Gaussian and GMM likelihood

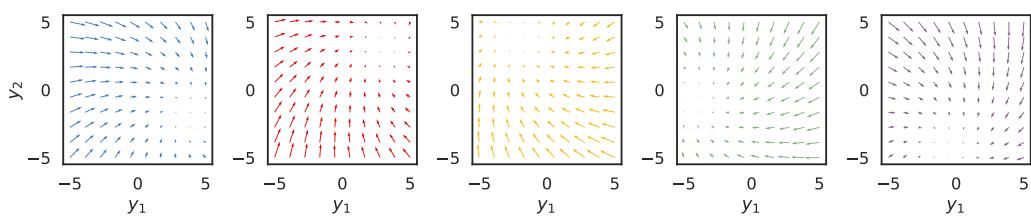
If  $\mathbf{y}_t$  is continuous, it is common to use a Gaussian observation model:

$$p(\mathbf{y}_t | z_t = j) = \mathcal{N}(\mathbf{y}_t | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \quad (30.6)$$

As a simple example, suppose we have an HMM with 3 hidden states, each of which generates a 2d Gaussian. We can represent these Gaussian distributions are 2d ellipses, as shown in Figure 30.3(a). We call these “lilly pads”, because of their shape. We can imagine a frog hopping from one lilly pad to another. (This analogy is due to the late Sam Roweis.) It will stay on a pad for a while (corresponding to remaining in the same discrete state  $z_t$ ), and then jump to a new pad (corresponding to a transition to a new state). See Figure 30.3(b). The data we see are just the 2d points (e.g., water droplets) coming from near the pad that the frog is currently on. Thus this model is like a Gaussian mixture model (Section 29.2.1), in that it generates clusters of observations, except now there is temporal correlation between the data points.

We can also use more flexible observation models. For example, if we use a  $K$ -component GMM, then we have

$$p(\mathbf{y}_t | z_t = j) = \sum_{k=1}^K \pi_{jk} \mathcal{N}(\mathbf{y}_t | \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}) \quad (30.7)$$



10 *Figure 30.4: Illustration of the observation dynamics for each of the 5 hidden states. The attractor*  
 11 *point corresponds to the steady state solution for the corresponding autoregressive process. Generated*  
 12 *by [arhmm\\_example.ipynb](#).*

13

14

#### 15 30.2.2.4 Autoregressive likelihoods

16 The standard HMM assumes the observations are conditionally independent given the hidden state.  
 17 In practice this is often not the case. However, it is straightforward to have direct arcs from  $\mathbf{y}_{t-1}$  to  
 18  $\mathbf{y}_t$  as well as from  $z_t$  to  $\mathbf{y}_t$ , as in Figure 30.1b. This is known as an **auto-regressive HMM**.

19 For continuous data, we can use an observation model of the form

20

$$21 p(\mathbf{y}_t | \mathbf{y}_{t-1}, z_t = j, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{y}_t | \mathbf{W}_j \mathbf{y}_{t-1} + \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \quad (30.8)$$

22

23 This is a linear regression model, where the parameters are chosen according to the current hidden  
 24 state. (We could also use a nonlinear model, such as a neural network.) Such models are widely  
 25 used in econometrics, where they are called **regime switching Markov model** [Ham90]. Similar  
 26 models can be defined for discrete observations.

27 We can also consider higher-order extensions, where we condition on the last  $L$  observations:

28

$$29 p(\mathbf{y}_t | \mathbf{y}_{t-L:t-1}, z_t = j, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{y}_t | \sum_{\ell=1}^L \mathbf{W}_{j,\ell} \mathbf{y}_{t-\ell} + \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \quad (30.9)$$

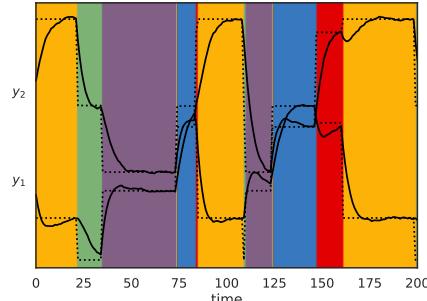
30

31 The AR-HMM essentially combines two Markov chains, one on the hidden variables, to capture long  
 32 range dependencies, and one on the observed variables, to capture short range dependencies [Ber99].  
 33 Since all the visible nodes are observed, adding connections between them just changes the likelihood,  
 34 but does not complicate the task of posterior inference (see Section 8.3.3).

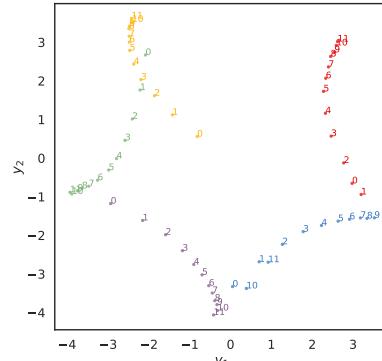
35 Let us now consider a 2d example of this, due to Scott Linderman. We use a left-to-right transition  
 36 matrix with 5 states. In addition, the final state returns to first state, so we just cycle through the  
 37 states. Let  $\mathbf{y}_t \in \mathbb{R}^2$ , and suppose we set  $\mathbf{W}_j$  to a rotation matrix with a small angle of 7 degrees, and  
 38 we set each  $\boldsymbol{\mu}_j$  to 72 degrees, so each state rotates 1/5 of the way around the circle. If the model stays  
 39 in the same state  $j$  for a long time, the observed dynamics will converge to the steady state  $\mathbf{y}_{*,j}$ , which  
 40 satisfies  $\mathbf{y}_{*,j} = \mathbf{W}_j \mathbf{y}_{*,j} + \boldsymbol{\mu}_j$ ; we can solve for the steady state vector using  $\mathbf{y}_{*,j} = (\mathbf{I} - \mathbf{W}_j)^{-1} \boldsymbol{\mu}_j$ .  
 41 We can visualize the induced 2d flow for each of the 5 states as shown in Figure 30.4.

42 In Figure 30.5(a), we show a trajectory sampled from this model. We see that the two components  
 43 of the observation vector undergo different dynamics, depending on the underlying hidden state. In  
 44 Figure 30.5(b), we show the same data in a 2d scatter plot. The first observation is the yellow dot  
 45 (from state 2) at  $(-0.8, 0.5)$ . The dynamics converge to the stationary value of  $\mathbf{y}_{*,2} = (-2.0, 3.8)$ .

46



(a)



(b)

Figure 30.5: Samples from the 2d AR-HMM. (a) Time series plot of  $y_{t,1}$  and  $y_{t,2}$ . (The latter are shifted up vertically by 4.) (b) The background color is the generating state. The dotted lines represent the stationary value for that component of the observation. (b) Scatter plot of observations. Colors denote the generating state. We show the first 12 samples from each state. Generated by [arhmm\\_example.ipynb](#).

Then the system jumps to the green state (state 3), so it adds an offset of  $\mathbf{b}_3$  to the last observation, and then converges to the staionary value of  $\mathbf{y}_{*,3} = (-4.3, -0.8)$ . And so on.

### 30.2.2.5 “Deep” likelihoods

We can easily use (conditional) deep generative models, such as VAEs (Section 22.2) or normalizing flows (Chapter 24), as the likelihood model. For example, [HNBK18] shows how to perform unsupervised learning of grammatical structure from sequences of word embeddings, as opposed to sequences of words, using a mixture of normalizing flows as the observation model.

## 30.3 HMMS: Applications

### 30.3.1 Segmentation of time series data

In this section, we give a variant of the casino example from Section 8.1.2, where our goal is to segment a time series into different regimes, each of which corresponds to a different statistical distribution. In Figure 30.6a we show the data, corresponding to counts generated from some process (e.g., visits to a web site, or number of infections). We see that the count rate seems to be roughly constant for a while, and then changes at certain points. We would like to segment this data stream into  $K$  different regimes or states, each of which is associated with a Poisson observation model with rate  $\lambda_k$ :

$$p(y_t | z_t = k) = \text{Poi}(y_t | \lambda_k) \quad (30.10)$$

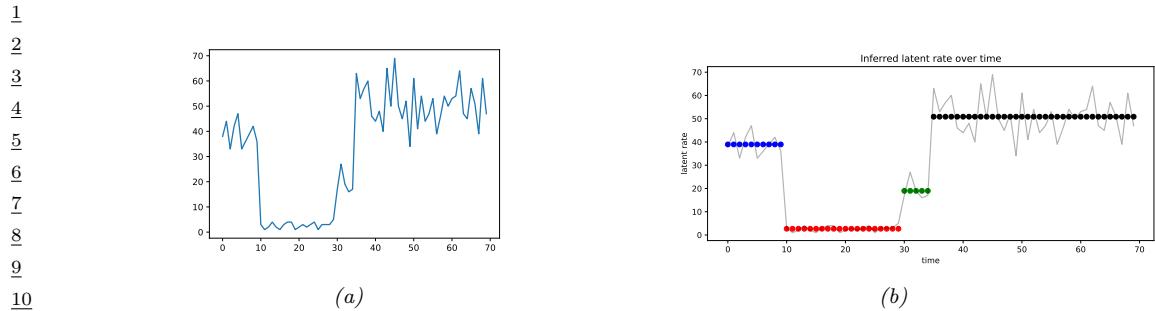


Figure 30.6: (a) A sample time series dataset of counts. (b) A segmentation of this data using a 4 state HMM. Generated by [hmm\\_poisson\\_changepoint\\_jax.ipynb](#).

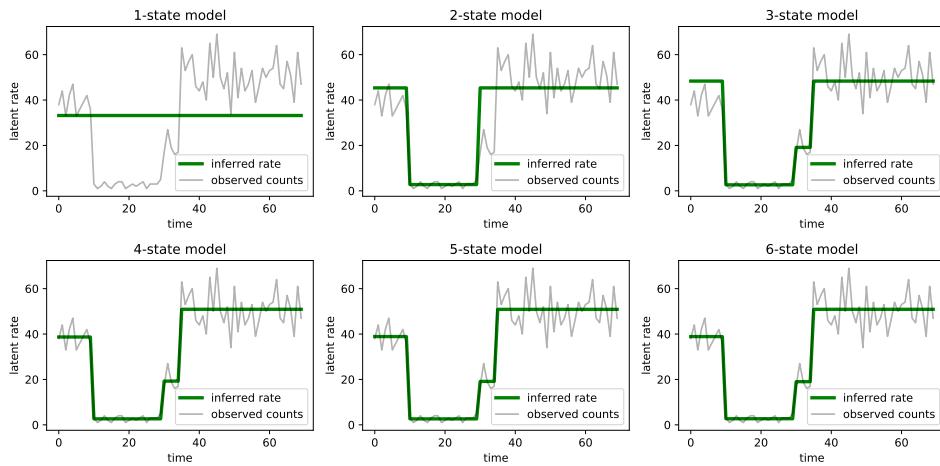


Figure 30.7: Segmentation of the time series using HMMs with 1–6 states. Generated by [hmm\\_poisson\\_changepoint\\_jax.ipynb](#).

We use a uniform prior over the initial states. For the transition matrix, we the Markov chain stays in the same state with probability  $p = 0.95$ , and otherwise transitions to one of the other  $K - 1$  states uniformly at random:

$$z_1 \sim \text{Categorical} \left( \left\{ \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4} \right\} \right) \quad (30.11)$$

$$z_t | z_{t-1} \sim \text{Categorical} \left( \left\{ \begin{array}{ll} p & \text{if } z_t = z_{t-1} \\ \frac{1-p}{4-1} & \text{otherwise} \end{array} \right\} \right) \quad (30.12)$$

We compute a MAP estimate for the parameters  $\lambda_{1:K}$  using a log-Normal(5,5) prior. We optimize the log of the Poisson rates using gradient descent, initializing the parameters at a random value centered on the log of the overall count means. We show the results in Figure 30.6b. See the method has successfully partitioned the data into 4 regimes, which is in fact how it was generated. (We

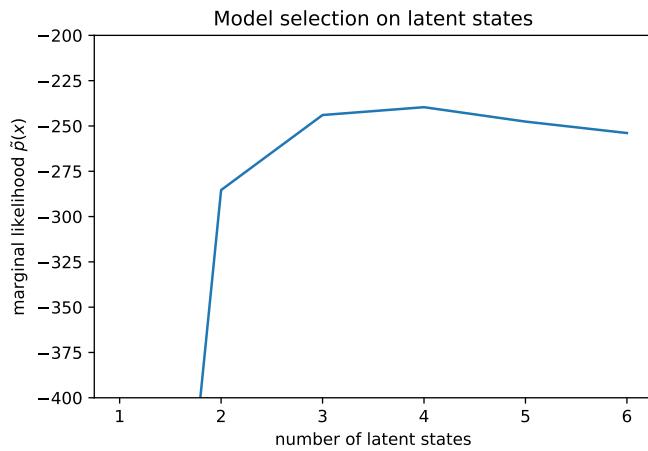


Figure 30.8: Marginal likelihood vs number of states  $K$  in the Poisson HMM. Generated by `hmm_poisson_changepoint_jax.ipynb`.

generating rates are  $\lambda = (40, 3, 20, 50)$ , with the changepoints happening at times  $(10, 30, 35)$ .)

In general we don't know the optimal number of states  $K$ . To solve this, we can fit many different models, as shown in Figure 30.7, for  $K = 1 : 6$ . We see that after  $K \geq 3$ , the model fits are very similar, since multiple states get associated to the same regime. We can pick the “best”  $K$  to be the one with the highest marginal likelihood. Rather than summing over both discrete latent states and integrating over the unknown parameters  $\lambda$ , we just maximize over the parameters (empirical Bayes approximation):

$$p(\mathbf{y}_{1:T}|K) \approx \max_{\lambda} \sum_{\mathbf{z}} p(\mathbf{y}_{1:T}, \mathbf{z}_{1:T}|\lambda, K) \quad (30.13)$$

We show this plot in Figure 30.8. We see the peak is at  $K = 3$  or  $K = 4$ ; after that it starts to go down, due to the Bayesian Occam’s razor effect.

### 30.3.2 Spelling correction

In this section, we illustrate how to use an HMM for **spelling correction**. The goal is to infer the sequence of words  $\mathbf{z}_{1:T}$  that the user meant to type, given observations of what they actually did type,  $\mathbf{y}_{1:T}$ .

#### 30.3.2.1 Baseline model

We start by using a simple unigram language model, so  $p(\mathbf{z}_{1:T}) = \prod_{1:T} p(z_t)$ , where  $p(z_t = k)$  is the prior probability of word  $k$  being used. These probabilities can be estimated by simply normalizing word frequency counts from a large training corpus. We ignore any Markov structure.

Now we turn to the observation model,  $p(y_t = v|z_t = k)$ , which is the probability the user types word  $v$  when they meant to type word  $k$ . For this, we use a **noisy channel model**, in which the

<sup>1</sup> “message”  $z_t$  gets corrupted by one of four kinds of error: substitution error, where we swap one  
<sup>2</sup> letter for another (e.g., “government” mistyped as “governmment”); transposition errors, where we  
<sup>3</sup> swap the order of two adjacent letters (e.g., “government” mistyped as “govermnent”); deletion errors,  
<sup>4</sup> where we omit one letter (e.g., “government” mistyped as “goverment”); and insertion errors, where  
<sup>5</sup> we add an extra latter (e.g., “government” mistyped as “governmennt”). If  $y$  differs from  $z$  by  $d$  such  
<sup>6</sup> errors, we say that  $y$  and  $z$  have an **edit distance** of  $d$ . Let  $\mathcal{D}(y, d)$  be the set of words that are edit  
<sup>7</sup> distance  $d$  away from  $y$ . We can then define the following likelihood function:  
<sup>8</sup>

$$\begin{aligned} \text{where } p_1 &> p_2 > p_3 > p_4. \\ \text{We can combine the likelihood with the prior to get the overall score for each hypothesis (i.e., candidate correction). This simple model, which was proposed by Peter Norvig}\textcolor{red}{^1}, \text{can work quite well. However, it also has some flaws. For example, the error model assumes that the smaller the edit distance, the more likely the word, but this is not always valid. For example, “reciet” gets corrected to “recite” instead of “receipt”, and “adres” gets corrected to “acres” not “address”. We can fix this problem by learning the parameters of the noise model based on a labeled corpus of } (z, x) \text{ pairs derived from actual spelling errors. One possible way to get such a corpus is to look at web search behavior: if a user types query } q_1 \text{ and then quickly changes it to } q_2 \text{ followed by a click on a link, it suggests that } q_2 \text{ is a manual correction for } q_1, \text{ so we can set } (z = q_2, y = q_1). \text{ This heuristic has been used in the Etsy search engine.}\textcolor{red}{^2} \text{ It is also possible to manually collect such data (see e.g., [Hag+17]), or to algorithmically create } (z, y) \text{ pairs, where } y \text{ is an automatically generated misspelling of } z \text{ (see e.g., [ECM18]).}$$

### 30.3.2.2 HMM model

The baseline model can work well, but has room for improvement. In particular, many errors will be hard to correct without context. For example, suppose the user typed “advice”: did they mean “advice” or “advise”? It depends on whether they intended to use a noun or a verb, which is hard to tell without looking at the sequence of words. To do this, we will “upgrade” our model to an HMM. We just have to replace our independence prior  $p(z_{1:T}) = \prod_t p(z_t)$  by a standard first-order language model on words,  $p(z_{1:T}) = \prod_t p(z_t|z_{t-1})$ . The parameters of this model can be estimated by counting bigrams in a large corpus of “clean” text (see Section 2.8.3.1). The observation model  $p(y_t|z_t)$  can remain unchanged.

Given this model, we can compute the top  $N$  most likely hidden sequences in  $O(NTK^2)$  time, where  $K$  is the number of hidden states, and  $T$  is the length of the sequence, as explained in Section 8.3.6.5. In a naive implementation, the number of hidden states  $K$  is the number of words in the vocabulary, which would make the method very slow. However, we can exploit sparsity of the

<sup>44</sup> 1. See his excellent tutorial at <http://norvig.com/spell-correct.html>.

<sup>45</sup> 2. See this blogpost by Mohit Nayyar for details: <https://codeascraft.com/2017/05/01/modeling-spelling-correction-for-search-at-etsy/>.

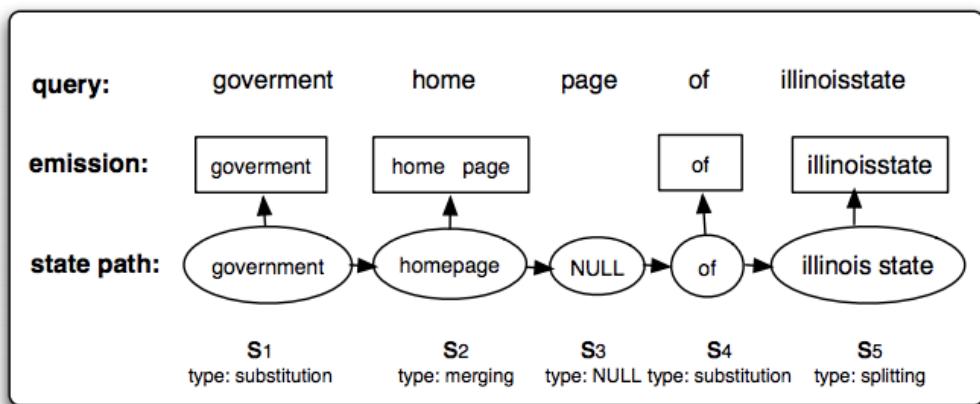


Figure 30.9: Illustration of an HMM applied to spelling correction. The top row, labeled “query”, represents the search query  $y_{1:T}$  typed by the user, namely “goverment home page of illnoisstate”. The bottom row, labeled “state path”, represents the most probable assignment to the hidden states,  $z_{1:T}$ , namely “government homepage of illinois state”. (The NULL state is a silent state, that is needed to handle the generation of two tokens from a single hidden state.) The middle row, labeled “emission”, represents the words emitted by each state, which match the observed data. From Figure 1 of [LDZ11].

likelihood function (i.e., the fact that  $p(y|z)$  is 0 for most values of  $z$ ) to generate small candidate lists of hidden states for each location in the sequence. This gives us a sparse belief state vector  $\alpha_t$ .

### 30.3.2.3 Extended HMM model

We can extend the HMM model to handle higher level errors, in addition to misspellings of individual words. In particular, [LDZ11; LDZ12] proposed modeling the following kinds of errors:

- Two words merged into one, e.g., “home page” → “homepage”.
- One word split into two, e.g., “illnoisstate” → “illinois state”.
- Within-word errors, such as substitution, transposition, insertion and deletion of letters, as we discussed in Section 30.3.2.2.

We can model this with an HMM, where we augment the state space with a **silent state**, that does not emit any symbols. Figure 30.9 illustrates how this model can “denoise” the observed query “goverment home page of illnoisstate” into the correctly formulated query “government homepage of illinois state”.

An alternative to using HMMs is to use supervised learning to fit a sequence-to-sequence translation model, using RNNs or transformers. This can work very well, but often needs much more training data, which can be problematic for **low-resource languages** [ECM18].

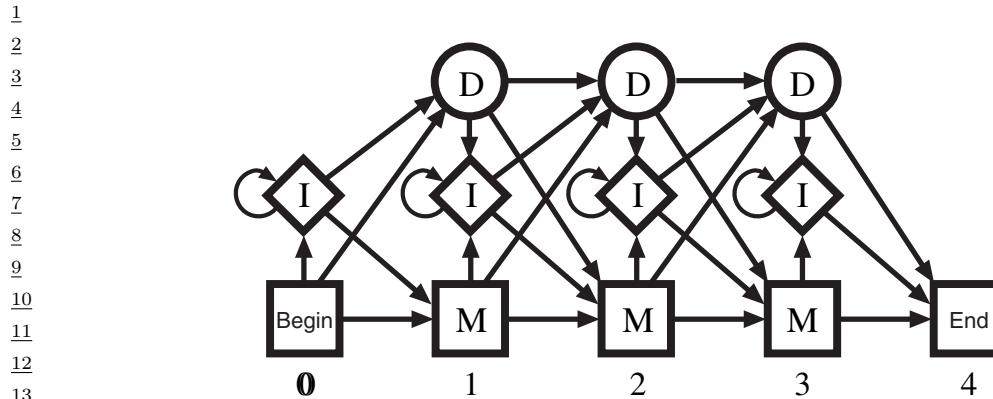


Figure 30.10: State transition diagram for a profile HMM. From Figure 5.7 of [Dur+98]. Used with kind permission of Richard Durbin.

Q5E940_BOVIN	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	76
RLAO_HUMAN	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	76
RLAO_MOUSE	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	76
RLAO_RAT	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	76
RLAO_CHICK	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	76
RLAO_RANBY	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	76
QT2UG3_BRAKE	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	76
RLAO_DROME	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	76
RLAO_DICD1	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	76
Q54120_PALDI	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	75
RLAO_PLADI	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	75
RLAO_SULSTO	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	75
RLAO_SULTO	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	79
RLAO_ARPE	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	80
RLAO_AERPE	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	86
RLAO_PYRAC	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	85
RLAO_METAC	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	78
RLAO_METAC	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	79
RLAO_ARCFU	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	75
RLAO_ARCFU	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	88
RLAO_METRA	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	88
RLAO_METRA	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	82
RLAO_METVA	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	82
RLAO_METVA	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	81
RLAO_PTYRO	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	77
RLAO_PTYRO	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	77
RLAO_PTYRU	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	76
RLAO_HALVO	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	72
RLAO_HALVO	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	79
RLAO_HALSA	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	79
RLAO_HALSA	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	72
RLAO_THEVO	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	72
RLAO_PICTO	-----M-----G-----E-----R-----E-----T-----S-----W-----Y-----L-----K-----E-----I-----L-----D-----D-----P-----C-----F-----I-----V-----A-----D-----N-----Y-----K-----G-----K-----E-----R-----L-----G-----R-----K-----A-----I-----G-----K-----E-----H-----L-----E-----N-----M-----P-----A-----L-----E-----	72

Figure 30.11: Example of multiple sequence alignment. We show the first 90 positions of the acidic ribosomal protein P0 from several organisms. Colors represent functional properties of the corresponding amino acid. Dashes represent insertions or deletions. From [https://en.wikipedia.org/wiki/Multiple\\_sequence\\_alignment](https://en.wikipedia.org/wiki/Multiple_sequence_alignment). Used with kind permission of Wikipedia author Miguel Andrade.

### 30.3.3 Protein sequence alignment

An important application of HMMs is to the problem of **protein sequence alignment** [Dur+98]. Here the goal is to determine if a test sequence  $y_{1:T}$  belongs to a protein family or not, and if so, how it aligns with the canonical representation of that family. (Similar methods can be used to align DNA and RNA sequences.) The technique we will use is similar to the spelling correction model in Section 30.3.2.2, but can be trained in an unsupervised way from raw sequences, without having to manually create labeled  $(z, y)$  pairs.

To solve the alignment problem, let us initially assume we have a set of aligned sequences from a protein family, from which we can generate a **consensus sequence**. This defines a probability distribution over symbols at each location  $t$  in the string; denote each **position-specific scoring**

1 matrix by  $\theta_t(v) = p(y_t = v)$ . These parameters can be estimated by counting.

2 Now we turn the PSSM into an HMM with 3 hidden states, representing the events that the  
3 location  $t$  matches the consensus sequence,  $z_t = M$ , or inserts its own unique symbol,  $z_t = I$ , or  
4 deletes (skips) the corresponding consensus symbol,  $z_t = D$ . We define the observation models for  
5 these 3 events as follows. For matches, we use the PSSM  $p(y_t = v | z_t = M) = \theta_t(v)$ . For insertions  
6 we use the uniform distribution  $p(y_t = v | z_t = I) = 1/V$ , where  $V$  is the size of the vocabulary. For  
7 deletions, we use  $p(y_t = - | z_t = D)$ , where “-” is a special deletion symbol used to pad the generated  
8 sequence to the correct length. The corresponding state transition matrix is shown in Figure 30.10:  
9 we see that matches and deletions advance one location along the consensus sequence, but insertions  
10 stay in the same location (represented by the self-transition from  $I$  to  $I$ ). This model is known as a  
11 **profile HMM**.

12 Given a profile HMM with consensus parameters  $\theta$ , we can compute  $p(\mathbf{y}_{1:T} | \theta)$  in  $O(T)$  time using  
13 the forwards algorithm, as described in Section 8.3.1. This can be used to decide if the sequence  
14 belongs to this family or not, by thresholding the log-odds score,  $L(\mathbf{y}) = \log p(\mathbf{y} | \theta) / p(\mathbf{y} | \mathcal{M}_0)$ , where  
15  $\mathcal{M}_0$  is a baseline model, such as the uniform distribution. If the string matches, we can compute an  
16 alignment to the consensus using the Viterbi algorithm, as described in Section 8.3.6. See Figure 30.11  
17 for an illustration of such a **multiple sequence alignment**. If we don’t have an initial set of aligned  
18 sequences from which to compute the consensus sequence  $\theta$ , we can use the Baum-Welch algorithm  
19 (Section 30.4.1) to compute the MLE for the parameters  $\theta$  from a set of unaligned sequences. For  
20 details, see e.g., [Dur+98, Ch.6].

## 23 30.4 HMMS: parameter learning

25 In this section, we discuss how to compute a point estimate or the full posterior over the model  
26 parameters of an HMM given a set of partially observed sequences.

### 28 30.4.1 The Baum-Welch (EM) algorithm

30 In this section, we discuss how to compute an approximate MLE for the parameters of an HMM  
31 using the EM algorithm which is an iterative bound optimization algorithm (see Section 6.7.3 for  
32 details). When applied to HMMS, the resulting method is known as the **Baum-Welch** algorithm  
33 [Bau+70].

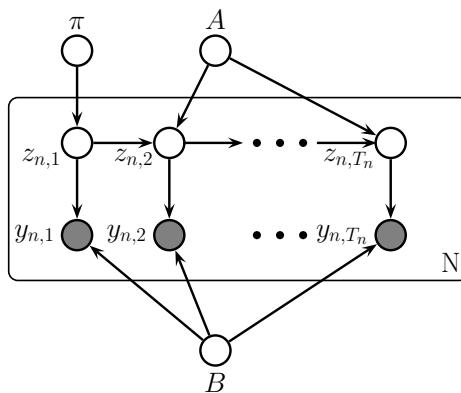
#### 35 30.4.1.1 Log likelihood

37 The joint probability of a single sequence is given by

$$\begin{aligned} \text{38} \quad p(\mathbf{y}_{1:T}, \mathbf{z}_{1:T} | \theta) &= [p(z_1 | \pi)] \left[ \prod_{t=2}^T p(z_t | z_{t-1}, \mathbf{A}) \right] \left[ \prod_{t=1}^T p(\mathbf{y}_t | z_t, \mathbf{B}) \right] \end{aligned} \quad (30.15)$$

$$\begin{aligned} \text{41} \quad &= \left[ \prod_{k=1}^K \pi_k^{\mathbb{I}(z_1=k)} \right] \left[ \prod_{t=2}^T \prod_{j=1}^K \prod_{k=1}^K A_{jk}^{\mathbb{I}(z_{t-1}=j, z_t=k)} \right] \left[ \prod_{t=1}^T \prod_{k=1}^K p(\mathbf{y}_t | \theta_k)^{\mathbb{I}(z_t=k)} \right] \end{aligned} \quad (30.16)$$

45 where  $\theta = (\pi, \mathbf{A}, \mathbf{B})$ . Of course, we cannot compute this objective, since  $\mathbf{z}_{1:T}$  is hidden. So instead  
46 we will optimize the expected complete data log likelihood, where expectations are taken using the  
47



*Figure 30.12: HMM with plate notation.  $A$  are the parameters for the state transition matrix  $p(z_t|z_{t-1})$  and  $B$  are the parameters for the discrete observation model  $p(x_t|z_t)$ .  $T_n$  is the length of the  $n$ 'th sequence.*

parameters from the previous iteration of the algorithm:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \mathbb{E}_{p(\mathbf{z}_{1:T}|\mathbf{y}_{1:T}, \boldsymbol{\theta}^{\text{old}})} [\log p(\mathbf{y}_{1:T}, \mathbf{z}_{1:T}|\boldsymbol{\theta})] \quad (30.17)$$

This can be easily summed over  $N$  sequences. See Figure 30.12 for the graphical model.

The above objective is a lower bound on the observed data log likelihood,  $\log p(\mathbf{y}_{1:T}|\boldsymbol{\theta})$ , so the entire procedure is a bound optimization method that is guaranteed to converge to a local optimum. (In fact, in the case of HMMs, it can be shown to converge to (close to) one of the global optima [YBW15].)

### 30.4.1.2 E step

Let  $A_{jk} = p(z_t = k|z_{t-1} = j)$  be the  $K \times K$  transition matrix. For the first time slice, let  $\pi_k = p(z_1 = k)$  be the initial state distribution. Let  $\boldsymbol{\theta}_k$  represent the parameters of the observation model for state  $k$ .

To compute the expected sufficient statistics, we first run the forwards-backwards algorithm on each sequence (see Section 8.3.3). This returns the following node and edge marginals:

$$\gamma_{n,t}(j) \triangleq p(z_t = j|\mathbf{y}_{n,1:T_n}, \boldsymbol{\theta}^{\text{old}}) \quad (30.18)$$

$$\xi_{n,t}(j, k) \triangleq p(z_{t-1} = j, z_t = k|\mathbf{y}_{n,1:T_n}, \boldsymbol{\theta}^{\text{old}}) \quad (30.19)$$

We can then derive the expected counts as follows (note that we pool the sufficient statistics across time, since the parameters are tied, as well as across sequences):

$$\mathbb{E}[N_k^1] = \sum_{n=1}^N \gamma_{n,1}(k), \quad \mathbb{E}[N_k] = \sum_{n=1}^M \sum_{t=2}^{T_n} \gamma_{n,t}(k), \quad \mathbb{E}[N_{jk}] = \sum_{n=1}^N \sum_{t=2}^{T_n} \xi_{n,t}(j, k) \quad (30.20)$$

Given the above quantities, we can compute the expected complete data log likelihood as follows:

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) &= \sum_{k=1}^K \mathbb{E}[N_k^1] \log \pi_k + \sum_{j=1}^K \sum_{k=1}^K \mathbb{E}[N_{jk}] \log A_{jk} \\ &\quad + \sum_{n=1}^N \sum_{t=1}^{T_n} \sum_{k=1}^K p(z_t = k | \mathbf{y}_n, \boldsymbol{\theta}^{\text{old}}) \log p(\mathbf{y}_{n,t} | \boldsymbol{\theta}_k) \end{aligned} \quad (30.21)$$

where  $T_n$  is the length of sequence  $n$ .

### 30.4.1.3 M step

We can estimate the transition matrix and initial state probabilities by maximizing the objective subject to the sum to one constraint. The result is just a normalized version of the expected counts:

$$\hat{A}_{jk} = \frac{\mathbb{E}[N_{jk}]}{\sum_{k'} \mathbb{E}[N_{jk'}]}, \quad \hat{\pi}_k = \frac{\mathbb{E}[N_k^1]}{N} \quad (30.22)$$

This result is quite intuitive: we simply add up the expected number of transitions from  $j$  to  $k$ , and divide by the expected number of times we transition from  $j$  to anything else.

For a categorical observation model, the expected sufficient statistics are

$$\mathbb{E}[M_{kv}] = \sum_{n=1}^N \sum_{t=1}^{T_n} \gamma_{n,t}(k) \mathbb{I}(\mathbf{y}_{n,t} = v) = \sum_{n=1}^N \sum_{t: \mathbf{y}_{n,t}=v} \gamma_{n,t}(k) \quad (30.23)$$

The M step has the form

$$\hat{B}_{kv} = \frac{\mathbb{E}[M_{kv}]}{\mathbb{E}[N_k]} \quad (30.24)$$

This result is quite intuitive: we simply add up the expected number of times we are in state  $k$  and we see a symbol  $v$ , and divide by the expected number of times we are in state  $k$ . See Algorithm 11 for the pseudocode.

For a Gaussian observation model, the expected sufficient statistics are given by

$$\mathbb{E}[\bar{\mathbf{y}}_k] = \sum_{n=1}^N \sum_{t=1}^{T_n} \gamma_{n,t}(k) \mathbf{y}_{n,t}, \quad \mathbb{E}[(\bar{\mathbf{y}}\bar{\mathbf{y}})^T_k] = \sum_{n=1}^N \sum_{t=1}^{T_n} \gamma_{n,t}(k) \mathbf{y}_{n,t} \mathbf{y}_{n,t}^T \quad (30.25)$$

The M step becomes

$$\hat{\mu}_k = \frac{\mathbb{E}[\bar{\mathbf{y}}_k]}{\mathbb{E}[N_k]} \quad (30.26)$$

$$\hat{\Sigma}_k = \frac{\mathbb{E}[(\bar{\mathbf{y}}\bar{\mathbf{y}})^T_k] - \mathbb{E}[N_k] \hat{\mu}_k \hat{\mu}_k^T}{\mathbb{E}[N_k]} \quad (30.27)$$

In practice, we often need to add a log prior to these estimates to ensure the resulting  $\hat{\Sigma}_k$  estimate is well-conditioned. See [Mur22, Sec 4.5.2] for details.

---

1  
2   **Algorithm 32:** Baum Welch algorithm for (discrete observation) HMMs  
3   1 Initialize parameters  $\theta$  ;  
4   2 **for** each iteration **do**  
5   3    // E step ;  
6   4    Initialize expected counts:  $\mathbb{E}[N_k] = 0$ ,  $\mathbb{E}[N_{jk}] = 0$ ,  $\mathbb{E}[M_{kv}] = 0$ ;  
7   5    **for** each datacase  $n$  **do**  
8   6      Use forwards-backwards algorithm on  $y_n$  to compute  $\gamma_{n,t}$  and  $\xi_{n,t}$   
9   7      ( Equations 30.18–30.19 ) ;  
10   8       $\mathbb{E}[N_k] := \mathbb{E}[N_k] + \sum_{t=2}^{T_n} \gamma_{n,t}(k)$  ;  
11   9       $\mathbb{E}[N_{jk}] := \mathbb{E}[N_{jk}] + \sum_{t=2}^{T_n} \xi_{n,t}(j, k)$  ;  
12   9       $\mathbb{E}[M_{kv}] := \mathbb{E}[M_{kv}] + \sum_{t:x_{n,t}=v} \gamma_{n,t}(k)$   
13  
14   10     // M step ;  
15   11     Compute new parameters  $\theta = (\mathbf{A}, \mathbf{B}, \pi)$  using Equations 30.22  
16

---

17  
18  
19 **30.4.1.4 Initialization**

20 As usual with EM, we must take care to ensure that we initialize the parameters carefully, to minimize  
21 the chance of getting stuck in poor local optima. There are several ways to do this, such as  
22

- 23   • Use some fully labeled data to initialize the parameters.  
24  
25   • Initially ignore the Markov dependencies, and estimate the observation parameters using the  
26    standard mixture model estimation methods, such as K-means or EM.  
27  
28   • Randomly initialize the parameters, use multiple restarts, and pick the best solution.

29   Techniques such as deterministic annealing [UN98; RR01a] can help mitigate the effect of local  
30 minima. Also, just as K-means is often used to initialize EM for GMMs, so it is common to initialize  
31 EM for HMMs using Viterbi training. The Viterbi algorithm is explained in Section 8.3.6, but  
32 basically it is an algorithm to compute the single most probable path. As an approximation to  
33 the E step, we can replace the sum over paths with the statistics computed using this single path.  
34 Sometimes this can give better results [AG11].  
35

36  
37 **30.4.1.5 Example: casino HMM**

38 In this section, we fit the casino HMM from Section 8.1.2. The true generative model is shown in  
39 Figure 30.13a. We used this to generate 5 sequences of varying length (max 3000), totalling 5670  
40 observations. We initialized the model with random parameters, and then ran EM for 20 iterations.  
41 We got the results in Figure 30.13b.  
42

43 **30.4.1.6 Example: AR-HMM**

44 In this section, we revisit the 2d AR-HMM example from Section 30.2.2.4. We generate a single  
45 sequence of length 10,000, and fit the model using EM. In Figure 30.14(a) we show samples from the  
46  
47

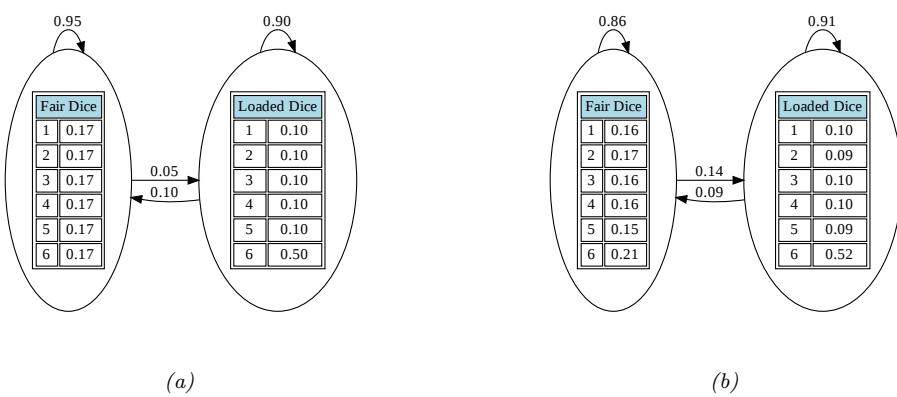


Figure 30.13: Illustration of the casino HMM. (a) True parameters used to generate the data. (b) Estimated parameters, using EM for 20 iterations. Generated by `hmm_casino_em_train.py`.

true model, and in Figure 30.14(b) we show samples from the learned model. We can see that the results are very similar, modulo the change in color due to label switching.

To avoid the label switching problem, we can find the optimal permutation between the learned states and the true states by solving a **linear assignment problem**, also called a **minimum weight matching** in a bipartite graph. This finds the binary matrix  $\mathbf{X}$  which minimizes

$$\mathcal{L}(\mathbf{X}) = \sum_i \sum_j C_{ij} X_{ij} \quad (30.28)$$

where  $X_{ij} = 1$  if row  $i$  is assigned to column  $j$ , and  $C_{ij}$  is the cost. This can be computed using the **Hungarian algorithm**. In the context of label switching, we first define  $N_{ij}$  as the number of time steps where the true label is state  $i$  and the predicted state is  $j$ , and then set  $C_{ij} = -N_{ij}$  (since we want to maximize agreement).

### 30.4.2 Parameter estimation using SGD

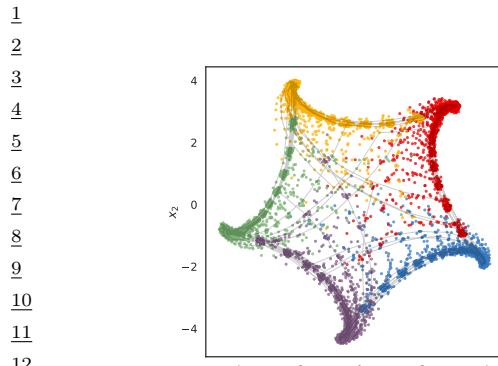
Although the EM algorithm is the “traditional” way to fit HMMs, it is inherently a batch algorithm, so it does not scale well to large datasets (with many sequences). Although it is possible to extend bound optimization to the online case (see e.g., [Mai15]), this can take a lot of memory.

A simple alternative is to optimize  $\log p(\mathbf{y}_{1:T}|\boldsymbol{\theta})$  using SGD. We can compute this objective using the forwards algorithm, as shown in Equation (8.37):

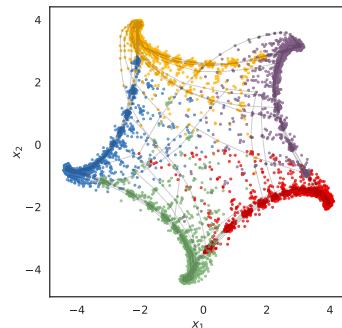
$$\log p(\mathbf{y}_{1:T}|\boldsymbol{\theta}) = \sum_{t=1}^T \log p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \boldsymbol{\theta}) = \sum_{t=1}^T \log Z_t \quad (30.29)$$

where the normalization constant for each time step is given by

$$Z_t \triangleq p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \sum_{j=1}^K p(z_t = j|\mathbf{y}_{1:t-1})p(\mathbf{y}_t|z_t = j) \quad (30.30)$$



(a)



(b)

Figure 30.14: (a) Samples from the true AR-HMM. (b) Samples from the learned AR-HMM.

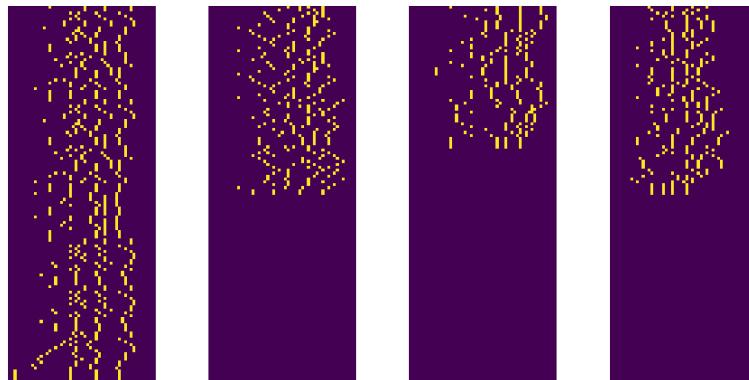


Figure 30.15: First 4 training sequences from the Bach Chorales dataset. Vertical axis represents time (start of music at top), horizontal axis represents the 51 notes. Generated by [hmm\\_bach\\_chorales.ipynb](#).

Of course, we need to ensure the transition matrix remains a valid row stochastic matrix, i.e., that  $0 \leq A_{ij} \leq 1$  and  $\sum_j A_{ij} = 1$ . Similarly, if we have categorical observations, we need to ensure  $B_{jk}$  is a valid row stochastic matrix, and if we have Gaussian observations, we need to ensure  $\Sigma_k$  is a valid psd matrix. These constraints are automatically taken care of in EM. When using SGD, we can reparameterize to an unconstrained form, as proposed in [BC94].

#### 30.4.2.1 Example: Bach Chorales

In this section, we give an example of fitting an HMM with SGD to some music data from [BLBV12]. The training set consists of 229 sequences of Chorales composed by J. S. Bach. Each sequence has a

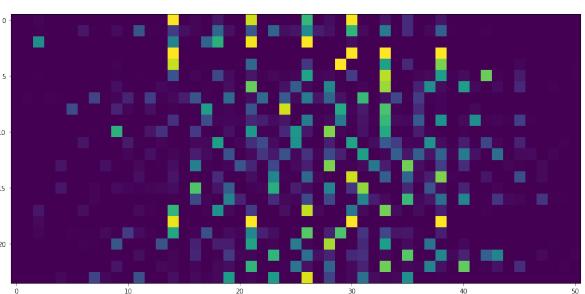


Figure 30.16: Learned observation distributions for the 51 notes from a 24 state HMM. Generated by `hmm_bach_chorales.ipynb`.

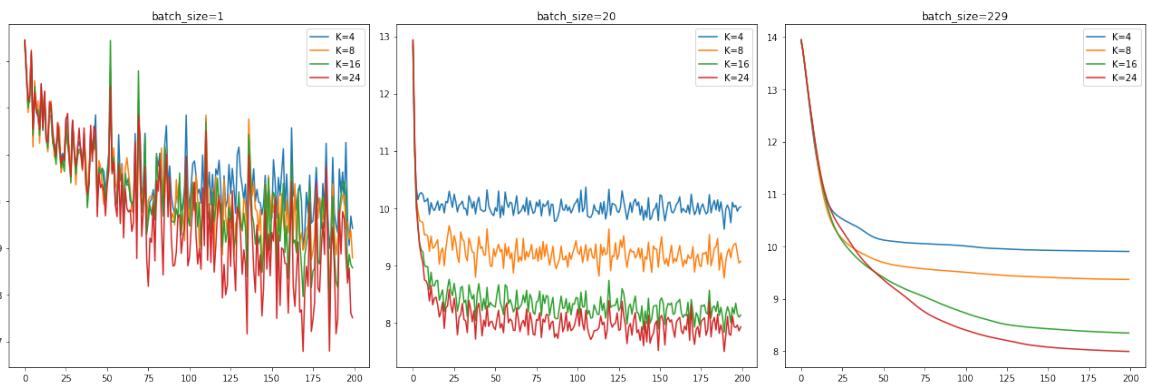


Figure 30.17: Negative log likelihood vs iterations on the Bach Chorales dataset for different number of hidden states  $K$ , and batch sizes of  $B = 1$  (left),  $B = 20$  (middle) and  $B = N = 229$  (right). Generated by `hmm_bach_chorales.ipynb`.

maximum length of 129, and contains 51 notes.<sup>3</sup> Thus the data matrix is a boolean tensor of size  $N \times T_{\max} \times D = 229 \times 129 \times 51$ , although the actual sequence lengths are variable, so the data is stored in a ragged array. See Figure 30.15 for a visualization of some of the data.

We model the observation at each time step using a factored observation distribution of the form

$$p(\mathbf{y}_t | z_t = k) = \prod_{d=1}^D \text{Ber}(y_{td} | B_d(k)) \quad (30.31)$$

where  $B_d(k) = p(y_{td} = 1 | z_t = k)$ . We then fit an HMM with  $K \in \{4, 8, 16, 24\}$  states to this using SGD. In Figure 30.16 we show the learned observation model for  $K = 24$  (which was the model with the best test set loglikelihood). We see that some states generate multiple notes at once, presumably corresponding to chords.

<sup>3</sup> We follow the same preprocessing as used in <https://pyro.ai/examples/hmm.html>, where they drop 37 notes that are never played.

In Figure 30.17, we show how the negative log likelihood on the training set is reduced across SGD iterations. Obviously the training loss is lower for larger  $K$ . In addition, the loss curve is noisier for small batch sizes. The test set negative log likelihood using  $K = 24$  is 8.05 nats per time step, which is comparable to the value of 8.28 reported in Table 1 of [Obe+19] using an HMM with  $K = 16$ .

### 30.4.3 Parameter estimation using spectral methods

Fitting HMMs using maximum likelihood is difficult, because the log likelihood is not convex. Thus there are many local optima, and EM and SGD can give poor results. An alternative approach is to marginalize out the hidden variables, and work instead with predictive distributions in the visible space. For discrete observation HMMs, with observation matrix  $B_{jk} = p(y_t = k|z_t = j)$ , such a distribution has the form

$$[\mathbf{y}_t]_k \triangleq p(y_t = k|\mathbf{y}_{1:t-1}) \quad (30.32)$$

This is called a **predictive state representation** [SJR04].

Suppose there are  $m$  possible hidden states, and  $n$  possible visible symbols, where  $n \geq m$ . One can show [HKZ12; Joh12] that the PSR vectors lie in a subspace in  $\mathbb{R}^n$  with a dimensionality of  $m \leq n$ . Intuitively this is because the linear operator  $\mathbf{A}$  defining the hidden state update in Equation (8.36), combined with the mapping to observables via  $\mathbf{B}$ , induces low rank structure in the output space. Furthermore, we can estimate a basis for this low rank subspace using SVD applied to the observable matrix of co-occurrence counts:

$$[\mathbf{P}_2]_{ij} = p(y_t = i, y_{t-1} = j) \quad (30.33)$$

We also need to estimate the third order statistics

$$[\mathbf{P}_3]_{ijk} = p(y_t = i, y_{t-1} = j, y_{t-2} = k) \quad (30.34)$$

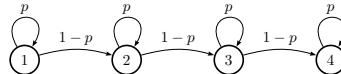
Using these quantities, it is possible to perform recursive updating of our predictions while working entirely in visible space. This is called **spectral estimation**, or **tensor decomposition** [HKZ12; AHK12; Rod14; Ana+14; RSG17].

We can use spectral methods to get a good initial estimate of the parameters for the latent variable model, which can then be refined using EM (see e.g., [Smi+00]). Alternatively, we can use them “as is”, without needing EM at all. See [Mat14] for a comparison of these methods. See also Section 31.2.5.3 where we discuss spectral methods for fitting linear dynamical systems.

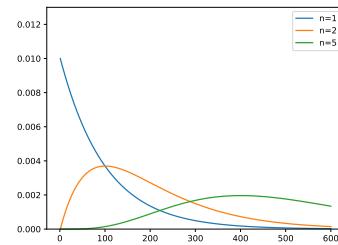
### 30.4.4 Bayesian parameter inference

MLE methods can easily overfit, and can suffer from numerical problems, especially when sample sizes are small. In [Fot+14], they show how to apply stochastic variational inference (Section 10.3.2) to HMMs, by leveraging the conjugate structure. An alternative approach is to marginalize out the discrete latents (using the forwards algorithm), and then to use SVI to target the log posterior

$$\log p(\boldsymbol{\theta}, \mathcal{D}) = \log p(\boldsymbol{\theta}) + \sum_{n=1}^N \log p(\mathbf{y}_{1:T,n}|\boldsymbol{\theta}) \quad (30.35)$$



(a)



(b)

Figure 30.18: (a) A Markov chain with  $n = 4$  repeated states and self loops. (b) The resulting distribution over sequence lengths, for  $p = 0.99$  and various  $n$ . Generated by `hmm_self_loop_dist.py`.

This can be thought of as “collapsed SVT”, and was proposed in [Obe+19].

We can also apply HMC (Section 12.5) to the same log joint. This is a form of “collapsed MCMC”. This is generally faster than the older technique of block Gibbs sampling [Sco02], that alternates between sampling latent sequences  $\mathbf{z}_{1:T,1:N}^s$  using the forwards filtering backwards sampling algorithm (Section 8.3.7) and sampling the parameters from their full conditionals,  $p(\boldsymbol{\theta}|\mathbf{y}_{1:T}, \mathbf{z}_{1:T,1:N}^s)$ , since the high correlation between  $\mathbf{z}$  and  $\boldsymbol{\theta}$  makes this coordinate-wise approach rather slow.

## 30.5 HMMS: Generalizations

In this section, we discuss various extensions of the vanilla HMM introduced in Section 30.1.

### 30.5.1 Hidden semi-Markov model (HSMM)

In a standard HMM (Section 30.1), the probability we remain in state  $i$  for exactly  $d$  steps is

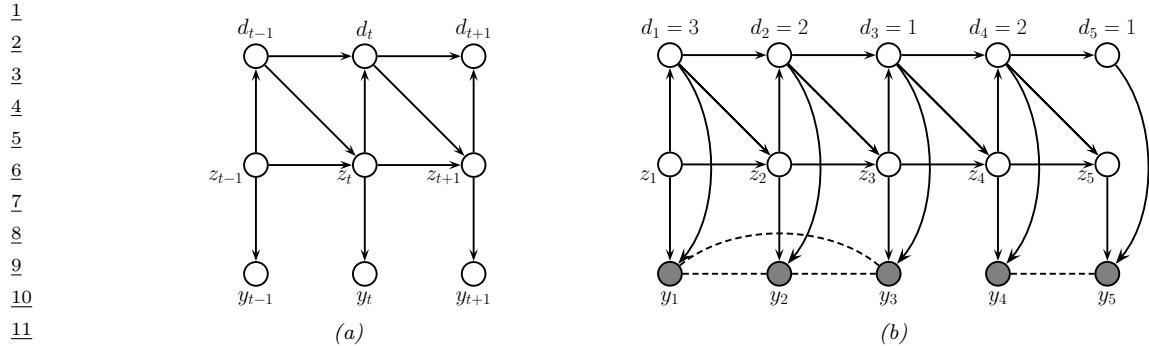
$$p(d_i = d) = (1 - A_{ii})A_{ii}^d \propto \exp(d \log A_{ii}) \quad (30.36)$$

where  $A_{ii}$  is the self-loop probability. This is called the **geometric distribution**. However, this kind of exponentially decaying function of  $d$  is sometimes unrealistic.

A simple way to model non-geometric waiting times is to replace each state with  $n$  new states, each with the same emission probabilities as the original state. For example, consider the model in Figure 30.18(a). Obviously the smallest sequence this can generate is of length  $n = 4$ . Any path of length  $d$  through the model has probability  $p^{d-n}(1-p)^n$ ; multiplying by the number of possible paths we find that the total probability of a path of length  $d$  is

$$p(d) = \binom{d-1}{n-1} p^{d-n} (1-p)^n \quad (30.37)$$

This is equivalent to the negative binomial distribution. By adjusting  $n$  and the self-loop probabilities  $p$  of each state, we can model a wide range of waiting times: see Figure 30.18(b).



*Figure 30.19: Encoding a hidden semi-Markov model as a PGM-D. (a)  $d_t$  is a deterministic down counter (duration variable). Each observation is generated independently. (b) Similar to (a), except now we generate the observations within each segment as a block. In this figure, we represent the non-Markovian dependencies between the observations within each segment by using undirected edges. We represent the conditional independence between the observations across different segments by disconnecting  $y_{1:3}$  from  $y_{4:5}$ ; this can be enforced by ‘breaking the link’ whenever  $d_t = 1$  (representing the end of a segment).*

A more general solution is to use a **semi-Markov model**, in which the next state not only depends on the previous state, but also on how long we’ve been in that state. When the state space is not observed directly, the result is called a **hidden semi-Markov model (HSMM)**, a **variable duration HMM**, or an **explicit duration HMM** [Yu10].

One way to represent a HSMM is to use the graphical model shown in Figure 30.19(a). The  $d_t \in \{1, \dots, D\}$  node is a state duration counter, where  $D$  is the maximum duration of any state. When we first enter state  $j$ , we sample  $d_t$  from the duration distribution for that state,  $d_t \sim p_j(\cdot)$ . Thereafter,  $d_t$  deterministically counts down until  $d_t = 1$ . More precisely, we define the following CPD:

$$p(d_t = d' | d_{t-1} = d, z_t = j) = \begin{cases} D_j(d') & \text{if } d = 1 \\ 1 & \text{if } d' = d - 1 \text{ and } d > 0 \\ 0 & \text{otherwise} \end{cases} \quad (30.38)$$

Note that  $D_j(d)$  could be represented as a table (a non-parametric approach) or as some kind of parametric distribution, such as a Gamma distribution. If  $D_j(d)$  is a geometric distribution, this emulates a standard HMM.

While  $d_t > 1$ , the state  $z_t$  is not allowed to change. When  $d_t = 1$ , we make a stochastic transition to a new state. More precisely, we define the state CPD as follows:

$$p(z_t = k | z_{t-1} = j, d_{t-1} = d) = \begin{cases} 1 & \text{if } d > 0 \text{ and } j = k \\ A_{jk} & \text{if } d = 1 \\ 0 & \text{otherwise} \end{cases} \quad (30.39)$$

This ensures that the model stays in the same state for the entire duration of the segment. At each step within this segment, an observation is generated.

HSMMs are useful not only because they can model the duration of each state explicitly, but also because they can model the distribution of a whole subsequence of observations at once, instead

of assuming all observations are generated independently at each time step. That is, they can use likelihood models of the form  $p(\mathbf{y}_{t:t+l}|z_t = k, d_t = l)$ , which generate  $l$  correlated observations if the duration in state  $k$  is for  $l$  time steps. This approach, known as a **segmental HMM**, is useful for modeling data that is piecewise linear, or shows other local trends [ODK96]. We can also use an RNN to model each segment, resulting in an **RNN-HSMM** model [Dai+17].

More precisely, we can define a segmental HMM as follows:

$$p(\mathbf{y}, \mathbf{z}, \mathbf{d}) = \left[ p(z_1)p(d_1|z_1) \prod_{t=2}^T p(z_t|z_{t-1}, d_{t-1})p(d_t|z_t, d_{t-1}) \right] p(\mathbf{y}|\mathbf{z}, \mathbf{d}) \quad (30.40)$$

In a standard HSMM, we assume

$$p(\mathbf{y}|\mathbf{z}, \mathbf{d}) = \prod_{t=1}^T p(y_t|z_t) \quad (30.41)$$

so the duration variables only determine the hidden state dynamics. To define  $p(\mathbf{y}|\mathbf{z}, \mathbf{d})$  for a segmental HMM, let us use  $s_i$  and  $e_i$  to denote the start and end times of segment  $i$ . This sequence can be computed deterministically from  $\mathbf{d}$  using  $s_1 = 1$ ,  $s_i = s_{i-1} + d_{s_{i-1}}$ , and  $e_i = s_i + d_{s_i} - 1$ . We now define the observation model as follows:

$$p(\mathbf{y}|\mathbf{z}, \mathbf{d}) = \prod_{i=1}^{|\mathbf{s}|} p(\mathbf{y}_{s_i:e_i}|z_{s_i}, d_{s_i}) \quad (30.42)$$

See Figure 30.19(b) for the PGM-D.

If we use an RNN for each segment, we have

$$p(\mathbf{y}_{s_i:e_i}|z_{s_i}, d_{s_i}) = \prod_{t=s_i}^{e_i} p(y_t|\mathbf{y}_{s_i:t-1}, z_{s_i}) = \prod_{t=s_i}^{e_i} p(y_t|h_t, z_{s_i}) \quad (30.43)$$

where  $h_t$  is the hidden state that is deterministically updated given the previous observations in this sequence.

As shown in [Chi14], it is possible to compute  $p(z_t, d_t|\mathbf{y}_{1:T})$  in  $O(TK^2 + TKD)$  time, where  $T$  is the sequence length,  $K$  is the number of states, and  $D$  is the maximum duration of any segment. In [Dai+17], they show how to train an approximate inference algorithm, based on a mean field approximation  $q(\mathbf{z}, \mathbf{d}|\mathbf{y}) = \prod_t q(z_t|\mathbf{y})q(d_t|\mathbf{y})$ , to compute the posterior in  $O(TK + TD)$  time.

### 30.5.2 HSMMs for changepoint detection

In this section, we discuss how to use HSMM-like models for the problem of **changepoint detection**, which is the task of detecting when there are “abrupt” changes in the distribution of the observed values in a time series. For a review of offline methods to this problem, see e.g., [AC17; TOV18]. (See also [BW20] for a recent empirical evaluation of various methods, focused on the 1d time series case.)

In this section, we focus on the online case. The methods we discuss can (in principle) be used for high-dimensional time series segmentation. Our starting point is the hidden semi-Markov models (HSMM) discussed in Section 30.5.1. This is like an HMM in which we explicitly model the duration

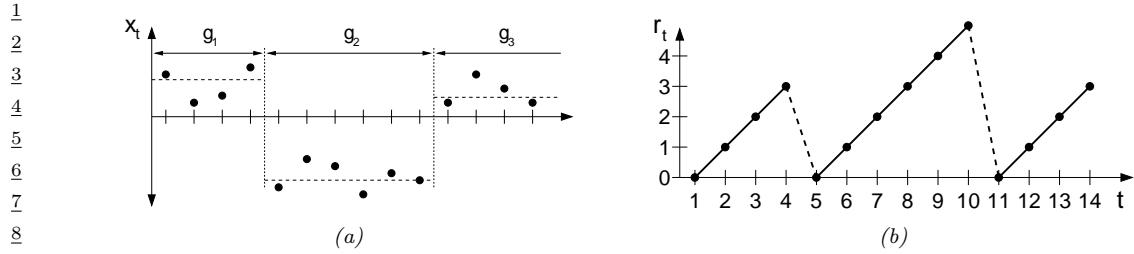


Figure 30.20: Illustration of Bayesian online changepoint detection (BOCPD). (a) Hypothetical segmentation of a univariate time series divided by changepoints on the mean into three segments of lengths  $g_1 = 4$ ,  $g_2 = 6$ , and an undetermined length for  $g_3$  (since it the third segment has not yet ended). From Figure 1 of [AM07]. Used with kind permission of Ryan Adams.

spent in each state. This is done by augmenting the latent state  $z_t$  with a duration variable  $d_t$  which is initialized according to a duration distribution,  $d_t \sim D_{z_t}(\cdot)$ , and which then *counts down* to 0. An alternative approach is to add a variable  $r_t \{0, 1, \dots\}$  which encodes the **run length** for the current state; this starts at 0 whenever a new segment is created, and then *counts up* by one at each step. The transition dynamics is specified by

$$p(r_t|r_{t-1}) = \begin{cases} H(r_{t-1} + 1) & \text{if } r_t = 0 \\ 1 - H(r_{t-1} + 1) & \text{if } r_t = r_{t-1} + 1 \\ 0 & \text{otherwise} \end{cases} \quad (30.44)$$

where  $H(\tau)$  is a **hazard function**:

$$H(\tau) = \frac{p_g(\tau)}{\sum_{t=\tau}^{\infty} p_g(t)} \quad (30.45)$$

where  $p_g(t)$  is the probability of a gap of length  $t$ . See Figure 30.20 for an illustration. If we set  $p_g$  to be a geometric distribution with parameter  $\lambda$ , then the hazard function is the constant  $H(\tau) = 1/\lambda$ .

The advantage of the run-length representation is that we can define the observation model for a segment in a causal way (that only depends on past data):

$$p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, r_t = r, z_t = k) = p(\mathbf{y}_t|\mathbf{y}_{t-r:t-1}, z_t = k) = \int p(\mathbf{y}_t|\boldsymbol{\eta})p(\boldsymbol{\eta}|\mathbf{y}_{t-r:t-1}, z_t = k)d\boldsymbol{\eta} \quad (30.46)$$

where  $\boldsymbol{\eta}$  are the parameters that are “local” to this segment. This called the **underlying predictive model** or **UPM** for the segment. The posterior over the UPM parameters is given by

$$p(\boldsymbol{\eta}|\mathbf{y}_{t-r:t-1}, z_t = k) \propto p(\boldsymbol{\eta}|z_t = k) \prod_{i=t-r}^{t-1} p(\mathbf{y}_i|\boldsymbol{\eta}) \quad (30.47)$$

where we initialize the prior for  $\boldsymbol{\eta}$  using hyper-parameters chosen by state  $k$ . If the model is conjugate exponential, we can compute this marginal likelihood in closed form, and we have

$$\pi_t^{r,k} = p(\mathbf{y}_t|\mathbf{y}_{t-r:t-1}, z_t = k) = p(\mathbf{y}_t|\psi_t^{r,k}) \quad (30.48)$$

where  $\psi_t^{r,k}$  are the parameters of the posterior predictive distribution at time  $t$  based on the last  $r$  observations (and using a prior from state  $k$ ).

In the special case in which we have  $K = 1$  hidden states, then each segment is modeled independently, and we get a **product partition model** [BH92]:

$$p(\mathbf{y}|\mathbf{r}) = p(\mathbf{y}_{s_1:e_1}) \dots p(\mathbf{y}_{s_N:e_N}) \quad (30.49)$$

where  $s_i$  and  $e_i$  are the start and end of segment  $i$ , which can be computed from the run lengths  $\mathbf{r}$ . (We initialize with  $r_0 = 0$ .) Thus there is no information sharing between segments. This can be useful for timeseries in which there are abrupt changes, and where the new parameters are unrelated to the old ones.

Detecting the locations of these changes is called **changepoint detection**. An exact online algorithm for solving this task was proposed in [FL07] and independently in [AM07]; in the latter paper, they call the method **Bayesian online changepoint detection** or **BOCPD**. We can compute a posterior over the current run length recursively as follows:

$$p(r_t|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{y}_{1:t-}, r_t)p(r_t|\mathbf{y}_{1:t-1}) \quad (30.50)$$

where we initialize with  $p(r_0 = 0) = 1$ . The marginal likelihood  $p(\mathbf{y}_t|\mathbf{y}_{1:t-}, r_t)$  is given by Equation (30.46) (with  $z_t = 1$  dropped, since there is just one state). The prior predictive is given by

$$p(r_t|\mathbf{y}_{1:t-1}) = \sum_{r_{t-1}} p(r_t|r_{t-1})p(r_{t-1}|\mathbf{y}_{1:t-1}) \quad (30.51)$$

The one step ahead predictive distribution is given by

$$p(\mathbf{y}_{t+1}|\mathbf{y}_{1:t}) = \sum_{r_t} p(\mathbf{y}_{t+1}|\mathbf{y}_{1:t}, r_t)p(r_t|\mathbf{y}_{1:t}) \quad (30.52)$$

### 30.5.2.1 Example

See Figure 30.21 for an illustration of this method applied to a 1d **well-log dataset** from [RF96; FC03]. The likelihood is a univariate Gaussian,  $p(y_t|\mu) = \mathcal{N}(y_t|\mu, \sigma^2)$ , where  $\sigma^2 = 4000^2$  is fixed, and  $\mu$  is inferred using a Gaussian prior  $p(\mu) = \mathcal{N}(\mu|\mu_0 = 1.15 \times 10^5, \sigma_0^2 = 1 \times 10^8)$ . The hazard function is set to a geometric distribution,  $H(r_t) = 1/\lambda$  where  $\lambda = 250$ . We see that sudden changes in the mean are detected, and then the run length resets to 0. The posterior predictive distribution at the start of each segment is broad, but rapidly concentrates as more data is collected. Despite the simplicity of this method, it was shown to be one of the best performing approaches to changepoint detection in an extensive recent benchmark [BW20].

### 30.5.2.2 Extensions

Although the above method is exact, each update step takes  $O(t)$  time, so the total cost of the algorithm is  $O(T^2)$ . We can reduce this by pruning out states with low probability. In particular, we can use particle filtering (Section 13.2) with  $N$  particles, together with a stratified optimal resampling method, to reduce the cost to  $O(TN)$ . See [FL07] for details.

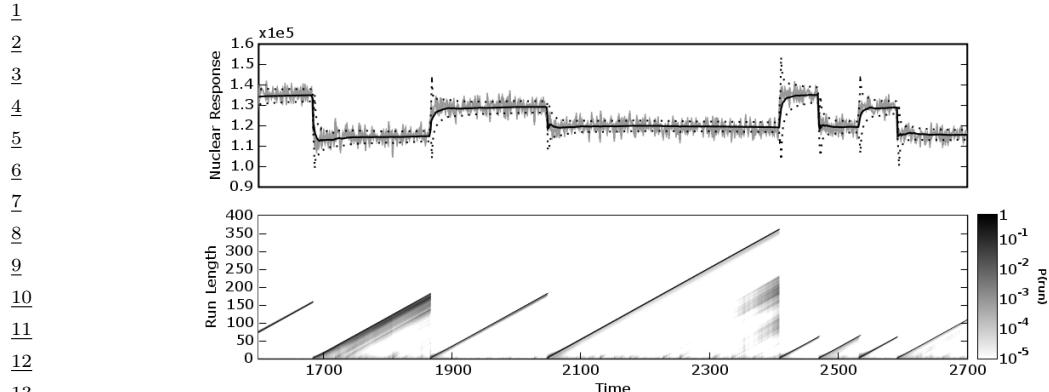


Figure 30.21: Illustration of BOCPD applied to the well-log dataset. Top row: Nuclear magnetic response during the drilling of a well. The data are plotted in light gray, the dark line shows the posterior predictive mean, and the dashed lines the  $1\sigma$  error bars. Bottom row: Posterior probability over run lengths,  $p(r_t|y_{1:t})$ , using a logarithmic scale. Darker colors indicate higher probability. From Figure 2 of [AM07]. Used with kind permission of Ryan Adams.

19  
20  
21  
22

23 In addition, the above method relies on a conjugate exponential model in order to compute the  
24 marginal likelihood, and update the posterior parameters for each  $r_t$  in  $O(1)$  time. For more complex  
25 models, we need to use approximations. In [TBS13], they use variational Bayes (Section 10.2.3), and  
26 in [Mav16], they use particle filtering (Section 13.2), which is more general, but much slower.  
27

It is possible to extend the model in various other ways. In [FL11], they allow for Markov  
dependence between the parameters of neighboring segments. In [STR10], they use a Gaussian  
process (Chapter 18) to represent the UPM, which captures correlations between observations within  
the same segment. In [KJD18], they use generalized Bayesian inference (Section 14.1.3) to create a  
method that is more robust to model misspecification.

In [Gol+17], they extend the model by modeling the probability of a sequence of observations,  
rather than having to make the decision about whether to insert a changepoint or not based on just  
on the likelihood ratio of a single time step.

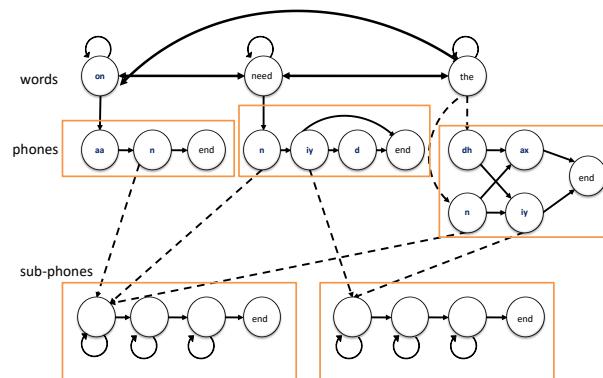
In [AE+20], they extend the model by allowing for multiple discrete states, as in an HSMM. In  
addition, they add both the run length  $r_t$  and the duration  $d_t$  to the state space. This allows the  
method to specify not just when the current segment started, but also when it is expected to end.  
In addition, it allows the UPM to depend on the duration of the segment, and not just on past  
observations. For example, we can use

$$p(y_t|r_t, d_t, \eta) = \mathcal{N}(y_t | \phi(r_t/d_t)^\top \boldsymbol{\eta}, \sigma^2) \quad (30.53)$$

where  $0 \leq r_t/d_t \leq 1$ , and  $\phi()$  is a set of learned basis functions. This allows observation sequences  
for the same hidden state to have a common functional shape, even if the time spent in each state is  
different.

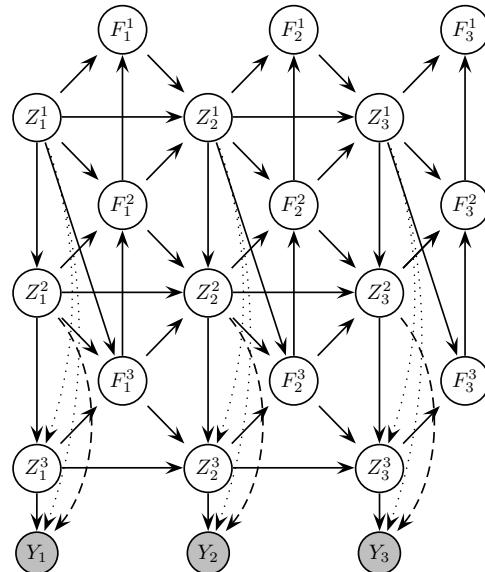
47

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14



15  
16 Figure 30.22: An example of an HHMM for an ASR system which can recognize 3 words. The top level  
17 represents bigram word probabilities. The middle level represents the phonetic spelling of each word. The  
18 bottom level represents the subphones of each phone. (It is traditional to represent a phone as a 3 state HMM,  
19 representing the beginning, middle and end; these are known as subphones.) Adapted from Figure 7.5 of  
[JM00].  
20

21  
22



42 Figure 30.23: An HHMM represented as a PGM-D.  $Z_t^\ell$  is the state at time  $t$ , level  $\ell$ ;  $F_t^\ell = 1$  if the HMM at  
43 level  $\ell$  has finished (entered its exit state), otherwise  $F_t^\ell = 0$ . Shaded nodes are observed; the remaining nodes  
44 are hidden. We may optionally clamp  $F_T^\ell = 1$ , where  $T$  is the length of the observation sequence, to ensure  
45 all models have finished by the end of the sequence. From Figure 2 of [MP01].  
46  
47

1 **30.5.3 Hierarchical HMMs**

3 A **hierarchical HMM** (HHMM) [FST98] is an extension of the HMM that is designed to model  
4 domains with hierarchical structure. Figure 30.22 gives an example of an HHMM used in automatic  
5 speech recognition, where words are composed of phones which are composed of subphones. We  
6 can always “flatten” an HHMM to a regular HMM, but a factored representation is often easier to  
7 interpret, and allows for more efficient inference and model fitting.

8 HHMMs have been used in many application domains, e.g., speech recognition [Bil01], gene finding  
9 [Hu+00], plan recognition [BVW02], monitoring transportation patterns [Lia+07], indoor robot  
10 localization [TMK04], etc. HHMMs are less expressive than stochastic context free grammars (SCFGs)  
11 since they only allow hierarchies of bounded depth, but they support more efficient inference. In  
12 particular, inference in SCFGs (using the inside outside algorithm, [JM08]) takes  $O(T^3)$  whereas  
13 inference in an HHMM takes  $O(T)$  time [MP01; WM12].

14 We can represent an HHMM as a directed graphical model as shown in Figure 30.23.  $Q_t^\ell$  represents  
15 the state at time  $t$  and level  $\ell$ . A state transition at level  $\ell$  is only “allowed” if the chain at the level  
16 below has “finished”, as determined by the  $F_t^{\ell-1}$  node. (The chain below finishes when it chooses to  
17 enter its end state.) This mechanism ensures that higher level chains evolve more slowly than lower  
18 level chains, i.e., lower levels are nested within higher levels.

19 A variable duration HMM can be thought of as a special case of an HHMM, where the top level is  
20 a deterministic counter, and the bottom level is a regular HMM, which can only change states once  
21 the counter has “timed out”. See [MP01] for further details.

24 **30.5.4 Factorial HMMs**

25 An HMM represents the hidden state using a single discrete random variable  $z_t \in \{1, \dots, K\}$ . To  
26 represent 10 bits of information would require  $K = 2^{10} = 1024$  states. By contrast, consider a  
27 **distributed representation** of the hidden state, where each  $z_{t,m} \in \{0, 1\}$  represents the  $m$ 'th bit  
28 of the  $t$ 'th hidden state. Now we can represent 10 bits using just 10 binary variables. This model is  
29 called a **factorial HMM** [GJ97].

31 More precisely, the model is defined as follows:

$$\begin{aligned} \text{32} \\ \text{33} \quad p(\mathbf{z}, \mathbf{y}) &= \prod_t \left[ \prod_m p(z_{tm} | z_{t-1,m}) \right] p(\mathbf{y}_t | \mathbf{z}_t) \end{aligned} \tag{30.54}$$

36 where  $p(z_{tm} = k | z_{t-1,m} = j) = A_{mj}$  is an entry in the transition matrix for chain  $m$ ,  $p(z_{1m} =$   
37  $k | z_{0m}) = p(z_{1m} = k) = \pi_{mk}$ , is the initial state distribution for chain  $m$ , and

$$\begin{aligned} \text{39} \\ \text{40} \quad p(\mathbf{y}_t | \mathbf{z}_t) &= \mathcal{N} \left( \mathbf{y}_t | \sum_{m=1}^M \mathbf{W}_m z_{tm}, \boldsymbol{\Sigma} \right) \end{aligned} \tag{30.55}$$

43 is the observation model, where  $\mathbf{z}_{tm}$  is a 1-of- $K$  encoding of  $z_{tm}$  and  $\mathbf{W}_m$  is a  $D \times K$  matrix (assuming  
44  $\mathbf{y}_t \in \mathbb{R}^D$ ). Figure 30.24(a) illustrates the model for the case where  $M = 3$ .

45 An interesting application of FHMMs is to the problem of **energy disaggregation** [KJ12a].  
46 In this problem, we observe the total energy usage of a house at each moment in time, i.e., the  
47

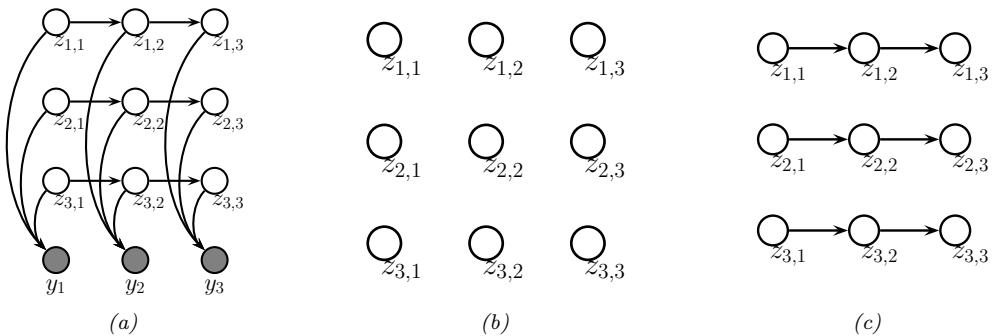


Figure 30.24: (a) A factorial HMM with 3 chains. (b) A fully factorized approximation. (c) A product-of-chains approximation. Adapted from Figure 2 of [GJ97].

observation model has the form

$$p(y_t | \mathbf{z}_t) = \mathcal{N}(y_t | \sum_{m=1}^M w_m z_{tm}, \sigma^2) \quad (30.56)$$

where  $w_m$  is the amount of energy used by device  $m$ , and  $z_{tm} = 1$  if device  $m$  is being used at time  $t$  and  $z_{tm} = 0$  otherwise. The transition model is assumed to be

$$p(z_{t,m} = 1 | z_{t-1,m}) = \begin{cases} A_{01} & \text{if } z_{t-1,m} = 0 \\ A_{11} & \text{if } z_{t-1,m} = 1 \end{cases} \quad (30.57)$$

We do not know which devices are turned on at each time step (i.e., the  $z_{tm}$  are hidden), but by applying inference in the FHMM over time, we can separate the total energy into its parts, and thereby determine which devices are using the most electricity.

#### 30.5.4.1 Structured mean field for FHMMs

Even though each chain in an FHMM is a priori independent, they become coupled in the posterior due to having an observed common child,  $\mathbf{y}_t$ . Exact inference for this model therefore takes  $O(TM K^{M+1})$  time. In this section, we derive a structured mean field algorithm, from [GJ97], that takes  $O(TMK^2I)$  time, where  $I$  is the number of mean field iterations (typically  $I \sim 10$  suffices for good performance).

We can write the exact posterior in the following form:

$$p(\mathbf{z} | \mathbf{y}) = \frac{1}{Z(\mathbf{y})} \exp(-\mathcal{E}(\mathbf{z}, \mathbf{y})) \quad (30.58)$$

$$\begin{aligned} \mathcal{E}(\mathbf{z}, \mathbf{y}) &= \frac{1}{2} \sum_{t=1}^T \left( \mathbf{y}_t - \sum_m \mathbf{W}_m z_{tm} \right)^T \Sigma^{-1} \left( \mathbf{y}_t - \sum_m \mathbf{W}_m z_{tm} \right) \\ &\quad - \sum_{m=1}^M \mathbf{z}_{1m}^\top \tilde{\pi}_m - \sum_{t=2}^T \sum_{m=1}^M \mathbf{z}_{t-1,m}^\top \tilde{\mathbf{A}}_m \mathbf{z}_{t,m} \end{aligned} \quad (30.59)$$

1 where  $\tilde{\mathbf{A}}_m \triangleq \log \mathbf{A}_m$  and  $\tilde{\boldsymbol{\pi}}_m \triangleq \log \boldsymbol{\pi}_m$ , where the log is applied elementwise.  
 2

3 We can approximate the posterior as a product of marginals, as in Figure 30.24(b), but a better  
 4 approximation is to use a product of chains, as in Figure 30.24(c). Each chain can be tractably  
 5 updated individually, using the forwards-backwards algorithm (Section 8.3.3). More precisely, we  
 6 assume

$$7 \quad q(\mathbf{z}|\mathbf{y}) = \frac{1}{Z_q} \prod_{m=1}^M q(z_{1m}|\boldsymbol{\psi}_{1m}) \prod_{t=2}^T q(z_{tm}|z_{t-1,m}, \boldsymbol{\psi}_{tm}) \quad (30.60)$$

$$8 \quad q(z_{1m}|\boldsymbol{\psi}_{1m}) = \prod_{k=1}^K (\xi_{1mk} \pi_{mk})^{z_{1mk}} \quad (30.61)$$

$$9 \quad q(z_{tm}|z_{t-1,m}, \boldsymbol{\psi}_{tm}) = \prod_{k=1}^K \left( \xi_{tmk} \prod_{j=1}^K (A_{mj} z_{t-1,m,j})^{z_{tmk}} \right)^{z_{tmk}} \quad (30.62)$$

10 Here the variational parameter  $\xi_{tmk}$  plays the role of an approximate local evidence, averaging out  
 11 the effects of the other chains. This is in contrast to the exact local evidence, which couples all the  
 12 chains together.

13 By separating out the approximate local evidence terms, we can rewrite the above as  $q(\mathbf{z}|\mathbf{y}) =$   
 14  $\frac{1}{Z_q(\mathbf{y})} \exp(-\mathcal{E}_q(\mathbf{z}, \mathbf{y}))$ , where  
 15

$$16 \quad \mathcal{E}_q(\mathbf{z}, \mathbf{y}) = - \sum_{t=1}^T \sum_{m=1}^M \mathbf{z}_{tm}^\top \tilde{\boldsymbol{\psi}}_{tm} - \sum_{m=1}^M \mathbf{z}_{1m}^\top \tilde{\boldsymbol{\pi}}_m - \sum_{t=2}^T \sum_{m=1}^M \mathbf{z}_{t-1,m}^\top \tilde{\mathbf{A}}_m \mathbf{z}_{t,m} \quad (30.63)$$

17 where  $\tilde{\boldsymbol{\psi}}_{tm} = \log \boldsymbol{\psi}_{tm}$ . We see that this has the same temporal factors as the exact log joint in  
 18 Equation (30.59), but the local evidence terms are different: the dependence on the visible data  $\mathbf{y}$   
 19 has been replaced by dependence on ‘‘virtual data’’  $\boldsymbol{\psi}$ .

20 The objective function is given by

$$21 \quad D_{\text{KL}}(q\|\tilde{p}) = \mathbb{E}_q [\log q - \log \tilde{p}] \quad (30.64)$$

$$22 \quad = -\mathbb{E}_q [\mathcal{E}_q(\mathbf{z}, \mathbf{y})] - \log Z_q(\mathbf{y}) + \mathbb{E}_q [\mathcal{E}(\mathbf{z}, \mathbf{y})] + \log Z(\mathbf{y}) \quad (30.65)$$

23 where  $q = q(\mathbf{z}|\mathbf{y})$  and  $\tilde{p} = p(\mathbf{z}|\mathbf{y})$ . One can show that we can optimize this using coordinate descent,  
 24 where each update step is given by

$$25 \quad \boldsymbol{\psi}_{tm} = \exp \left( \mathbf{W}_m^\top \boldsymbol{\Sigma}^{-1} \tilde{\mathbf{y}}_{tm} - \frac{1}{2} \boldsymbol{\delta}_m \right) \quad (30.66)$$

$$26 \quad \boldsymbol{\delta}_m \triangleq \text{diag}(\mathbf{W}_m^\top \boldsymbol{\Sigma}^{-1} \mathbf{W}_m) \quad (30.67)$$

$$27 \quad \tilde{\mathbf{y}}_{tm} \triangleq \mathbf{y}_t - \sum_{\ell \neq m}^M \mathbf{W}_\ell \mathbb{E}[\mathbf{z}_{t,\ell}] \quad (30.68)$$

28 The intuitive interpretation of  $\tilde{\mathbf{y}}_{tm}$  is that it is the observation  $\mathbf{y}_t$  minus the predicted effect from  
 29 all the other chains apart from  $m$ . This is then used to compute the approximate local evidence,  
 30

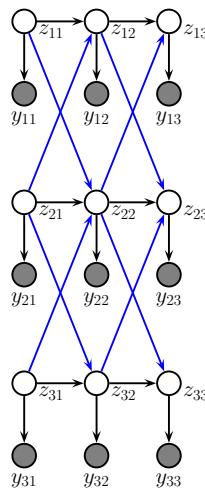


Figure 30.25: A coupled HMM with 3 chains.

$\mathbf{y}_{tm}$ . Having computed the  $\psi_{tm}$  terms for each chain, we can perform forwards-backwards in parallel, using these approximate local evidence terms to compute  $q(\mathbf{z}_{t,m}|\mathbf{y}_{1:T})$  for each  $m$  and  $t$ .

The update cost is  $O(TM^2)$  for a full “sweep” over all the variational parameters, since we have to run forwards-backwards  $M$  times, for each chain independently. This is the same cost as a fully factorized approximation, but is much more accurate.

### 30.5.5 Coupled HMMs

If we have multiple related data streams, we can use a **coupled HMM** [Bra96]. This is a series of HMMs where the state transitions depend on the states of neighboring chains. That is, we represent the conditional distribution for each time slice as

$$p(\mathbf{z}_t, \mathbf{y}_t | \mathbf{z}_{t-1}) = \prod_m p(\mathbf{y}_{tm} | z_{tm}) p(z_{tm} | \mathbf{z}_{t-1, m-1:m+1}) \quad (30.69)$$

with boundary conditions defined in the obvious way. See Figure 30.25 for an illustration with  $M = 3$  chains.

Coupled HMMs have been used for various tasks, such as **audio-visual speech recognition** [Nef+02], modeling freeway traffic flows [KM00], and modeling conversational interactions between people [Bas+01].

However, there are two drawbacks to this model. First, exact inference takes  $O(T(K^M)^2)$ , as in an factorial HMM; however, in practice this is not usually a problem, since  $M$  is often small. Second, the model requires  $O(MK^4)$  parameters to specify, if there are  $M$  chains with  $K$  states per chain, because each state depends on its own past plus the past of its two neighbors. There is a closely related model, known as the **influence model** [Asa00], which uses fewer parameters, by computing a convex combination of pairwise transition matrices.

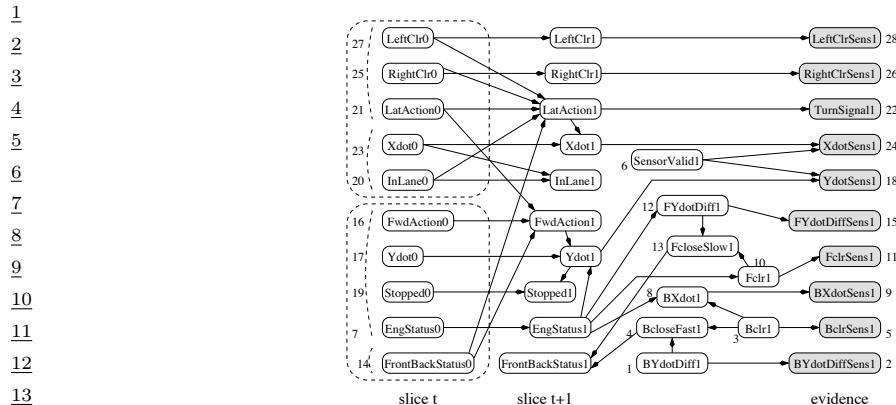


Figure 30.26: The BATnet DBN. The transient nodes are only shown for the second slice, to minimize clutter. The dotted lines are used to group related variables. Used with kind permission of Daphne Koller.

### 30.5.6 Dynamic Bayes nets (DBN)

A **dynamic Bayesian network (DBN)** is a way to represent a stochastic process using a directed graphical model [Mur02]. (Note that the network is not dynamic (the structure and parameters are fixed), rather it is a network representation of a dynamical system.) A DBN can be considered as a natural generalization of an HMM.

An example is shown in Figure 30.26, which is a DBN designed to monitor the state of a simulated autonomous car known as the “Bayesian Automated Taxi”, or “BATmobile” [For+95]. To define the model, you just need to specify the structure of the first time-slice, the structure between two time-slices, and the form of the CPDs. For details, see [KF09a].

28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47

# 31 State-space models

## 31.1 Introduction

A **state space model (SSM)** is a generalization of an HMM in which the hidden state vector, which evolves over time, is real-valued,  $\mathbf{z}_t \in \mathbb{R}^L$ . This generates some noisy observations, often real-valued,  $\mathbf{y}_t \in \mathbb{R}^D$ . The goal is to model the relationship between the hidden state trajectory,  $\mathbf{z}_{1:T}$ , and the observed data sequence,  $\mathbf{y}_{1:T}$ . We then use an inference algorithm to infer the hidden state,  $p(\mathbf{z}_{1:t} | \mathbf{y}_{1:t})$ , and/or predict future visible states,  $p(\mathbf{y}_{t+1:T} | \mathbf{y}_{1:t})$ . (We may also have optional observed inputs, such as **controls** or **actions**, denoted  $\mathbf{u}_{1:T}$ .)

We will focus on discrete-time SSMs, which can be written as follows:

$$\mathbf{z}_t = f(\mathbf{z}_{t-1}, \mathbf{u}_t, \boldsymbol{\epsilon}_t) \tag{31.1}$$

$$\mathbf{y}_t = h(\mathbf{z}_t, \mathbf{u}_t, \boldsymbol{\eta}_t) \tag{31.2}$$

where  $f$  is the dynamics model,  $h$  is the observation model, and  $\boldsymbol{\epsilon}_t$  and  $\boldsymbol{\eta}_t$  are the dynamics and observation noise. This defines the following joint distribution:

$$p(\mathbf{y}_{1:T}, \mathbf{z}_{1:T} | \mathbf{u}_{1:T}) = \left[ p(\mathbf{z}_1 | \mathbf{u}_1) \prod_{t=2}^T p(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{u}_t) \right] \left[ \prod_{t=1}^T p(\mathbf{y}_t | \mathbf{z}_t) \right] \tag{31.3}$$

where  $p(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{u}_t)$  is the transition model induced by  $f_z(\mathbf{z}_{t-1}, \mathbf{u}_t, \boldsymbol{\epsilon}_t)$  and  $p(\mathbf{y}_t | \mathbf{z}_t)$  is the observation model induced by  $f_x(\mathbf{z}_t, \boldsymbol{\eta}_t)$ . See Figure 31.1 for an illustration of the corresponding graphical model.

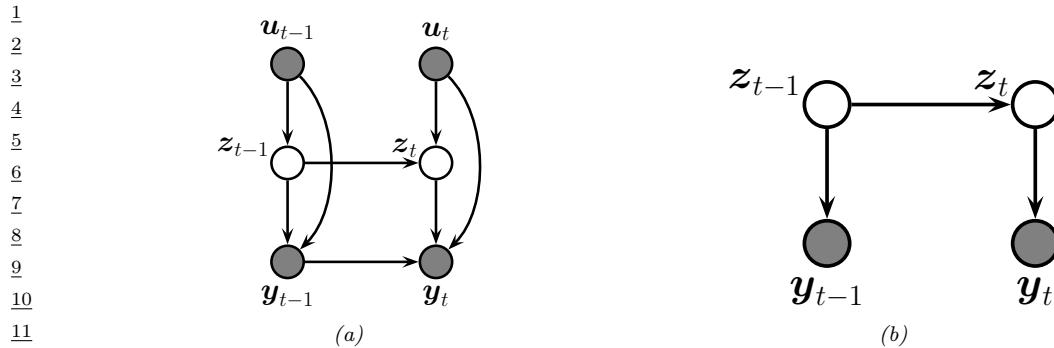
There are a variety of algorithms for performing inference in this model, depending on the assumptions we make about the forms of these distributions. For example, if everything is Gaussian, we can use Kalman filtering and its variants, as discussed in Chapter 8. For more general models, we can use variational inference (Chapter 10) or particle filtering (Section 13.2). We will illustrate some of these algorithms in the sections below, in the context of specific models. For more details, see e.g., [Sim06; Fra08; DK12; Sar13; PFW21; Tri21].

## 31.2 Linear dynamical systems

Consider the state space model in Equation (31.3). If we assume **additive Gaussian noise**, the model becomes

$$\mathbf{z}_t = f(\mathbf{z}_{t-1}, \mathbf{u}_t) + \boldsymbol{\epsilon}_t \tag{31.4}$$

$$\mathbf{y}_t = h(\mathbf{z}_t, \mathbf{u}_t) + \boldsymbol{\eta}_t \tag{31.5}$$



*Figure 31.1: State-space model represented as a graphical model. (a) Generic form, with inputs  $u_t$ , hidden state  $z_t$ , and observations  $y_t$ . We assume the observation likelihood is first-order auto-regressive. (b) Simplified form, with no inputs, and Markovian observations.*

where  $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$  and  $\eta_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$ . We will call these **Gaussian SSMs**.

An important special case of a Gaussian SSM arises when  $f$  and  $h$  are linear functions. This is known as a **linear-Gaussian state space model (LG-SSM)** or **linear dynamical system (LDS)**. In this case, we have

$$\mathbf{z}_t = \mathbf{A}_t \mathbf{z}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \epsilon_t \quad (31.6)$$

$$\mathbf{y}_t = \mathbf{C}_t \mathbf{z}_t + \mathbf{D}_t \mathbf{u}_t + \eta_t \quad (31.7)$$

Typically we assume the parameters  $\theta_t = (\mathbf{A}_t, \mathbf{C}_t, \mathbf{B}_t, \mathbf{D}_t, \mathbf{Q}_t, \mathbf{R}_t)$  are independent of time, so the model is stationary. (We discuss how to estimate the parameters in Section 31.2.5.) Given the parameters, we discuss how to perform online posterior inference of the latent states using the Kalman filter in Section 8.4.1, and offline inference using the Kalman smoother in Section 8.4.4. But first, we give some examples.

### 31.2.1 Example: Noiseless 1d spring-mass system

In this section, we consider an example from Wikipedia<sup>1</sup> of a **spring mass system** operating in 1d. Like many physical systems, this is best modeled in **continuous time**, although we will later discretize it.

Let  $x(t)$  be the position of an object which is attached by a spring to a wall, and let  $\dot{x}(t)$  and  $\ddot{x}(t)$  be its velocity and acceleration. By **Newton's laws of motion**, we have the following **ordinary differential equation**:

$$m\ddot{x}(t) = u(t) - b\dot{x}(t) - kx(t) \quad (31.8)$$

where  $u(t)$  is an externally applied force (e.g., someone tugging on the object),  $b$  is the viscous friction coefficient,  $k$  is the spring constant, and  $m$  is the mass of the object. See Figure 31.3 for the setup. We assume that we only observe the position, and not the velocity.

1. [https://en.wikipedia.org/wiki/State-space\\_representation#Moving\\_object\\_example](https://en.wikipedia.org/wiki/State-space_representation#Moving_object_example)

We now proceed to represent this as a first order Markov system. For simplicity, we ignore the noise ( $\mathbf{Q}_t = \mathbf{R}_t = \mathbf{0}$ ). We define the state space to contain the position and velocity,  $\mathbf{z}(t) = [x(t), \dot{x}(t)]$ . Thus the model becomes

$$\dot{\mathbf{z}}(t) = \mathbf{A}\mathbf{z}(t) + \mathbf{B}\mathbf{u}(t) \quad (31.9)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{z}(t) + \mathbf{D}\mathbf{u}(t) \quad (31.10)$$

where

$$\begin{pmatrix} \dot{x}(t) \\ \ddot{x}(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{pmatrix} \begin{pmatrix} x(t) \\ \dot{x}(t) \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} u(t) \quad (31.11)$$

$$y(t) = (1 \ 0) \begin{pmatrix} x(t) \\ \dot{x}(t) \end{pmatrix} \quad (31.12)$$

To simulate from this system, we need to evaluate the state at a set of discrete time intervals,  $t_k = k\Delta$ , where  $\Delta$  is the sampling rate or step size. There are many ways to discretize an ODE.<sup>2</sup> Here we discuss the **generalized bilinear transform** [ZCC07]. In this approach, we specify a step size  $\Delta$  and compute

$$\mathbf{z}_{k+1} = \underbrace{(\mathbf{I} - \alpha\Delta\mathbf{A})^{-1}(\mathbf{I} + (1 - \alpha)\Delta\mathbf{A})}_{\bar{\mathbf{A}}} \mathbf{z}_k + \underbrace{\Delta(\mathbf{I} - \alpha\Delta\mathbf{A})^{-1}\mathbf{B}}_{\bar{\mathbf{B}}} \mathbf{u}_k \quad (31.13)$$

If we set  $\alpha = 0$ , we recover **Euler's method**, which simplifies to

$$\mathbf{z}_{k+1} = \underbrace{(\mathbf{I} + \Delta\mathbf{A})}_{\bar{\mathbf{A}}} \mathbf{z}_k + \underbrace{\Delta\mathbf{B}}_{\bar{\mathbf{B}}} \mathbf{u}_k \quad (31.14)$$

If we set  $\alpha = 1$ , we recover the **backward Euler method**. If we set  $\alpha = \frac{1}{2}$  we get the **bilinear method**, which preserves the stability of the system [ZCC07]; we will use this in ???. Regardless of how we do the discretization, the resulting discrete time SSM becomes

$$\mathbf{z}_k = \bar{\mathbf{A}}\mathbf{z}_{k-1} + \bar{\mathbf{B}}\mathbf{u}_k \quad (31.15)$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{z}_k + \mathbf{D}\mathbf{u}_k \quad (31.16)$$

Now consider simulating a system where we periodically “tug” on the object, so the force increases and then decreases for a short period, as shown in the top row of Figure 31.3. We can discretize the dynamics and compute the corresponding state and observation at integer time points. The result is shown in the bottom row of Figure 31.3. We see that the object’s location changes smoothly, since it integrates the force over time.

### 31.2.2 Example: Noisy 2d tracking problem

We now consider a 2d example, such as a particle moving in  $\mathbb{R}^2$ . We follow the approach in Section 31.2.1 to convert the dynamics to discrete time. For notational simplicity, we revert to

<sup>2</sup> See discussion at <https://en.wikipedia.org/wiki/Discretization>.

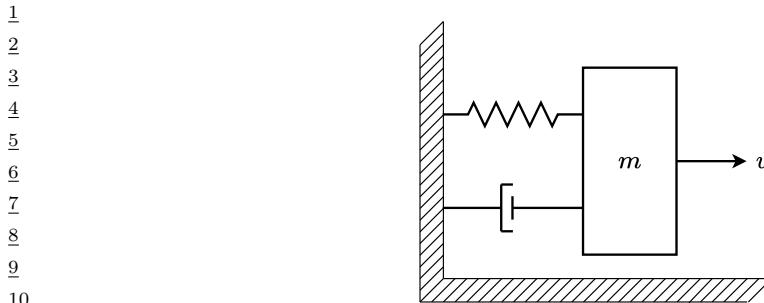


Figure 31.2: Illustration of the spring mass system.

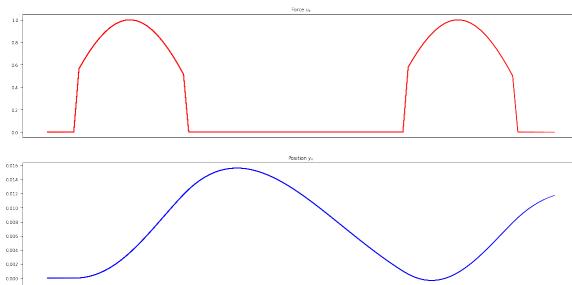


Figure 31.3: Signals generated by the spring mass system. Top row shows the input force. Bottom row shows the observed location of the end-effector. Adapted from a figure by Sasha Rush. Generated by [ssm\\_spring\\_demo.ipynb](#).

indexing (discrete) time with  $t$  rather than  $k$ , and we drop the overline symbol from the matrices. Furthermore, we ignore any input or control signals.

Let the state be the position and velocity of the object,  $\mathbf{z}_t = (u_t \quad \dot{u}_t \quad v_t \quad \dot{v}_t)$ . (We use  $u$  and  $v$  for the two coordinates, to avoid confusion with the state and observation variables.) If we use Euler discretization, the dynamics become

$$\underbrace{\begin{pmatrix} u_t \\ \dot{u}_t \\ v_t \\ \dot{v}_t \end{pmatrix}}_{\mathbf{z}_t} = \underbrace{\begin{pmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} u_{t-1} \\ \dot{u}_{t-1} \\ v_{t-1} \\ \dot{v}_{t-1} \end{pmatrix}}_{\mathbf{z}_{t-1}} + \boldsymbol{\epsilon}_t \quad (31.17)$$

where  $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$  is the **process noise**.

Let us assume that the process noise is a **white noise process** added to the velocity components of the state, but not to the location. (This is known as a **random accelerations model**.) One can

show [Sar13, p60] that the covariance matrix of the discrete time noise process is given by

$$\mathbf{Q} = \begin{pmatrix} \frac{q_1^c \Delta^3}{3} & 0 & \frac{q_1^c \Delta^2}{2} & 0 \\ 0 & \frac{q_2^c \Delta^3}{3} & 0 & \frac{q_2^c \Delta^2}{2} \\ \frac{q_1^c \Delta^2}{2} & 0 & q_1^c \Delta^2 & 0 \\ 0 & \frac{q_2^c \Delta^2}{2} & 0 & q_2^c \Delta^2 \end{pmatrix} \quad (31.18)$$

where  $q_i^c$  is the **spectral density** (continuous time variance) of the process noise for the  $i$ 'th component of the velocity. For simplicity, we often take  $q_1^c = q_2^c = q$  and  $\Delta = 1$ , in which case this simplifies to

$$\mathbf{Q} = \begin{pmatrix} q/3 & 0 & q/2 & 0 \\ 0 & q/3 & 0 & q/2 \\ q/2 & 0 & q & 0 \\ 0 & q/2 & 0 & q \end{pmatrix} \quad (31.19)$$

We sometimes further approximate this by  $\mathbf{Q} = q\mathbf{I}$ .

Now suppose that at each discrete time point we observe the location, corrupted by Gaussian noise. Thus the observation model becomes

$$\underbrace{\begin{pmatrix} y_{1,t} \\ y_{2,t} \end{pmatrix}}_{\mathbf{y}_t} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\mathbf{C}} \underbrace{\begin{pmatrix} u_t \\ \dot{u}_t \\ v_t \\ \dot{v}_t \end{pmatrix}}_{\mathbf{z}_t} + \boldsymbol{\eta}_t \quad (31.20)$$

where  $\boldsymbol{\eta}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$  is the **observation noise**. We see that the observation matrix  $\mathbf{C}$  simply “extracts” the relevant parts of the state vector.

Suppose we sample a trajectory and corresponding set of noisy observations from this model,  $(\mathbf{z}_{1:T}, \mathbf{y}_{1:T}) \sim p(\mathbf{z}, \mathbf{y} | \boldsymbol{\theta})$ . (We use diagonal observation noise,  $\mathbf{R} = \text{diag}(\sigma_1^2, \sigma_2^2)$ .) The results are shown in Figure 31.4(a). We can use the **Kalman filter** (Section 8.4.1) to compute  $p(\mathbf{z}_t | \mathbf{y}_{1:t}, \boldsymbol{\theta})$  for each  $t$ ,. (We initialize the filter with a vague prior, namely  $p(\mathbf{z}_0) = \mathcal{N}(\mathbf{z}_0 | \mathbf{0}, 10^5 \mathbf{I})$ .) The results are shown in Figure 31.4(b). We see that the posterior mean (red line) is close to the ground truth, but there is considerable uncertainty (shown by the confidence ellipses). To improve results, we can use the **Kalman smoother** (Section 8.4.4) to compute  $p(\mathbf{z}_t | \mathbf{y}_{1:T}, \boldsymbol{\theta})$ , where we condition on all the data, past and future. The results are shown in Figure 31.4(c). Now we see that the resulting estimate is smoother, and the uncertainty is reduced. (The uncertainty is larger at the edges because there is less information in the neighbors to condition on.)

The dynamics of the object in Figure 31.4 are rather simple, since we assumed a linear model with no external inputs, except for Gaussian process noise. However, suppose we change the (discrete time) linear transition matrix to the following:

$$\mathbf{A} = \begin{pmatrix} 0.1 & 1.1 & \Delta & 0 \\ -1 & 1 & 0 & \Delta \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{pmatrix} \quad (31.21)$$

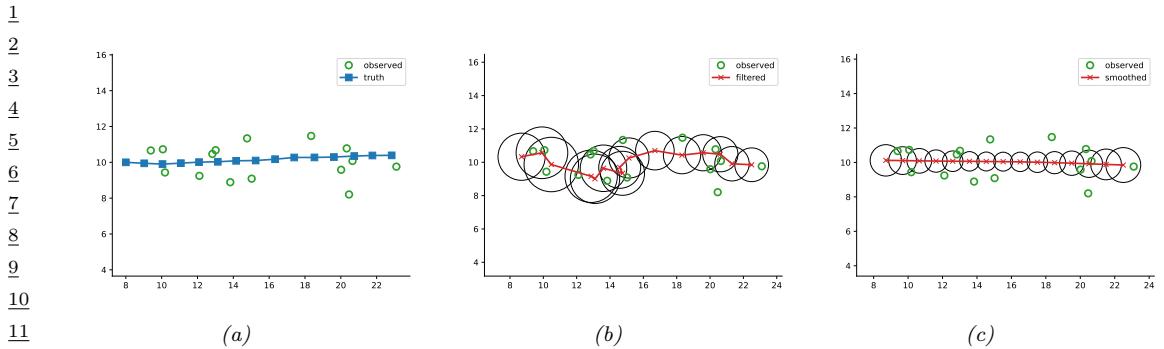


Figure 31.4: Illustration of Kalman filtering and smoothing for a linear dynamical system. (Repeated from Figure 8.5.) (a) Observations (green circles) are generated by an object moving to the right (true location denoted by blue squares). (b) Results of online Kalman filtering. Red cross is the posterior mean, circles are 95% confidence ellipses derived from the posterior covariance. (c) Same as (b), but using offline Kalman smoothing. The MSE in the trajectory for filtering is 3.13, and for smoothing is 1.71. Generated by kf\_tracking.py.

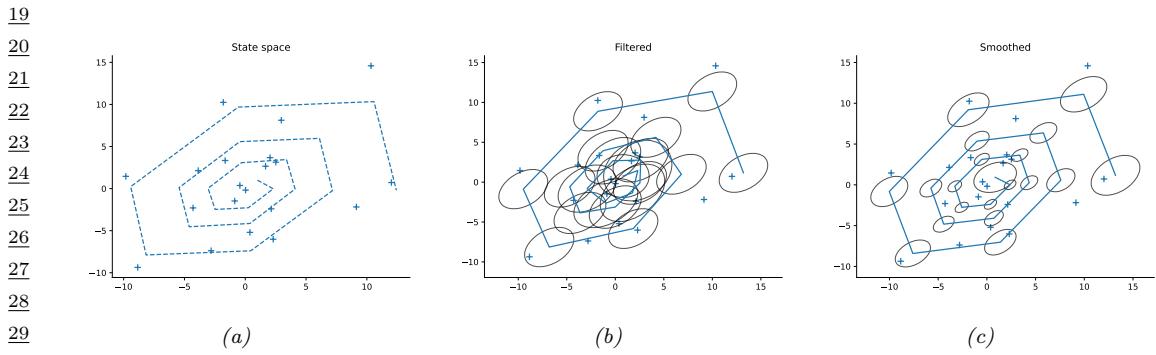


Figure 31.5: Illustration of Kalman filtering and smoothing for a linear dynamical system. (a) Observed data. (b) Filtering. (c) Smoothing. Generated by kf\_spiral.py.

The eigenvectors of the top left block of this transition matrix are complex, resulting in cyclical behavior, as explained in [Str15]. Furthermore, since the velocities are shrinking at each step by a factor of 0.1, the cycling behavior becomes a spiral inwards, as illustrated by the solid line in Figure 31.5(a). The crosses correspond to noisy measurements of the location, as before. In Figure 31.5(b-c), we show the results of Kalman filtering and smoothing.

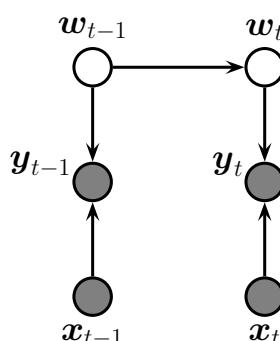
41

42

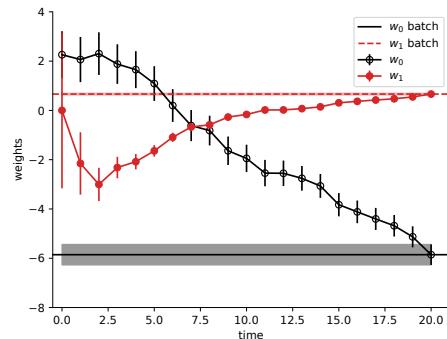
### 43 31.2.3 Example: Online linear regression

44

45 In Section 15.2.1, we discuss how to compute  $p(\mathbf{w}|\sigma^2, \mathcal{D})$  for a linear regression model in batch mode, 46 using a Gaussian prior of the form  $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ . In this section, we discuss how to compute 47



(a)



(b)

Figure 31.6: (a) A dynamic generalization of linear regression. (Repeated from Figure 8.9.) (b) Illustration of the recursive least squares algorithm applied to the model  $p(y|x, \mathbf{w}) = \mathcal{N}(y|w_0 + w_1 x, \sigma^2)$ . We plot the marginal posterior of  $w_0$  and  $w_1$  vs number of data points. (Error bars represent  $\mathbb{E}[w_j|y_{1:t}, \mathbf{x}_{1:t}] \pm \sqrt{\mathbb{V}[w_j|y_{1:t}, \mathbf{x}_{1:t}]}$ .) After seeing all the data, we converge to the offline (batch) Bayes solution, represented by the horizontal lines. (Shading represents the marginal posterior variance.) Generated by `linreg_kf.py`.

this posterior online, by repeatedly performing the following update:

$$p(\mathbf{w}|\mathcal{D}_{1:t}) \propto p(\mathcal{D}_t|\mathbf{w})p(\mathbf{w}|\mathcal{D}_{1:t-1}) \quad (31.22)$$

$$\propto p(\mathcal{D}_t|\mathbf{w})p(\mathcal{D}_{t-1}|\mathbf{w}) \dots p(\mathcal{D}_1|\mathbf{w})p(\mathbf{w}) \quad (31.23)$$

where  $\mathcal{D}_t = (\mathbf{x}_t, y_t)$  is the  $t$ 'th labeled example, and  $\mathcal{D}_{1:t-1}$  are the first  $t-1$  examples. (For brevity, we drop the conditioning on  $\sigma^2$ .) We see that the previous posterior,  $p(\mathbf{w}|\mathcal{D}_{1:t-1})$ , becomes the current prior, which gets updated by  $\mathcal{D}_t$  to become the new posterior,  $p(\mathbf{w}|\mathcal{D}_{1:t})$ . This is an example of sequential Bayesian updating or online Bayesian inference.

We can implement this method by using a linear Gaussian state space model. The basic idea is to let the hidden state represent the regression parameters, and to let the (time-varying) observation model represent the current data vector. If we assume the regression parameters do not change, the dynamics model becomes

$$p(\mathbf{w}_t|\mathbf{w}_{t-1}) = \mathcal{N}(\mathbf{w}_t|\mathbf{w}_{t-1}, 0) = \delta(\mathbf{w}_t - \mathbf{w}_{t-1}) \quad (31.24)$$

If we do let the parameters change over time, by adding non-zero **process noise**  $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ , we get a so-called **dynamic linear model** [Har90; WH97; PPC09].) The (non-stationary) observation model has the form

$$p(y_t|\mathbf{w}_t, \mathbf{x}_t) = \mathcal{N}(y_t|\mathbf{C}_t z_t, \mathbf{R}_t) = \mathcal{N}(y_t|\mathbf{x}_t^\top \mathbf{w}_t, \sigma^2) \quad (31.25)$$

See Figure 31.6a for the model. (We assume that  $\sigma^2$  is a known constant, so it is omitted from the diagram.)

If we apply the Kalman filter to this model, we recover an algorithm known as **recursive least squares** or **RLS**; see Section 8.4.2 for the details. In Figure 8.9b, we show that this converges to the optimal offline Bayes posterior after a single pass over the data.

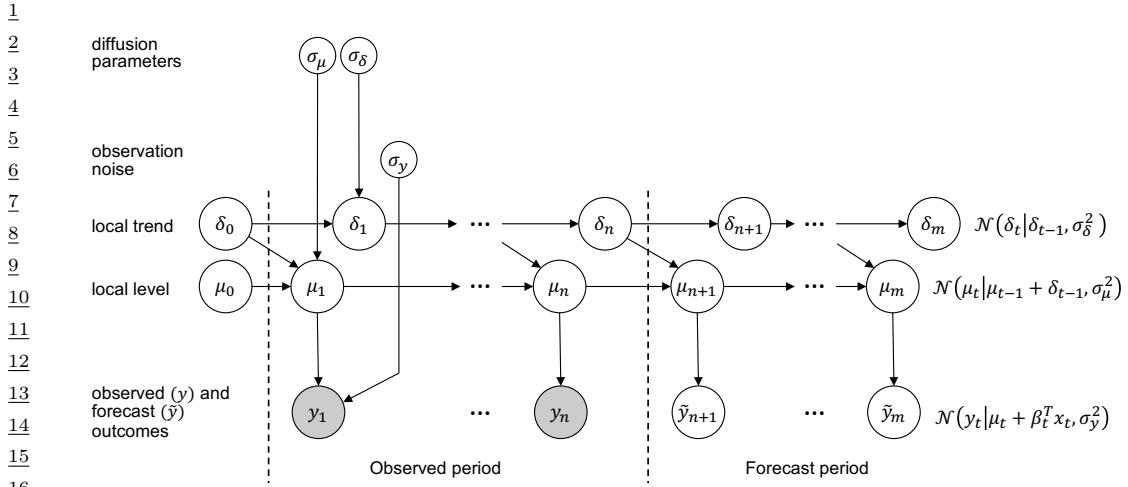


Figure 31.7: A graphical model representation of the local level STS model. Adapted from Figure 2 of [Bro+15].  
Used with kind permission of Kay Brodersen.

In Section 8.8.4, we extend this approach to perform online parameter estimation for logistic regression, and in Section 17.6.1, we extend this approach to perform nline parameter estimation for MLPs.

### 31.2.4 Example: structural time series forecasting

In Section 19.3.1, we discuss how to use LDS models for time series forecasting using a **structural time series** model. There are many kinds of STS model, but the simplest is the **local linear model**, to capture linear trends. (We do not add external covariates,  $\mathbf{x}_t$ , since for forecasting into the future, these will be unknown.) This model can be written as follows:

$$\underbrace{\begin{pmatrix} \mu_t \\ \delta_t \end{pmatrix}}_{\mathbf{z}_t} = \underbrace{\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} \mu_{t-1} \\ \delta_{t-1} \end{pmatrix}}_{\mathbf{z}_{t-1}} + \underbrace{\begin{pmatrix} \epsilon_{\mu,t} \\ \epsilon_{\delta,t} \end{pmatrix}}_{\boldsymbol{\epsilon}_t} \quad (31.26)$$

$$y_t = \underbrace{\begin{pmatrix} 1 & 0 \end{pmatrix}}_{\mathbf{C}} \underbrace{\begin{pmatrix} \mu_t \\ \delta_t \end{pmatrix}}_{\mathbf{z}_t} + \epsilon_{y,t} \quad (31.27)$$

See Figure 31.7 for the graphical model. We can use the Kalman filter to compute  $p(\mathbf{z}_t | \mathbf{y}_{1:t})$ , and then make predictions forwards in time. See Section 19.3.1 for details.

42

### 31.2.5 Parameter estimation

In the examples in Section 31.2.1, Section 31.2.2 and Section 31.2.3, the transition and observation matrices,  $\mathbf{A}$  and  $\mathbf{C}$ , were specified by hand, and the only unknown parameters were the noise terms

**Q** and **R**. This is commonly the case when the hidden state has some well-specified physical meaning. However, there are also applications where the hidden state is just a latent vector we introduce to compress the data, similar to PCA; in this case we also need to estimate **A** and **C**.

There are many approaches for estimating the parameters of state space models. (In the control theory community, this is known as **systems identification** [Lju87].) In the case of linear dynamical systems, many of the methods are similar to techniques used to fit HMMs, discussed in Section 30.4. For example, we can use EM, SGD, or spectral methods, as we discuss below. (We can also use Bayesian inference, see Section 31.4.2.)

### 31.2.5.1 Training with fully observed data

If we observe the hidden state sequences, we can fit the model by computing the MLEs for the parameters by solving a multivariate linear regression problem for  $\mathbf{z}_{t-1} \rightarrow \mathbf{z}_t$  and for  $\mathbf{z}_t \rightarrow \mathbf{y}_t$ . That is, we can estimate **A** by solving the least squares problem  $\mathcal{L}(\mathbf{A}) = \sum_{t=1}^T (\mathbf{z}_t - \mathbf{A}\mathbf{z}_{t-1})^2$ , Similarly, we can estimate **C** by minimizing  $\mathcal{L}(\mathbf{C}) = \sum_{t=1}^T (\mathbf{C}\mathbf{z}_t - \mathbf{y}_t)^2$ .

We can estimate the system noise covariance **Q** from the residuals in predicting  $\mathbf{z}_t$  from  $\mathbf{z}_{t-1}$ , and estimate the observation noise covariance **R** from the residuals in predicting  $\mathbf{y}_t$  from  $\mathbf{z}_t$ . However, we can set **Q** = **I** without loss of generality, since an arbitrary noise covariance can be modeled by appropriately modifying **A**. Also, by analogy with factor analysis, we can require **R** to be diagonal without loss of generality. Doing this reduces the number of free parameters and improves numerical stability.

### 31.2.5.2 EM for LG-SSM

If we only observe the output sequence, we can compute ML or MAP estimates of the parameters using EM. The method is conceptually quite similar to the Baum-Welch algorithm for HMMs (Section 30.4.1), except we use Kalman smoothing instead of forwards-backwards in the E step, and use different calculations in the M step. In particular, we need to compute the following expected sufficient statistics:

$$\sum_{t=1}^T \mathbb{E}[\mathbf{z}_t \mathbf{z}_t^\top], \sum_{t=1}^T \mathbb{E}[\mathbf{z}_t \mathbf{z}_{t+1}^\top], \sum_{t=1}^T \mathbb{E}[\mathbf{z}_t \mathbf{y}_t^\top] \quad (31.28)$$

where all expectations are wrt  $p(\mathbf{z}_t | \mathbf{y}_{1:T}, \boldsymbol{\theta})$ . We can maximize the expected complete data log likelihood by using a weighted least squares method. The details can be found in [GH96a].

Note that computing the expected sufficient statistics in Equation (31.28) in the inner loop of EM takes  $O(T)$  time, which can be expensive for long sequences. In [Mar10b], a faster method, known as **ASOS** (approximate second order statistics), is proposed. In this approach, various statistics are precomputed in a single pass over the sequence, and from then on, all iterations take constant time (independent of  $T$ ).

### 31.2.5.3 Subspace identification methods

EM does not always give satisfactory results, because it is sensitive to the initial parameter estimates. One way to avoid this is to use a different approach known as a **subspace identification (SSID)** [OM96; Kat05].

1 To understand this approach, let us initially assume there is no observation noise and no system  
2 noise. In this case, we have  $\mathbf{z}_t = \mathbf{A}\mathbf{z}_{t-1}$  and  $\mathbf{y}_t = \mathbf{C}\mathbf{z}_t$ , and hence  $\mathbf{y}_t = \mathbf{C}\mathbf{A}^{t-1}\mathbf{z}_1$ . Consequently all  
3 the observations must be generated from a  $\dim(\mathbf{z}_t)$ -dimensional linear manifold or subspace. We can  
4 identify this subspace using PCA. Once we have an estimate of the  $\mathbf{z}_t$ 's, we can fit the model as if it  
5 were fully observed. We can either use these estimates in their own right, or use them to initialize  
6 EM. Several papers (e.g., [Smi+00; BK15]) have shown that initializing EM this way gives much  
7 better results than initializing EM at random, or just using SSID without EM.

8  
9 Although the theory only works for noise-free data, we can try to estimate the system noise  
10 covariance  $\mathbf{Q}$  from the residuals in predicting  $\mathbf{z}_t$  from  $\mathbf{z}_{t-1}$ , and to estimate the observation noise  
11 covariance  $\mathbf{R}$  from the residuals in predicting  $\mathbf{y}_t$  from  $\mathbf{z}_t$ . We can either use these estimates in their  
12 own right, or use them to initialize EM. Because this method relies on taking an SVD, it is called a  
13 **spectral estimation method**. Similar methods can also be used for HMMs (see Section 30.4.3).

1415

#### 16 31.2.5.4 Numerical stability

17  
18 When estimating the dynamics matrix  $\mathbf{A}$ , it is very useful to impose a constraint on its eigenvalues.  
19 To see why this is important, consider the case of no system noise. In this case, the hidden state at  
20 time  $t$  is given by

21

$$\underline{22} \quad \mathbf{z}_t = \mathbf{A}^t \mathbf{z}_1 = \mathbf{U} \Lambda^t \mathbf{U}^{-1} \mathbf{z}_1 \quad (31.29)$$

23

24 where  $\mathbf{U}$  is the matrix of eigenvectors for  $\mathbf{A}$ , and  $\Lambda = \text{diag}(\lambda_i)$  contains the eigenvalues. If any  
25  $\lambda_i > 1$ , then for large  $t$ ,  $\mathbf{z}_t$  will blow up in magnitude. Consequently, to ensure stability, it is useful  
26 to require that all the eigenvalues are less than 1 [SBG07]. Of course, if all the eigenvalues are less  
27 than 1, then  $\mathbb{E}[\mathbf{z}_t] = \mathbf{0}$  for large  $t$ , so the state will return to the origin. Fortunately, when we add  
28 noise, the state becomes non-zero, so the model does not degenerate.

2930

### 31 31.3 Non-linear dynamical systems

32  
33 In this section, we consider a **nonlinear dynamical system (NLDS)** with additive Gaussian noise.  
34 The corresponding generative model is as follows:

35

$$\underline{37} \quad \mathbf{z}_t = f(\mathbf{z}_{t-1}, \mathbf{u}_t) + \boldsymbol{\epsilon}_t \quad (31.30)$$

$$\underline{38} \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t) \quad (31.31)$$

$$\underline{39} \quad \mathbf{y}_t = h(\mathbf{z}_t, \mathbf{u}_t) + \boldsymbol{\eta}_t \quad (31.32)$$

$$\underline{40} \quad \boldsymbol{\eta}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t) \quad (31.33)$$

41

42 Henceforth we will ignore the inputs  $\mathbf{u}_t$  for brevity.

43 Inferring the states of an NLDS model is in general computationally difficult. Fortunately, there  
44 are a variety of approximate inference schemes that can be used, such as the extended Kalman filter  
45 (Section 8.5.2), the unscented Kalman filter (Section 8.6.2), the particle filtering (Section 13.2), etc.  
46

47

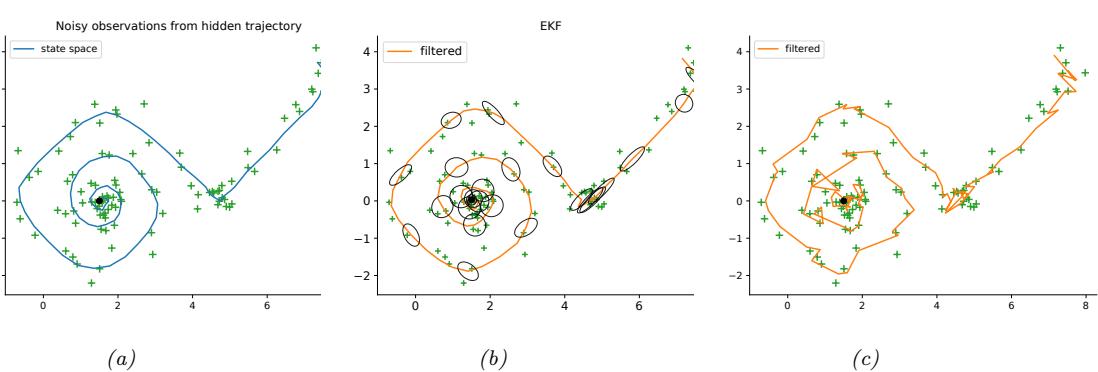


Figure 31.8: Illustration of filtering applied to a 2d nonlinear dynamical system. (a) True underlying state and observed data. (b) Extended Kalman filter estimate Generated by [ekf\\_vs\\_ukf.py](#). (c) Particle filter estimate using 2000 samples. Generated by [bootstrap\\_filter.py](#).

### 31.3.1 Example: nonlinear 2d tracking problem

Consider the following nonlinear dynamical system in 2d:

$$\mathbf{f}(\mathbf{z}) = (z_1 + \Delta \sin(z_2), z_2 + \Delta \cos(z_1)) \quad (31.34)$$

$$\mathbf{y} = \mathbf{h}(\mathbf{z}) = (z_1, z_2) \quad (31.35)$$

where  $\Delta$  is the step size. We assume Gaussian noise is added to the state transitions and observations. In Figure 31.8b, we show the results of EKF, and in Figure 31.8c, we show the results of PF (using the dynamics model as a proposal); we see that both methods work quite well in this simple problem.

### 31.3.2 Example: Simultaneous localization and mapping (SLAM)

Consider a robot moving around a 2d surface. It needs to learn a map of the environment, and keep track of its location (pose) within that map. This problem is known as **simultaneous localization and mapping**, or **SLAM** for short. SLAM is widely used in mobile robotics (see e.g., [SC86; CN01; TBF06] for details). It is also useful in augmented reality, where the task is to recursively estimate the 3d pose of a handheld camera with respect to a set of 2d visual landmarks (this is known as **visual SLAM**). See e.g., [TUI17; SMT18; Cza+20; DH22] for details.

Let us assume we can represent the map as the 2d locations of a set of  $K$  landmarks, denote them by  $\mathbf{l}^1, \dots, \mathbf{l}^K$  (each is a vector in  $\mathbb{R}^2$ ). (We can use data association to figure out which landmark generated each observation, as discussed in Section 31.3.4.) Let  $\mathbf{r}_t$  represent the unknown location of the robot at time  $t$ . Let  $\mathbf{z}_t = (\mathbf{r}_t, \mathbf{l}_t^{1:K})$  be the combined state space.

The motion model is defined as

$$p(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{u}_t) = p(\mathbf{r}_t | \mathbf{r}_{t-1}, \mathbf{l}_{t-1}^{1:K}, \mathbf{u}_t) \prod_{k=1}^K p(\mathbf{l}_t^k | \mathbf{l}_{t-1}^k) \quad (31.36)$$

where  $p(\mathbf{r}_t | \mathbf{r}_{t-1}, \mathbf{l}_{t-1}^{1:K}, \mathbf{u}_t)$  specifies how the robot moves given the control signal  $\mathbf{u}_t$  and the location of the obstacles  $\mathbf{l}_{t-1}^{1:K}$ . (Note that in this section, we assume that a human is joysticking the robot

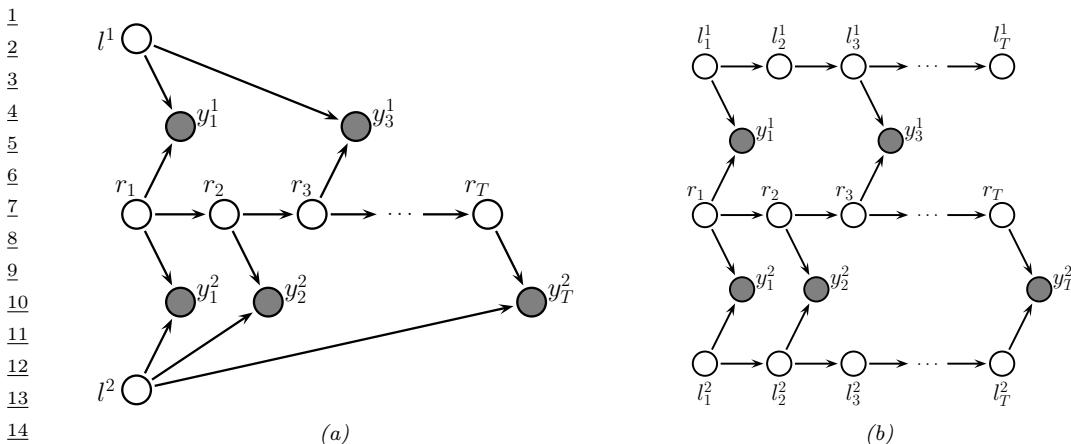


Figure 31.9: Illustration of graphical model underlying SLAM.  $\mathbf{l}_t^k$  is the location of landmark  $k$  at time  $t$ ,  $\mathbf{r}_t$  is the location of the robot at time  $t$ , and  $\mathbf{y}_t$  is the observation vector. In the model on the left, the landmarks are static, on the right, their location can change over time. The robot's observations are based on the distance to the nearest landmarks from the current state, denoted  $f(\mathbf{r}_t, \mathbf{l}_t^k)$ . In this example, the robot gets the following observation trace:  $\mathbf{y}_1 = [f(\mathbf{r}_1, \mathbf{l}_1^1), f(\mathbf{r}_1, \mathbf{l}_1^2)]$ ,  $\mathbf{y}_2 = f(\mathbf{r}_2, \mathbf{l}_2^2)$  up to  $\mathbf{y}_T = f(\mathbf{r}_T, \mathbf{l}_T^2)$ . Thus we see that the number of observations per time step is variable. Adapted from Figure 15.A.3 of [KF09a].

<sup>24</sup>through the environment, so  $u_{1:t}$  is given as input, i.e., we do not address the decision-theoretic issue of choosing where to move.)

If the obstacles (landmarks) are static, we can define  $p(\mathbf{l}_t^k | \mathbf{l}_{t-1}^k) = \delta(\mathbf{l}_t^k - \mathbf{l}_{t-1}^k)$ , which is equivalent to treating the map as unknown parameter that is shared globally across all time steps. More generally, we can let the landmark locations evolve over time [Mur00].

29 The observations  $y_t$  measure the distance from  $r_t$  to the set of closest landmarks. Figure 31.9  
 30 shows the corresponding graphical model for the case where  $K = 2$ , and where on the first step it  
 31 sees landmarks 1 and 2, then just landmark 2, then just landmark 1, etc. We can then perform online  
 32 inference so that the robot can update its estimate of its own location, and the landmark locations.  
 33 If all the CPDs are linear-Gaussian, then we can use a Kalman filter to maintain our belief state

34 about the location of the robot and the location of the landmarks,  $p(\mathbf{z}_t|\mathbf{y}_{1:t}, \mathbf{u}_{1:t})$ . In the more  
35 general case of a nonlinear model, we can use the EKF (Section 8.5.2) or UKF (Section 8.6.2).

Over time, the uncertainty in the robot's location will increase, due to wheel slippage etc., but when the robot returns to a familiar location, its uncertainty will decrease again. This is called **closing the loop**, and is illustrated in Figure 31.10(a), where we see the uncertainty ellipses, representing  $\text{Cov}[\mathbf{z}_t | \mathbf{u}_{1:t}, \mathbf{u}_{1:t}]$ , grow and then shrink.

40 In addition to visualizing the uncertainty of the robot’s location, we can visualize the uncertainty  
41 about the map. To do this, consider the posterior precision matrix,  $\Lambda_t = \Sigma_t^{-1}$ . Zeros in the precision  
42 matrix correspond to absent edges in the corresponding undirected Gaussian graphical model (see  
43 Section 4.3.2.8). Initially all the landmarks are uncorrelated (by assumption), so the GGM is a  
44 disconnected graph, and  $\Lambda_t$  is diagonal. However, as the robot moves about, it will induce correlation  
45 between nearby landmarks. Intuitively this is because the robot is estimating its position based on  
46 distance to the landmarks, but the landmarks’ locations are being estimated based on the robot’s  
47

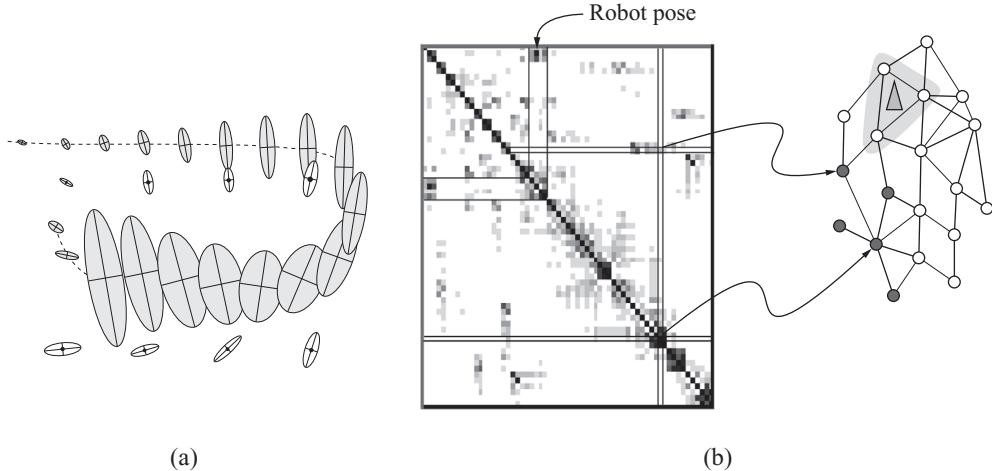


Figure 31.10: Illustration of the SLAM problem. (a) A robot starts at the top left and moves clockwise in a circle back to where it started. We see how the posterior uncertainty about the robot’s location increases and then decreases as it returns to a familiar location, closing the loop. If we performed smoothing, this new information would propagate backwards in time to disambiguate the entire trajectory. (b) We show the precision matrix, representing sparse correlations between the landmarks, and between the landmarks and the robot’s position (pose). This sparse precision matrix can be visualized as a Gaussian graphical model, as shown on the right. From Figure 15.A.3 of [KF09a]. Used with kind permission of Daphne Koller.

position, so they all become inter-dependent. This can be seen more clearly from the graphical model in Figure 31.9: it is clear that  $l^1$  and  $l^2$  are not d-separated by  $\mathbf{y}_{1:t}$ , because there is a path between them via the unknown sequence of  $\mathbf{r}_{1:t}$  nodes. Consequently, the precision matrix becomes denser over time. As a consequence of the precision matrix becoming denser, and each inference step takes  $O(K^3)$  time. This prevents the method from being applied to large maps.

One approach to the  $O(K^3)$  complexity problem is based on the observation that the correlation pattern moves along with the location of the robot (see Figure 31.10(b)). The remaining correlations become weaker over time. Consequently we can dynamically “prune out” weak edges from the GGM using a technique called the thin junction tree filter [Pas03] (junction trees are explained in Section 9.5).

A second approach is to notice that, conditional on knowing the robot’s path,  $\mathbf{r}_{1:t}$ , the landmark locations are independent, i.e.,  $p(\mathbf{l}_t | \mathbf{r}_{1:t}, \mathbf{y}_{1:t}) = \prod_{k=1}^K p(l_t^k | \mathbf{r}_{1:t}, \mathbf{y}_{1:t})$ . This can be seen by looking at the DGM in Figure 31.9. We can therefore sample the trajectory using particle filtering, and apply Kalman filtering to each landmark independently. See Section 13.5.2 for details.

### 31.3.3 Example: stochastic volatility models

In finance, it is common to model the the **log-returns**,  $y_t = \log(p_t/p_{t-1})$ , where  $p_t$  is the price of some asset at time  $t$ . A common model for this problem, known as a **stochastic volatility model**,

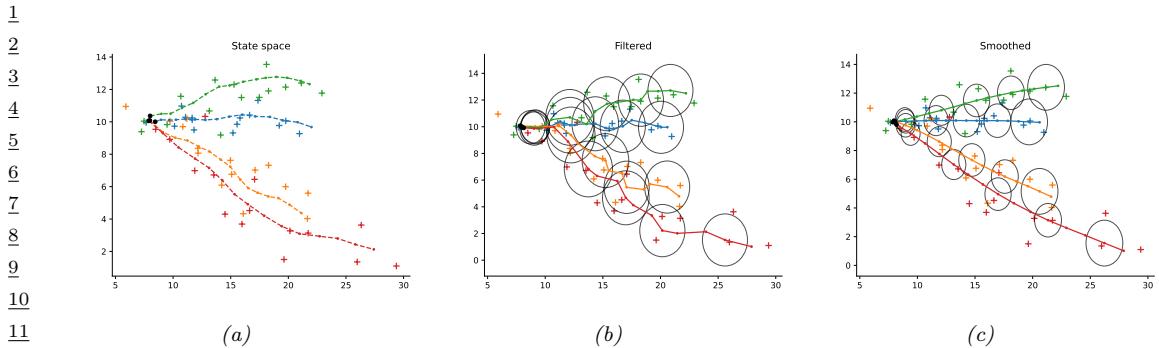


Figure 31.11: Illustration of Kalman filtering and smoothing for tracking multiple moving objects. Generated by [kf\\_parallel.py](#).

is the following:

$$p(y_t|z_t) = \sigma_y^2 \exp(z_t) \mathcal{N}(0, 1) \quad (31.37)$$

$$p(z_t|z_{t-1}) = \mathcal{N}(\mu + \rho(z_{t-1} - \mu), \sigma_z^2) \quad (31.38)$$

(For identifiability, we must either pick  $\sigma_y = 1$  or  $\mu = 0$ , with the latter being preferred for computational reasons [KSC98].) We see that the dynamical model is a first-order autoregressive process. We typically require that  $|\rho| < 1$ , to ensure the system is stationary. The observation model is Gaussian, but can be replaced by a heavy-tailed distribution such as a Student.

We can capture longer range temporal correlation by using a higher order auto-regressive process. To do this, we just expand the state space to contain the past  $K$  values. For example, if  $K = 2$  we have

$$\begin{pmatrix} z_t - \mu \\ z_{t-1} - \mu \end{pmatrix} = \begin{pmatrix} \rho_1 & \rho_2 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} z_{t-1} - \mu \\ z_{t-2} - \mu \end{pmatrix} + \begin{pmatrix} q_t \\ 0 \end{pmatrix} \quad (31.39)$$

where  $q_t \sim \mathcal{N}(0, \sigma_z^2)$ . Thus we have

$$z_t = \mu + \rho_1(z_{t-1} - \mu) + \rho_2(z_{t-2} - \mu) + q_t \quad (31.40)$$

For more details, see [KSC98].

### 31.3.4 Example: Multi-target tracking

The problem of **multi-target tracking** frequently arises in engineering applications (especially in aerospace and defence), and can be tackled by inference in various kinds of SSMs. This is a very large topic (see e.g. [Vo+15] for details), but we summarize the basic ideas below.

In the simplest setting, we know there are  $N$  objects we want to track, and each one generates its own uniquely identified observation. If we assume the objects are independent, we can apply Kalman filtering and smoothing in parallel, as shown in Figure 31.11. (In this example, each object follows a linear dynamical model with different initial random velocities, as in Section 31.2.2.)

47

More generally, at each step we may observe  $M$  measurements e.g., ‘‘blips’’ on a radar screen. We can have  $M < N$  due to occlusion or missed detections. We can have  $M > N$  due to clutter or false alarms. Or we can have  $M = N$ . In any case, we need to figure out the **correspondence** between the  $M$  detections  $\mathbf{x}_t^m$  and the  $N$  objects  $\mathbf{z}_t^i$ . This is called the problem of **data association**, and it arises in many application domains. See Figure 4.40(d) for an illustration, where we have  $M = 2$  observations per time step, but where there is uncertainty about how many objects are actually present.

We can model this problem by augmenting the state space with discrete variables  $s_t$  that represent the association matrix between the observations,  $\mathbf{y}_{t,1:M}$ , and the sources,  $\mathbf{z}_{t,1:N}$ . As we mentioned in Section 8.8.3.2, inference in such hybrid (discrete-continuous) models is intractable, due to the exponential number of posterior modes. In the sections below, we briefly mention a few approximate inference methods.

#### 31.3.4.1 Nearest neighbor approximation using Hungarian algorithm

A common way to perform approximate inference in this model is to compute an  $N \times M$  weight matrix, where  $W_{im}$  measures the ‘‘compatibility’’ between object  $i$  and measurement  $m$ , typically based on how close  $m$  is to where the model thinks  $i$  is (the so-called **nearest neighbor data association** heuristic).

We can make this into a square matrix by adding dummy background objects, which can explain all the false alarms, and adding dummy observations, which can explain all the missed detections. We can then compute the maximal weight bipartite matching using the **Hungarian algorithm**, which takes  $O(\max(N, M)^3)$  time (see e.g., [BDM09]).

Conditional on knowing the assignments of measurements to tracks, we can perform the usual Bayesian state update procedure (e.g., based on Kalman filtering). Note that objects that are assigned to dummy observations do not perform a measurement update, so their state estimate is just based on forwards prediction from the dynamics model.

#### 31.3.4.2 Other approximate inference techniques

The Hungarian algorithm can be slow (since it is cubic in the number of measurements), and can give poor results since it relies on hard assignment. Better performance can be obtained by using loopy belief propagation (Section 9.3). The basic idea is to approximately marginalize out the unknown assignment variables, rather than perform a MAP estimate. This is known as the **SPADA** method (sum-product algorithm for data association) [WL14; Mey+18].

The cost of each iteration of the iterative procedure is  $O(NM)$ . Furthermore, [WL14] proved this will always converge in a finite number of steps, and [Von13] showed that the corresponding solution will in fact be the global optimum. The SPADA method is more efficient, and more accurate, than earlier heuristic methods, such as **JPDA** (joint probabilistic data association) [BSWT11; Vo+15].

It is also possible to use sequential Monte Carlo methods to solve data association and tracking. See Section 13.2 for a general discussion of SMC, and [RAG04; Wan+17b] for a review of specific techniques for this model family.

1	Name	Distribution	$\eta_t$	$\phi$	$\gamma_t$
2	Gaussian	$\mathcal{N}(y \eta_t, \phi)$	Mean	Variance	-
3	Poisson	$\text{Poi}(y \gamma_t \exp(\eta_t))$	Log Rate	-	Exposure time
4	Binomial	$\text{Bin}(y \gamma_t, \sigma(\eta_t))$	Logit	-	Num. trials
5	Gamma	$\text{Ga}(y \exp(\eta_t), \phi)$	Log shape	Rate	-
6	Stochastic volatility	$y_t = \exp(\eta_t/2)\mathcal{N}(0, 1)$	Log volatilty	-	-

8 *Table 31.1: Some exponential family likelihoods. List is from [HV21]. In the stochastic volatility model*  
 9 *(Section 31.3.3), the dynamics is also constrained to have the form in Equation (31.38).*

10

11

12

### 13 31.3.4.3 Handling an unknown number of targets

14

15 In general, we do not know the true number of targets  $N$ , so we have to deal with variable-sized  
 16 state space. This is an example of an **open world** model [Rus15; LB19], which differs from the  
 17 standard **closed world assumption** where we know how many objects of interest there are. (See  
 18 Section 4.5.3.)

19 A common approximate solution to this is to create new objects whenever an observation cannot be  
 20 “explained” (i.e., generated with high likelihood) by any existing objects, and to prune out old objects  
 21 that have not been detected in a while (in order to keep the computational cost bounded). Sets  
 22 whose size and content are both random are called **random finite sets**. An elegant mathematical  
 23 framework for dealing with such objects is described in [Mah07; Mah13; Vo+15].

24

## 25 31.4 Other kinds of SSM

26

### 27 31.4.1 Exponential family SSM

28

29 It is natural to generalize Gaussian SSMs to the setting where the likelihood function is from the  
 30 exponential family. This is called an **exponential family state space model** (see e.g., [Vid99;  
 31 Hel17]). We will assume the observation model is as follows:

32

$$33 \quad p(y_t|\mathbf{z}_t, \mathbf{u}_t) = p_t(y_t|\mathbf{C}_t \mathbf{z}_t, \phi, \gamma_t) \quad (31.41)$$

34

35 where  $p_t(y_t|\eta_t, \phi, \gamma_t)$  is an exponential family with natural parameters  $\eta_t$ , and where  $\phi$  and  $\gamma_t$  are  
 36 optional extra parameters that depend on the model family. See Table 31.1 for some examples. (For  
 37 multivariate outputs, we assume the likelihood factorizes, so  $p(\mathbf{y}_t|\mathbf{z}_t, \mathbf{u}_t) = \prod_{d=1}^D p(y_{td}|\mathbf{z}_t, \mathbf{u}_t)$ .)

38

#### 39 31.4.1.1 Laplace EM method

40

41 In this section we discuss how to fit an SSM with linear-Gaussian latent dynamics and a GLM  
 42 likelihood using the **Laplace-EM** algorithm. As we discussed in Section 31.2.5.2, we need to compute  
 43 the expected sufficient statistics in the E step, and then we can easily maximize the expected complete  
 44 data log likelihood in the M step. We focus here on the E (inference) step.

45 First we compute the maximum  $\mathbf{z}_{1:T}^* = \arg\max_{\mathbf{z}_{1:T}} \log p(\mathbf{z}_{1:T}, \mathbf{y}_{1:T})$  using some gradient-based  
 46 optimizer. We then we compute the Hessian at this point, leveraging the fact that this is block  
 47

1 tridiagonal. In particular, let  
2

$$\underline{3} \quad \mathbf{H}_{t,y} = -\nabla_{\mathbf{z}_t}^2 \log p(\mathbf{y}_t | \mathbf{z}_t), \mathbf{H}_0 = \nabla \log p(\mathbf{z}_0) \quad (31.42)$$

$$\underline{5} \quad \mathbf{H}_{t,11} = -\nabla_{\mathbf{z}_t}^2 \log p(\mathbf{z}_{t+1} | \mathbf{z}_t), \mathbf{H}_{t,22} = -\nabla_{\mathbf{z}_t}^2 \log p(\mathbf{z}_t | \mathbf{z}_{t-1}), \mathbf{H}_{t,12} = -\nabla_{\mathbf{z}_t} \nabla_{\mathbf{z}_{t+1}} \log p(\mathbf{z}_{t+1} | \mathbf{z}_t) \quad (31.43)$$

7  
8 The Gaussian approximation to the posterior has the form  $p(\mathbf{z} | \mathbf{y}) \propto \mathcal{N}(\mathbf{z} | \boldsymbol{\mu}, \boldsymbol{\Sigma})$ , where  $\boldsymbol{\mu} = \mathbf{z}_{1:T}^*$ ,  
9 and the precision matrix is given by

$$\underline{10} \quad \boldsymbol{\Sigma}^{-1} = \begin{pmatrix} \mathbf{J}_{0,0} & \mathbf{J}_{0,1} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{J}_{1,0} & \mathbf{J}_{1,1} & \mathbf{J}_{1,2} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{J}_{2,1} & \mathbf{J}_{2,2} & \mathbf{J}_{1,2} & \cdots \end{pmatrix} \quad (31.44)$$

14 where  $\mathbf{J}_{t,t} = \mathbf{H}_{t,y} + \mathbf{H}_t + \mathbf{H}_{t,11}$  for  $t = 0$ ,  $\mathbf{J}_{t,t} = \mathbf{H}_{t,y} + \mathbf{H}_{t,11} + \mathbf{H}_{t,22}$  for  $t = 1 : T-1$ , and  
15  $\mathbf{J}_{t,t} = \mathbf{H}_{t,y} + \mathbf{H}_{t,22}$  for  $t = T$ .

16 Using this, the log joint has the following form, where  $\mathbf{J} = \boldsymbol{\Sigma}^{-1}$  and  $\mathbf{h} = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}$ :

$$\underline{18} \quad \log p(\mathbf{z}, \mathbf{y}) = \exp\left[-\frac{1}{2}\mathbf{z}^\top \mathbf{J} \mathbf{z} + \mathbf{z}^\top \mathbf{h} - \log Z(\mathbf{J}, \mathbf{h})\right] \quad (31.45)$$

$$\underline{20} \quad = \exp\left[-\frac{1}{2} \sum_{t=1}^T \mathbf{z}_t^\top \mathbf{J}_{t,t} \mathbf{z}_t - \sum_{t=1}^{T-1} \mathbf{z}_{t+1}^\top \mathbf{J}_{t+1,t} \mathbf{z}_t + \sum_{t=1}^T \mathbf{z}_t^\top \mathbf{h}_t - \log Z(\mathbf{J}, \mathbf{h})\right] \quad (31.46)$$

23 From this, we can compute the expected sufficient statistics needed for the E step using  $\mathbb{E}[\mathbf{z}_t | \mathbf{y}] = \nabla_{\mathbf{h}_t} \log Z(\mathbf{J}, \mathbf{h})$ ,  $\mathbb{E}[\mathbf{z}_t \mathbf{z}_t^\top | \mathbf{y}] = -2\nabla_{\mathbf{J}_{t,t}} \log Z(\mathbf{J}, \mathbf{h})$ , and  $\mathbb{E}[\mathbf{z}_{t+1} \mathbf{z}_t^\top | \mathbf{y}] = -\nabla_{\mathbf{J}_{t+1,t}} \log Z(\mathbf{J}, \mathbf{h})$ ,

26 To derive the MLE for the dynamics model from this, we can use a weighted least squares method  
27 [GH96a]. For the observation model, we can maximize the expected complete data log likelihood by  
28 sampling from the posterior.

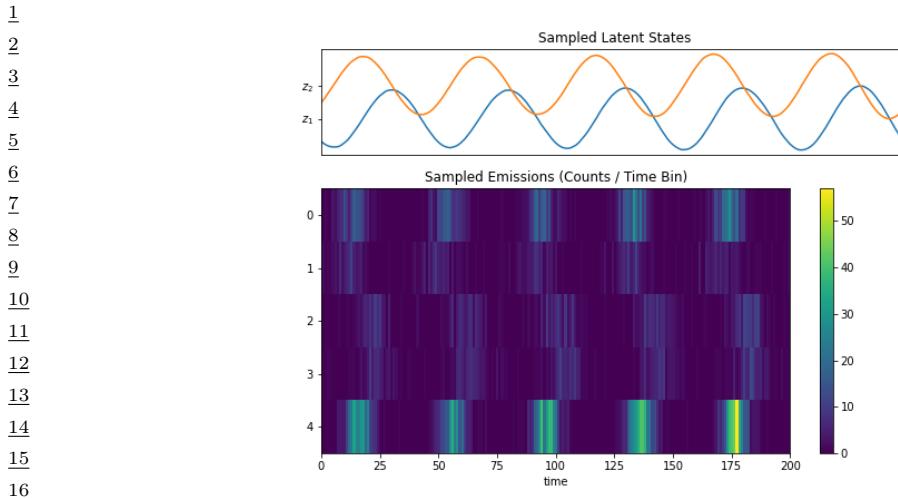
### 30 31.4.1.2 Example: Poisson likelihood

31 In this section we discuss consider an SSM with linear-Gaussian latent dynamics and a Poisson  
32 likelihood. Such models are widely used in neuroscience (see e.g., [Pan+10; Mac+11]). We create a  
33 model with 2 continuous latent variables, and we set the dynamics matrix  $\mathbf{A}$  to a random rotation  
34 matrix. The observation model has the form  $p(\mathbf{y}_t | \mathbf{z}_t) = \prod_{d=1}^D \text{Poi}(y_{td} | \exp(\mathbf{w}_d^\top \mathbf{z}_t))$ , where  $\mathbf{w}_d$  is a  
35 random vector, and we use  $D = 5$  observations per time step. Some samples from this model are  
36 shown in Figure 31.12.

37 We fit this model using the Laplace-EM algorithm from Section 31.4.1.1. We then perform posterior  
38 inference. We show the result of these two steps in Figure 31.13, where we compare the parameters  
39  $\mathbf{A}$  and the posterior trajectory  $\mathbb{E}[\mathbf{z}_t | \mathbf{y}_{1:T}]$  using the true model and the estimated model. We see  
40 good agreement.

### 42 31.4.1.3 Example: demand forecasting

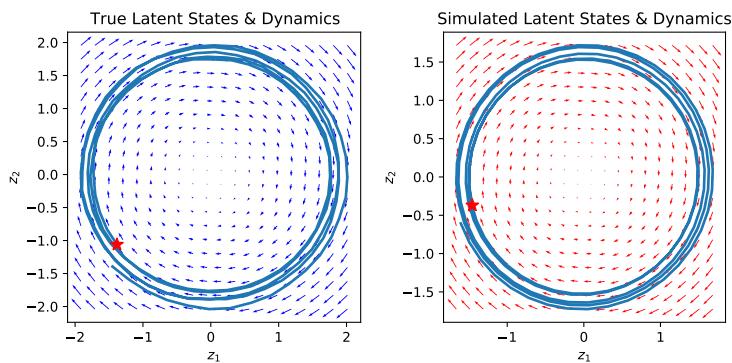
44 In this section, we consider the problem of **demand forecasting**, in which we want to predict how  
45 many of an item will be sold each day. Let the latent demand (for some item) be  $z_t$ , and the observed  
46 sales volume be  $y_t$ . Sometimes we get  $y_t = 0$  due to out-of-stock situations; if we do not model this



17 *Figure 31.12: Samples from a 2d LDS with 5 Poisson likelihood terms. Generated by [poisson\\_lds\\_example.ipynb](#).*

18

19



33 *Figure 31.13: Latent state trajectory (blue lines) and dynamics matrix  $\mathbf{A}$  (arrows) for (left) true model and (right) estimated model. The star marks the start of the trajectory. Generated by [poisson\\_lds\\_example.ipynb](#).*

34

35

38 properly, we may incorrectly infer that  $z_t = 0$ , thus underestimating demand. This may result in not  
39 ordering enough stock for the future, further compounding the error.

40 To avoid this, [SSF16] propose a model which combines SSMs with GLMs, as we discussed in  
41 Section 31.4.1. In particular, they consider a likelihood of the form  $y_t \sim \text{Poi}(y_t | g(d_t^y))$ , where  
42  $d_t = z_t + \mathbf{u}_t^\top \mathbf{w}$  is the instantaneous latent demand,  $g(d) = e^d$  or  $\log(1 + e^d)$  is the transfer function,  
43 and  $z_t = z_{t-1} + \alpha \mathcal{N}(0, 1)$  is a local random walk term for this latent demand (to capture serial  
44 correlation in the data). The covariates  $\mathbf{u}_t$  can encode seasonal indicators, such as distance from  
45 holidays, but can also encode out-of-stock signals. If the model knows that the item is unavailable,  
46 then  $y_t = 0$  is “explained away” (Section 4.2.3.2), and so we don’t need to update the latent state  $z_t$ .  
47

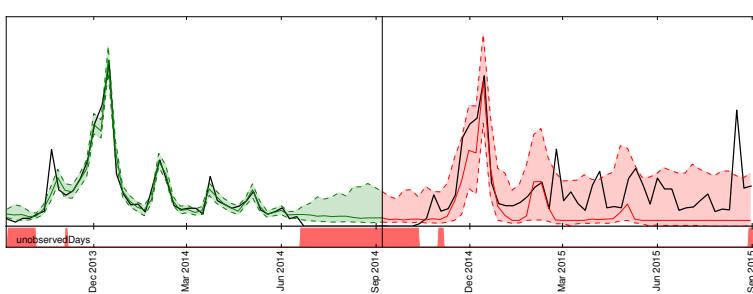


Figure 31.14: Visualization of a probabilistic demand forecast. The black line denotes the actual demand, while the green and red lines denote quantiles of the forecasted demand distribution (10th percentile, median and 90th percentile). Green lines denote the model samples in the training range, while the red lines show the actual probabilistic forecast on data unseen by the model. Note that the demand can be partially unobserved (e.g., due to out-of-stock situations), as indicated by the red bars at the bottom. From Figure 1 of [Bös+17]. Used with kind permission of Tim Januschowski.

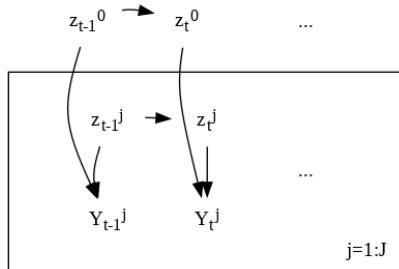


Figure 31.15: Illustration of hierarchical state-space model.

This avoids underestimating the demand, and thus ensures the supply chain is adequately stocked. See Figure 31.14 for an illustration.

The Poisson likelihood is not a good fit for datasets which have many zeros, such as inventory data, due to the out-of-stock problem. One solution is to use a **zero-inflated Poisson (ZIP)** model [Lam92] for the likelihood. This is a mixture model of the form  $p(y_t|d_t) = p_0\mathbb{I}(y_t = 0) + (1 - p_0)\text{Poi}(y_t|e^{d_t})$ , where  $p_0$  is the probability of the first mixture component. It is also common to use a (possibly zero-inflated) negative binomial model (Section 2.2.1.4) as the likelihood. This is used in [Cha14; Sal+19b] for the demand forecasting problem.

The disadvantage of these likelihoods is that they are not log-concave for  $d_t = 0$ , which complicates posterior inference. In particular, the Laplace approximation is a poor choice, since it may find a saddle point. In [SSF16], they tackle this using a log-concave **multi-stage likelihood**.

#### 31.4.1.4 Example: modeling electoral panel data

Suppose we perform a survey for the US presidential elections. Let  $N_t^j$  be the number of people who vote at time  $t$  in state  $j$ , and let  $Y_t^j$  be the number of those people who vote Democrat. (We assume

1  $N_t^j - Y_t^j$  vote Republican.) It is natural to want to model the dependencies in this data both across  
2 time (longitudinally) and across space (this is an example of **panel data**).

4 We can do this using a hierarchical SSM, as illustrated in Figure 31.15. The top level Markov  
5 chain,  $z_t^0$ , models national-level trends, and the state-specific chains,  $z_t^j$ , model local “random effects”.  
6 In practice we would usually also include covariates at the national level,  $\mathbf{u}_t^0$  and state level,  $\mathbf{u}_t^j$ .  
7 Thus the model becomes

$$\underline{8} \quad y_t^j \sim \text{Bin}(y_t^j | \pi_t^j, N_t^j) \quad (31.47)$$

$$\underline{9} \quad \pi_t^j = \boldsymbol{\sigma} \left[ (\mathbf{z}_t^0)^\top \mathbf{u}_t^0 + (\mathbf{z}_t^j)^\top \mathbf{u}_t^j \right] \quad (31.48)$$

$$\underline{11} \quad \mathbf{z}_t^0 = \mathbf{z}_{t-1}^0 + \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}) \quad (31.49)$$

$$\underline{13} \quad \mathbf{z}_t^j = \mathbf{z}_{t-1}^j + \mathcal{N}(\mathbf{0}, \tau^2 \mathbf{I}) \quad (31.50)$$

14 For more details, see [Lin13b].

### 16 31.4.2 Bayesian SSM

18 SSMs can be quite sensitive to their parameter values, which is a particular concern when they are  
19 used for forecasting applications (see Section 31.2.4), or when the latent states or parameters are  
20 interpreted for scientific purposes (see e.g., [AM+16]). In such cases, it is wise to represent our  
21 uncertainty about the parameters by using Bayesian inference.

22 There are various algorithms we can use to perform this task. For linear-Gaussian SSMs, it is  
23 possible to use variational Bayes EM [Bea03; BC07] (see Section 10.2.5), or blocked Gibbs sampling  
24 [CK94b; CMR05; FS07] (see Section 12.3.7). In the latter case, we alternate between sampling from  
25  $p(\mathbf{z}_{1:T} | \mathbf{y}_{1:T}, \boldsymbol{\theta})$  using the forwards-filter backwards-sampling algorithm (Section 8.3.7), and sampling  
26 from  $p(\boldsymbol{\theta} | \mathbf{z}_{1:T}, \mathbf{y}_{1:T})$ , which is easy to do if we use conjugate priors. (Note, however, that  $\boldsymbol{\theta}$  and  $\mathbf{z}$  are  
27 highly correlated, so this method can be slow.)

28 For non-linear and/or non-Gaussian models, things get more complex. One approach is to perform  
29 joint inference of  $p(\boldsymbol{\theta}, \mathbf{z}_{1:T} | \mathbf{y}_{1:T})$  using HMC. However, this can be very slow, since the size of  $\mathbf{z}_{1:T}$   
30 is often very large.

31 Another approach is to use particle MCMC methods (Section 13.7). In this approach, we use  
32 Metropolis Hastings (e.g., with an adaptive Gaussian proposal) for  $p(\boldsymbol{\theta} | \mathbf{y})$ , and then we approximate  
33 the marginal likelihood  $p(\mathbf{y} | \boldsymbol{\theta}) = \int p(\mathbf{z}, \mathbf{y} | \boldsymbol{\theta}) d\mathbf{z}$  using a Monte Carlo method, such as SMC. (This is  
34 called a **pseudo marginal** method.) To make this efficient, we should use data-driven proposals, as  
35 we discussed in Section 13.4. For example, for exponential family likelihoods with linear Gaussian  
36 dynamics, we can use a Laplace approximation to “Gaussianize” each likelihood (see Section 13.4.2).  
37 For nonlinear models, we can use the EKF to linearize the model (see Section 13.4.3).

38

### 39 31.4.3 GP-SSM

40

41 In Section 31.3 we discussed parametric nonlinear SSMs. We can also represent nonlinear dynamics  
42 and/or observation models using a non-parametric Gaussian process (Chapter 18). This is known as  
43 an **GP-SSM**, which stands for “Gaussian process state-space model”. For details, see e.g., [WHF06;  
44 UFF06; TDR10; FCR14; Ele+17; SS17b].

45 Note that using a GP to define the dynamics or observations of a nonlinear SSM, in order to gain  
46 modeling power, is different to the task of converting a GP into a linear-Gaussian SSM, in order to  
47

gain speed. The latter can be done for GPs with certain kinds of 1d kernel,  $\mathcal{K}(x, x')$  for  $x, x' \in \mathbb{R}$ , such as the Matern kernel. See Section 18.5.6 for details.

## 31.5 Deep state space models

Traditional state-space models assume linear dynamics and linear observation models, both with additive Gaussian noise. This is obviously very limiting. In this section, we allow the dynamics and/or observation model to be modeled by nonlinear and/or non-Markovian deep neural networks; we call these **deep state space models**, also known as **dynamical variational autoencoders**. (To be consistent with the literature on VAEs, we denote the observations by  $\mathbf{x}_t$  instead of  $\mathbf{y}_t$ .) For a detailed review, see [Gir+21].

### 31.5.1 Deep Markov models

Suppose we create a SSM in which we use a deep neural network for the dynamics model and/or observation model; the result is called a **deep Markov model** [KSS17] or **stochastic RNN** [BO14; Fra+16]. (This is not quite the same as a variational RNN, which we explain in Section 31.5.4.)

We can fit a DMM using SVI (Section 10.3.2). The key is to infer the posterior over the latents. From the first-order Markov properties, the exact posterior is given by

$$p(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = p(\mathbf{z}_0) \prod_{t=1}^T p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{1:T}) = p(\mathbf{z}_0) \prod_{t=1}^T p(\mathbf{z}_t|\mathbf{z}_{t-1}, \underline{\mathbf{x}_{1:t-1}}, \mathbf{x}_{t:T}) \quad (31.51)$$

where the cancellation follows since  $\mathbf{z}_t \perp \mathbf{x}_{1:t-1} | \mathbf{z}_{t-1}$ , as pointed out in [KSS17].

In general, it is intractable to compute  $p(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$ , so we approximate it with an inference network. There are many choices for  $q$ . A simple one is a fully factorized model,  $q(\mathbf{z}_{1:T}) = \prod_t q(\mathbf{z}_t|\mathbf{x}_{1:t})$ . This is illustrated in Figure 31.16a. Since  $\mathbf{z}_t$  only depends on past data,  $\mathbf{x}_{1:t}$  (which is accumulated in the RNN hidden state  $\mathbf{h}_t$ ), we can use this inference network at run time for online inference. However, for training the model offline, we can use a more accurate posterior by using

$$q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = q(\mathbf{z}_0) \prod_{t=1}^T q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{1:T}) = q(\mathbf{z}_0) \prod_{t=1}^T q(\mathbf{z}_t|\mathbf{z}_{t-1}, \underline{\mathbf{x}_{1:t-1}}, \mathbf{x}_{t:T}) \quad (31.52)$$

Note that the dependence on past observation  $\mathbf{x}_{1:t-1}$  is already captured by  $\mathbf{z}_{t-1}$ , as in Equation (31.51). The dependencies on future observations,  $\mathbf{x}_{t:T}$ , can be summarized by a backwards RNN, as shown in Figure 31.16b. Thus

$$q(\mathbf{z}_{1:T}, \mathbf{h}_{1:T}|\mathbf{x}_{1:T}) = q(\mathbf{z}_0) \prod_{t=T}^1 \mathbb{I}(\mathbf{h}_t = f(\mathbf{h}_{t+1}, \mathbf{x}_t)) \prod_{t=1}^T q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{h}_t) \quad (31.53)$$

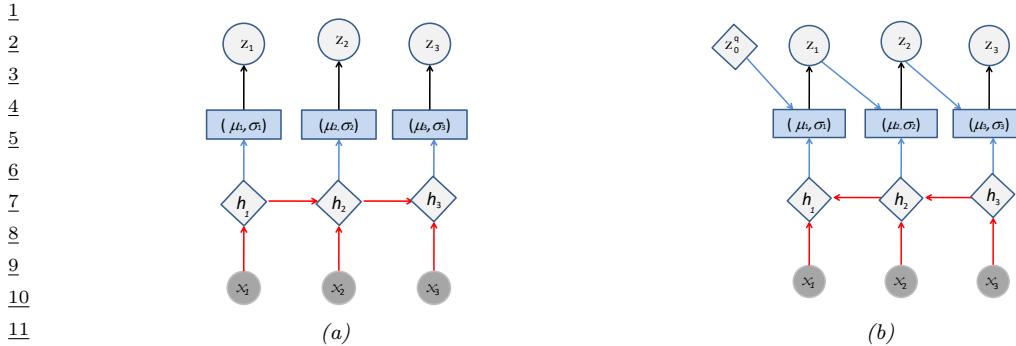


Figure 31.16: Inference networks for deep Markov model. (a) Fully factorized causal posterior  $q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_t q(\mathbf{z}_t|\mathbf{x}_{1:t})$ . The past observations  $\mathbf{x}_{1:t}$  are stored in the RNN hidden state  $\mathbf{h}_t$ . (b) Markovian posterior  $q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_t q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{t:T})$ . The future observations  $\mathbf{x}_{t:T}$  are stored in the RNN hidden state  $\mathbf{h}_t$ .

Given a fully factored  $q(\mathbf{z}_{1:T})$ , we can compute the ELBO as follows.

$$\log p(\mathbf{x}_{1:T}) = \log \left[ \sum_{\mathbf{z}_{1:T}} p(\mathbf{x}_{1:T}|\mathbf{z}_{1:T}) p(\mathbf{z}_{1:T}) \right] \quad (31.54)$$

$$= \log \mathbb{E}_{q(\mathbf{z}_{1:T})} \left[ p(\mathbf{x}_{1:T}|\mathbf{z}_{1:T}) \frac{p(\mathbf{z}_{1:T})}{q(\mathbf{z}_{1:T})} \right] \quad (31.55)$$

$$= \log \mathbb{E}_{q(\mathbf{z}_{1:T})} \left[ \prod_{t=1}^T \frac{p(\mathbf{x}_t|\mathbf{z}_t) p(\mathbf{z}_t|\mathbf{z}_{t-1})}{q(\mathbf{z}_t)} \right] \quad (31.56)$$

$$\geq \mathbb{E}_{q(\mathbf{z}_{1:T})} \left[ \sum_{t=1}^T \log p(\mathbf{x}_t|\mathbf{z}_t) + \log p(\mathbf{z}_t|\mathbf{z}_{t-1}) - \log q(\mathbf{z}_t) \right] \quad (31.57)$$

$$= \sum_{t=1}^T \mathbb{E}_{q(\mathbf{z}_t)} [\log p(\mathbf{x}_t|\mathbf{z}_t)] - \mathbb{E}_{q(\mathbf{z}_{t-1})} [D_{\text{KL}}(q(\mathbf{z}_t)\|p(\mathbf{z}_t|\mathbf{z}_{t-1}))] \quad (31.58)$$

If we assume that the variational posteriors are jointly Gaussian, we can use the reparameterization trick to use posterior samples to compute stochastic gradients of the ELBO. Furthermore, since we assumed a Gaussian prior, the KL term can be computed analytically.

### 31.5.2 Recurrent SSM

In a DMM, the observation model  $p(\mathbf{x}_t|\mathbf{z}_t)$  is first-order Markov, as is the dynamics model  $p(\mathbf{z}_t|\mathbf{z}_{t-1})$ . We can modify the model so that it captures long-range dependencies by adding deterministic hidden states as well. We can make the observation model depend on  $\mathbf{z}_{1:t}$  instead of just  $\mathbf{z}_t$  by using  $p(\mathbf{x}_t|\mathbf{h}_t)$ , where  $\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{z}_t)$ , so  $\mathbf{h}_t$  records all the stochastic choices. This is illustrated in Figure 31.17a. We can also make the dynamical prior depend on  $\mathbf{z}_{1:t-1}$  by replacing  $p(\mathbf{z}_t|\mathbf{z}_{t-1})$  with  $p(\mathbf{z}_t|\mathbf{h}_{t-1})$ , as is illustrated in Figure 31.17b. This is known as a **recurrent SSM**.

We can derive an inference network for an RSSM similar to the one we used for DMMs, except now we use a standard forwards RNN to compute  $q(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{x}_{1:t})$ .



Figure 31.17: Recurrent state space models. (a) Prior is first-order Markov,  $p(z_t|z_{t-1})$ , but observation model is not Markovian,  $p(x_t|h_t) = p(x_t|z_{1:t})$ , where  $h_t$  summarizes  $z_{1:t}$ . (b) Prior model is no longer first-order Markov either,  $p(z_t|h_{t-1}) = p(z_t|z_{1:t-1})$ . Diamonds are deterministic nodes, circles are stochastic.

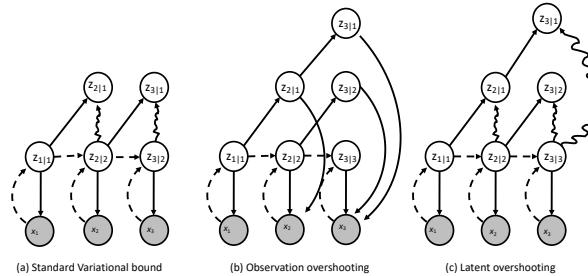


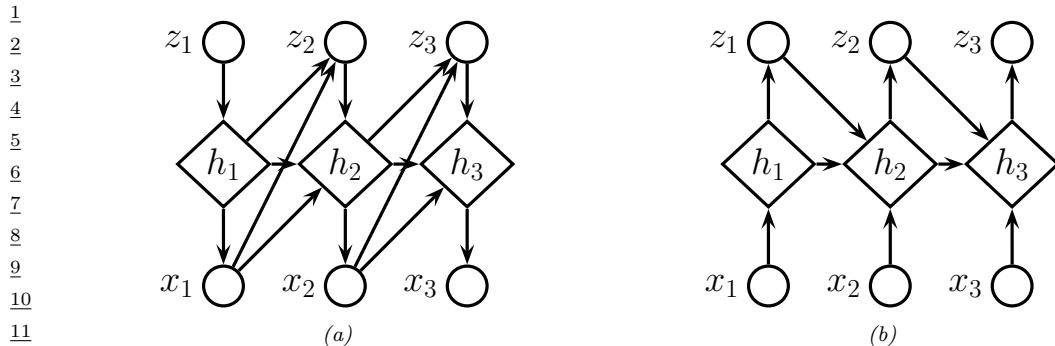
Figure 31.18: Unrolling schemes for SSMs. The labels  $s_{i:j}$  is shorthand for  $p(z_i|x_{1:j})$ . Solid lines denote the generative process, dashed lines the inference process. Arrows pointing at shaded circles represent log-likelihood loss terms. Wavy arrows indicate KL divergence loss terms. (a) Standard 1 step reconstruction of the observations. (b) Observation overshooting tries to predict future observations by unrolling in latent space. (c) Latent overshooting predicts future latent states and penalizes their KL divergence, but does not need to care about future observations. Adapted from Figure 3 of [Haf+19].

### 31.5.3 Improving multi-step predictions

In Figure 31.18(a), we show the loss terms involved in the ELBO. In particular, the wavy edge  $z_{t|t} \rightarrow z_{t|t-1}$  corresponds to  $\mathbb{E}_{q(z_{t-1})} [D_{\text{KL}}(q(z_t) \| p(z_t|z_{t-1}))]$ , and the solid edge  $z_{t|t} \rightarrow z_t$  corresponds to  $\mathbb{E}_{q(z_t)} [\log p(x_t|z_t)]$ . We see that the dynamics model,  $p(z_t|z_{t-1})$ , is only ever penalized in terms of how it differs from the one-step-ahead posterior  $q(z_t)$ , which can hurt the ability of the model to make long-term predictions.

One solution to this is to make multi-step forward predictions using the dynamics model, and use these to reconstruct future observations, and add these errors as extra loss terms. This is called **observation overshooting** [Amo+18], and is illustrated in Figure 31.18(b).

A faster approach, proposed in [Haf+19], is to apply a similar idea but in latent space. More precisely, let us compute the multi-step prediction model, by repeatedly applying the transition model and integrating out the intermediate states to get  $p(z_t|z_{t-d})$ . We can then compute the ELBO



<sup>13</sup>Figure 31.19: Variational RNN. (a) Generative model. (b) Inference model. The diamond-shaped nodes are deterministic.

17 for this as follows:

$$\log p_d(\mathbf{x}_{1:T}) \triangleq \log \int \prod_{t=1}^T p(\mathbf{z}_t | \mathbf{z}_{t-d}) p(\mathbf{x}_t | \mathbf{z}_t) d\mathbf{z}_{1:T} \quad (31.59)$$

$$\geq \sum_{t=1}^T \mathbb{E}_{q(\mathbf{z}_t)} [\log p(\mathbf{x}_t | \mathbf{z}_t)] - \mathbb{E}_{p(\mathbf{z}_{t-1} | \mathbf{z}_{t-d})q(\mathbf{z}_{t-d})} [D_{\mathbb{KL}}(q(\mathbf{z}_t) \| p(\mathbf{z}_t | \mathbf{z}_{t-1}))] \quad (31.60)$$

<sup>24</sup>To train the model so it is good at predicting at different future horizon depths  $d$ , we can average  
<sup>25</sup>the above over all  $1 \leq d \leq D$ . However, for computational reasons, we can instead just average  
<sup>26</sup>the KL terms, using weights  $\beta_d$ . This is called **latent overshooting** [Haf+19], and is illustrated in  
<sup>27</sup>Figure 31.18(c). The new objective becomes

$$\frac{28}{29} \quad \frac{1}{D} \sum_{d=1}^D \log p_d(\mathbf{x}_{1:T}) \geq \sum_{t=1}^T \mathbb{E}_{q(\mathbf{z}_t)} [\log p(\mathbf{x}_t | \mathbf{z}_t)] \quad (31.61)$$

$$-\frac{1}{D} \sum_{d=1}^D \beta_d \mathbb{E}_{p(\mathbf{z}_{t-1} | \mathbf{z}_{t-d}) q(\mathbf{z}_{t-d})} [D_{\mathbb{KL}}(q(\mathbf{z}_t) \| p(\mathbf{z}_t | \mathbf{z}_{t-1}))] \quad (31.62)$$

31.5.4 Variational RNNs

37A **variational RNN** (VRNN) [Chu+15] is similar to a recurrent SSM except the hidden states are generated conditional on all past hidden states *and* all past observations, rather than just the past hidden states. This is a more expressive model, but is slower to use for forecasting, since unrolling into the future requires generating observations  $\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \dots$  to “feed into” the hidden states, which controls the dynamics. This makes the model less useful for forecasting and model-based RL (see Section 37.4.5.2).

<sup>43</sup> More precisely, the generative model is as follows:

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \mathbf{h}_{1:T}) = \prod_{t=1}^T p(\mathbf{z}_t | \mathbf{h}_{t-1}, \mathbf{x}_{t-1}) \mathbb{I}(\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_{t-1}, \mathbf{z}_t)) p(\mathbf{x}_t | \mathbf{h}_t) \quad (31.63)$$

where  $p(\mathbf{z}_1|\mathbf{h}_0, \mathbf{x}_0) = p(\mathbf{z}_0)$  and  $\mathbf{h}_1 = f(\mathbf{h}_0, \mathbf{x}_0, \mathbf{z}_1) = f(\mathbf{z}_1)$ . Thus  $\mathbf{h}_t = (\mathbf{z}_{1:t}, \mathbf{x}_{1:t-1})$  is a summary of the past observations and past and current stochastic latent samples. If we marginalize out these deterministic hidden nodes, we see that the dynamical prior on the stochastic latents is  $p(\mathbf{z}_t|\mathbf{h}_{t-1}, \mathbf{x}_{t-1}) = p(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t-1})$ , whereas in a DMM, it is  $p(\mathbf{z}_t|\mathbf{z}_{t-1})$ , and in an RSSM, it is  $p(\mathbf{z}_t|\mathbf{z}_{1:t-1})$ . See Figure 31.19a for an illustration.

We can train VRNNs using SVI. In [Chu+15], they use the following inference network:

$$q(\mathbf{z}_{1:T}, \mathbf{h}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^T \mathbb{I}(\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{z}_{t-1}, \mathbf{x}_t)) q(\mathbf{z}_t|\mathbf{h}_t) \quad (31.64)$$

Thus  $\mathbf{h}_t = (\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t})$ . Marginalizing out these deterministic nodes, we see that the filtered posterior has the form  $q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_t q(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t})$ . See Figure 31.19b for an illustration. (We can also optionally replace  $\mathbf{x}_t$  with the output of a bidirectional RNN to get the smoothed posterior,  $q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_t q(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T})$ .)

This approach was used in [DF18] to generate simple videos of moving objects (e.g., a robot pushing a block); they call their method **stochastic video generation** or **SVG**. This was scaled up in [Vil+19], using simpler but larger architectures.

19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47



# 32 Graph learning

## 32.1 Introduction

Graphs are a very common way to represent data. In this chapter we discuss probability models for graphs. In Section 32.2, we assume the graph structure  $G$  is known, but we want to “explain” it in terms of a set of meaningful latent features; for this we use various kinds of latent variable models. In Section 32.3, we assume the graph structure  $G$  is unknown and needs to be inferred from correlated data,  $\mathbf{x}_n \in \mathbb{R}^D$ ; for this, we will use probabilistic graphical models with unknown topology. (See also Section 16.3.5, where we discuss graph neural networks, for performing supervised learning using graph-structured data.)

## 32.2 Latent variable models for graphs

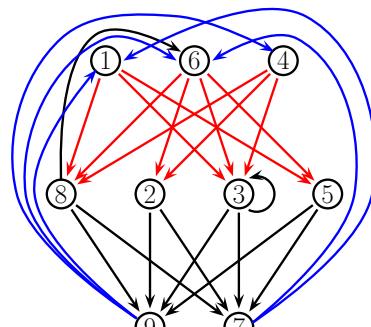
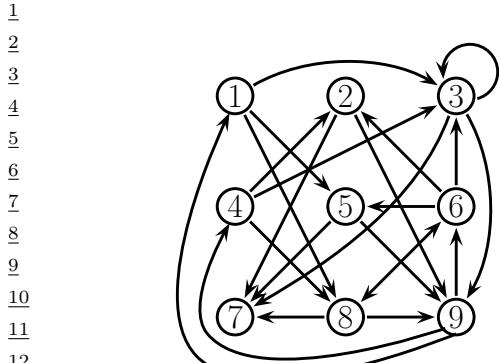
Graphs arise in many application areas, such as modeling social networks, protein-protein interaction networks, or patterns of disease transmission between people or animals. There are usually two primary goals when analysing such data: first, try to discover some “interesting structure” in the graph, such as clusters or communities; second, try to predict which links might occur in the future (e.g., who will make friends with whom). In this section, we focus on the former. More precisely, we will consider a variety of latent variable models for observed graphs.

### 32.2.1 Stochastic block model

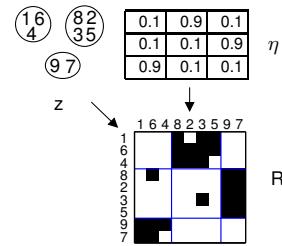
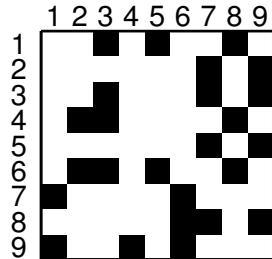
In Figure 32.1(a) we show a directed graph on 9 nodes. There is no apparent structure. However, if we look more deeply, we see it is possible to partition the nodes into three groups or blocks,  $B_1 = \{1, 4, 6\}$ ,  $B_2 = \{2, 3, 5, 8\}$ , and  $B_3 = \{7, 9\}$ , such that most of the connections go from nodes in  $B_1$  to  $B_2$ , or from  $B_2$  to  $B_3$ , or from  $B_3$  to  $B_1$ . This is illustrated in Figure 32.1(b).

The problem is easier to understand if we plot the adjacency matrices. Figure 32.2(a) shows the matrix for the graph with the nodes in their original ordering. Figure 32.2(b) shows the matrix for the graph with the nodes in their permuted ordering. It is clear that there is block structure.

We can make a generative model of block structured graphs as follows. First, for every node, sample a latent block  $q_i \sim \text{Cat}(\boldsymbol{\pi})$ , where  $\pi_k$  is the probability of choosing block  $k$ , for  $k = 1 : K$ . Second, choose the probability of connecting group  $a$  to group  $b$ , for all pairs of groups; let us denote this probability by  $\eta_{a,b}$ . This can come from a beta prior. Finally, generate each edge  $R_{ij}$  using the



16 *Figure 32.1: (a) A directed graph. (b) The same graph, with the nodes partitioned into 3 groups, making the  
17 block structure more apparent.*



30 *Figure 32.2: (a) Adjacency matrix for the graph in Figure 32.1(a). (b) Rows and columns are shown permuted  
31 to show the block structure. We also sketch of how the stochastic block model can generate this graph. From  
32 Figure 1 of [Kem+06]. Used with kind permission of Charles Kemp.*

33  
34  
35  
following model:

$$36 \quad p(R_{ij} = r | q_i = a, q_j = b, \boldsymbol{\eta}) = \text{Ber}(r | \eta_{a,b}) \quad (32.1)$$

37

38 This is called the **stochastic block model** [NS01]. Figure 32.4(a) illustrates the model as a DGM,  
39 and Figure 32.2(c) illustrates how this model can be used to cluster the nodes in our example.

40 Note that this is quite different from a conventional clustering problem. For example, we see  
41 that all the nodes in block 3 are grouped together, even though there are no connections between  
42 them. What they share is the property that they “like to” connect to nodes in block 1, and to receive  
43 connections from nodes in block 2. Figure 32.3 illustrates the power of the model for generating many  
44 different kinds of graph structure. For example, some social networks have hierarchical structure,  
45 which can be modeled by clustering people into different social strata, whereas others consist of a set  
46 of cliques.

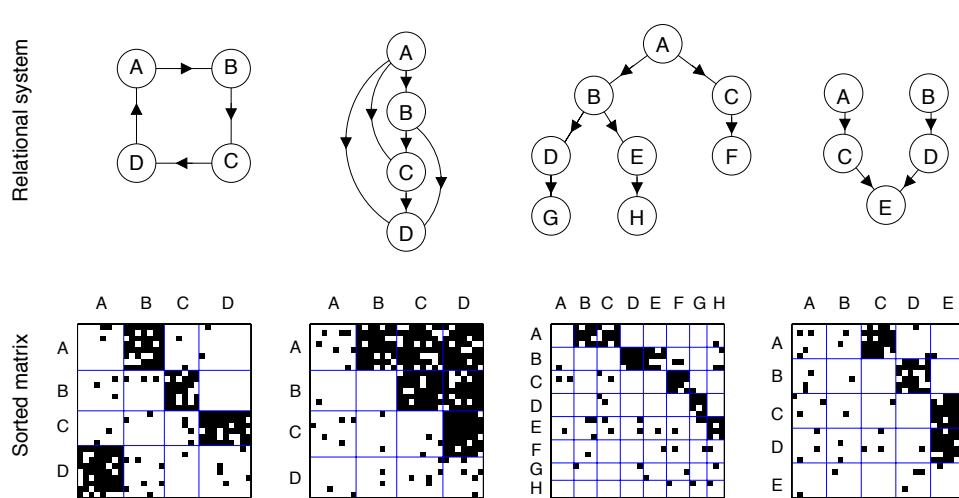


Figure 32.3: Some examples of graphs generated using the stochastic block model with different kinds of connectivity patterns between the blocks. The abstract graph (between blocks) represent a ring, a dominance hierarchy, a common-cause structure, and a common-effect structure. From Figure 4 of [Kem+10]. Used with kind permission of Charles Kemp.

Unlike a standard mixture model, it is not possible to fit this model using exact EM, because all the latent  $q_i$  variables become correlated. However, one can use variational EM [Air+08], collapsed Gibbs sampling [Kem+06], etc. We omit the details (which are similar to the LDA case).

In [Kem+06], they lifted the restriction that the number of blocks  $K$  be fixed, by replacing the Dirichlet prior on  $\pi$  by a Dirichlet process (see Section 33.2). This is known as the infinite relational model. See Section 32.2.3 for details.

If we have features associated with each node, we can make a discriminative version of this model, for example by defining

$$p(R_{ij} = r | q_i = a, q_j = b, \mathbf{x}_i, \mathbf{x}_j, \boldsymbol{\theta}) = \text{Ber}(r | \mathbf{w}_{a,b}^T f(\mathbf{x}_i, \mathbf{x}_j)) \quad (32.2)$$

where  $f(\mathbf{x}_i, \mathbf{x}_j)$  is some way of combining the feature vectors. For example, we could use concatenation,  $[\mathbf{x}_i, \mathbf{x}_j]$ , or elementwise product  $\mathbf{x}_i \otimes \mathbf{x}_j$  as in supervised LDA. The overall model is like a relational extension of the mixture of experts model.

### 32.2.2 Mixed membership stochastic block model

In [Air+08], they lifted the restriction that each node only belong to one cluster. That is, they replaced  $q_i \in \{1, \dots, K\}$  with  $\pi_i \in S_K$ . This is known as the **mixed membership stochastic block model**, and is similar in spirit to **fuzzy clustering** or **soft clustering**. Note that  $\pi_{ik}$  is not the same as  $p(z_i = k | \mathcal{D})$ ; the former represents **ontological uncertainty** (to what degree does each object belong to a cluster) whereas the latter represents **epistemological uncertainty** (which cluster does an object belong to). If we want to combine epistemological and ontological uncertainty, we can compute  $p(\pi_i | \mathcal{D})$ .

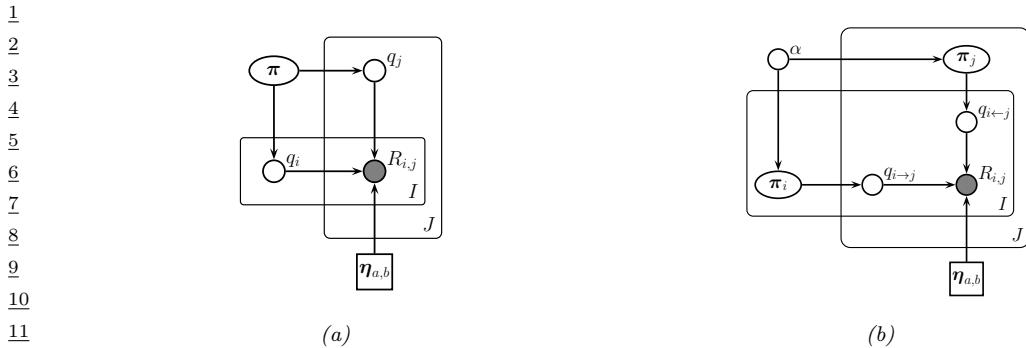


Figure 32.4: (a) Stochastic block model. (b) Mixed membership stochastic block model.

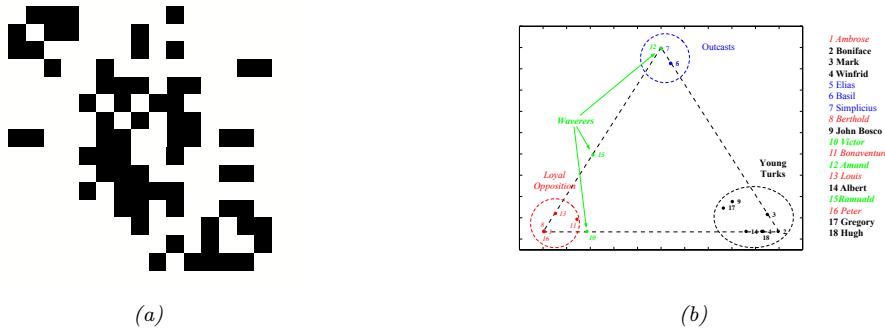


Figure 32.5: (a) Who-likes-whom graph for Sampson's monks. (b) Mixed membership of each monk in one of three groups. From Figures 2-3 of [Air+08]. Used with kind permission of Edo Airoldi.

In more detail, the generative process is as follows. First, each node picks a distribution over blocks,  $\pi_i \sim \text{Dir}(\boldsymbol{\alpha})$ . Second, choose the probability of connecting group  $a$  to group  $b$ , for all pairs of groups,  $\eta_{a,b} \sim \beta(\alpha, \beta)$ . Third, for each edge, sample two discrete variables, one for each direction:

$$q_{i \rightarrow j} \sim \text{Cat}(\pi_i), \quad q_{i \leftarrow j} \sim \text{Cat}(\pi_j) \tag{32.3}$$

Finally, generate each edge  $R_{ij}$  using the following model:

$$p(R_{ij} = 1 | q_{i \rightarrow j} = a, q_{i \leftarrow j} = b, \boldsymbol{\eta}) = \eta_{a,b} \tag{32.4}$$

See Figure 32.4(b) for the DGM.

Unlike the regular stochastic block model, each node can play a different role, depending on who it is connecting to. As an illustration of this, we will consider a data set that is widely used in the social networks analysis literature. The data concerns who-likes-whom amongst of group of 18 monks. It was collected by hand in 1968 by Sampson [Sam68] over a period of months. (These days, in the era of social media such as Facebook, a social network with only 18 people is trivially small, but the methods we are discussing can be made to scale.) Figure 32.5(a) plots the raw data, and Figure 32.5(b) plots  $\mathbb{E}[\pi_i]$  for each monk, where  $K = 3$ . We see that most of the monk belong

to one of the three clusters, known as the “young turks”, the “outcasts” and the “loyal opposition”. However, some individuals, notably monk 15, belong to two clusters; Sampson called these monks the “waverers”. It is interesting to see that the model can recover the same kinds of insights as Sampson derived by hand.

One prevalent problem in social network analysis is missing data. For example, if  $R_{ij} = 0$ , it may be due to the fact that person  $i$  and  $j$  have not had an opportunity to interact, or that data is not available for that interaction, as opposed to the fact that these people don’t want to interact. In other words, *absence of evidence is not evidence of absence*. We can model this by modifying the observation model so that with probability  $\rho$ , we generate a 0 from the background model, and we only force the model to explain observed 0s with probability  $1 - \rho$ . In other words, we robustify the observation model to allow for outliers, as follows:

$$p(R_{ij} = r | q_{i \rightarrow j} = a, q_{i \leftarrow j} = b, \boldsymbol{\eta}) = \rho \delta_0(r) + (1 - \rho) \text{Ber}(r | \eta_{a,b}) \quad (32.5)$$

See [Air+08] for details.

### 32.2.3 Infinite relational model

It is straightforward to extend the stochastic block model to model **relational data**: we just associate a latent variable  $q_i^t \in \{1, \dots, K_t\}$  with each entity  $i$  of each type  $t$ . We then define the probability of the relation holding between specific entities by looking up the probability of the relation holding between entities of that type. For example, if  $R : T^1 \times T^1 \times T^2 \rightarrow \{0, 1\}$ , we have

$$p(R(i, j, k) | q_i^1 = a, q_j^1 = b, q_k^2 = c, \boldsymbol{\eta}) = \text{Ber}(\eta_{a,b,c}) \quad (32.6)$$

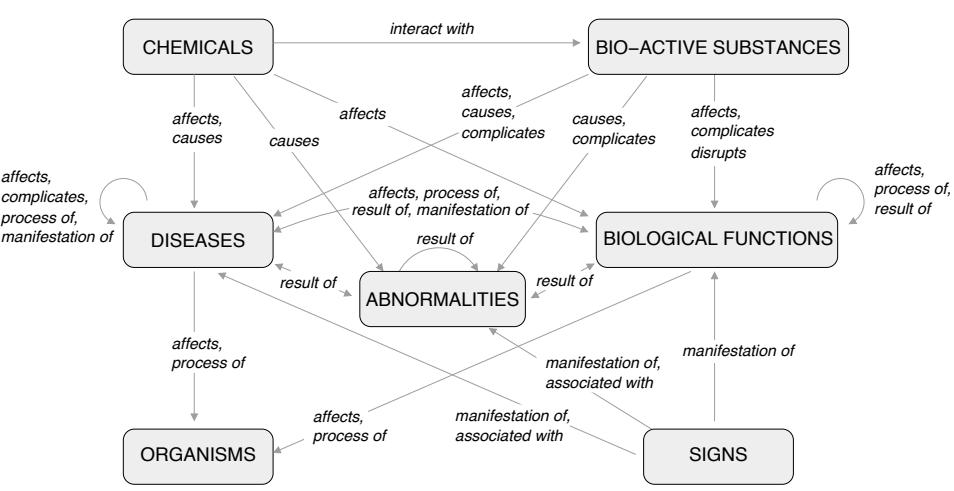
We can also have real-valued relations, where each edge has a weight. For example, we can write  $p(R(i, j, k) | q_i^1 = a, q_j^1 = b, q_k^2 = c, \boldsymbol{\mu}) = \mathcal{N}(\mu_{a,b,c} + \mu_i + \mu_j + \mu_k, \sigma^2)$ , where  $\mu_{a,b,c}$  captures the average response for that group of clusters, and  $\mu_i$ ,  $\mu_j$  and  $\mu_k$  are offsets for specific entities. (Allowing a different offset for every combination of  $i$ ,  $j$  and  $k$  would require too many parameters.) This model was proposed in [BBM07], who fit the model using an alternating minimization procedure.

If we allow the number of clusters  $K_t$  for each type of entity to be unbounded, by using a Dirichlet process, the model is called the **infinite relational model** (IRM) [Kem+06], also known as an **infinite hidden relational model** (IHRM) [Xu+06]. We can fit this model with variational Bayes [Xu+06; Xu+07] or collapsed Gibbs sampling [Kem+06]. Rather than go into algorithmic detail, we just sketch some interesting applications.

#### 32.2.3.1 Learning ontologies

An **ontology** refers to an organisation of knowledge. In AI, ontologies are often built by hand (see e.g., [RN10]), but it is interesting to try and learn them from data. In [Kem+06], they show how this can be done using the IRM.

The data comes from the Unified Medical Language System [McC03], which defines a semantic network with 135 concepts (such as “disease or syndrome”, “diagnostic procedure”, “animal”), and 49 binary predicates (such as “affects”, “prevents”). We can represent this as a ternary relation  $R : T^1 \times T^1 \times T^2 \rightarrow \{0, 1\}$ , where  $T^1$  is the set of concepts and  $T^2$  is the set of binary predicates. The result is a 3d cube. We can then apply the IRM to partition the cube into regions of roughly



*Figure 32.6: Illustration of an ontology learned by IRM applied to the Unified Medical Language System. The boxes represent 7 of the 14 concept clusters. Predicates that belong to the same cluster are grouped together, and associated with edges to which they pertain. All links with weight above 0.8 have been included. From Figure 9 of [Kem+10]. Used with kind permission of Charles Kemp.*

21

22

23 homogoneous response. The system found 14 concept clusters and 21 predicate clusters. Some of these  
 24 are shown in Figure 32.6. The system learns, for example, that biological functions affect organisms  
 25 (since  $\eta_{a,b,c} \approx 1$  where  $a$  represents the biological function cluster,  $b$  represents the organism cluster,  
 26 and  $c$  represents the affects cluster).

27

28

### 32.2.3.2 Clustering based on relations and features

30 We can also use IRM to cluster objects based on their relations and their features. For example,  
 31 [Kem+06] consider a political dataset (from 1965) consisting of 14 countries, 54 binary predicates  
 32 representing interaction types between countries (e.g., “sends tourists to”, “economic aid”), and 90  
 33 features (e.g., “communist”, “monarchy”). To create a binary dataset, real-valued features were  
 34 thresholded at their mean, and categorical variables were dummy-encoded. The data has 3 types:  $T^1$   
 35 represents countries,  $T^2$  represents interactions, and  $T^3$  represents features. We have two relations:  
 36  $R^1 : T^1 \times T^1 \times T^2 \rightarrow \{0, 1\}$ , and  $R^2 : T^1 \times T^3 \rightarrow \{0, 1\}$ . (This problem therefore combines aspects  
 37 of both the biclustering model and the ontology discovery model.) When given multiple relations,  
 38 the IRM treats them as conditionally independent. In this case, we have

39

$$40 \quad p(\mathbf{R}^1, \mathbf{R}^2 | \mathbf{q}^1, \mathbf{q}^2, \mathbf{q}^3, \boldsymbol{\theta}) = p(\mathbf{R}^1 | \mathbf{q}^1, \mathbf{q}^2, \boldsymbol{\theta}) p(\mathbf{R}^2 | \mathbf{q}^1, \mathbf{q}^3, \boldsymbol{\theta}) \quad (32.7)$$

41

42 The results are shown in Figure 32.7. The IRM divides the 90 features into 5 clusters, the first of  
 43 which contains “noncommunist”, which captures one of the most important aspects of this Cold-War  
 44 era dataset. It also clusters the 14 countries into 5 clusters, reflecting natural geo-political groupings  
 45 (e.g., US and UK, or the Communist Bloc), and the 54 predicates into 18 clusters, reflecting similar  
 46 relationships (e.g., “negative behavior” and “accusations”).

47

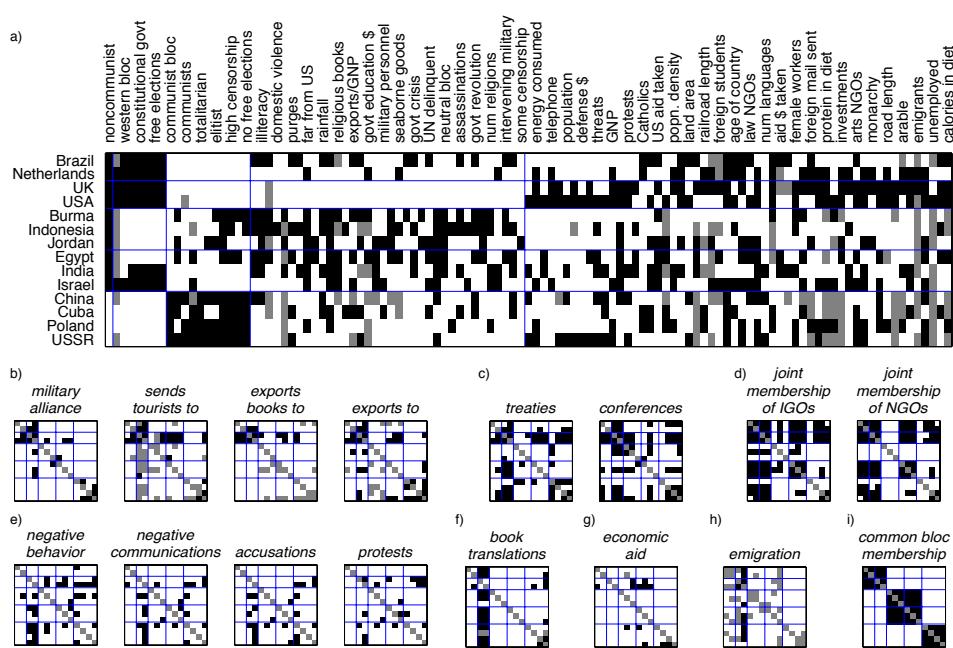


Figure 32.7: Illustration of IRM applied to some political data containing features and pairwise interactions. Top row (a): the partition of the countries into 5 clusters and the features into 5 clusters. Every second column is labelled with the name of the corresponding feature. Small squares at bottom (b-i): these are 8 of the 18 clusters of interaction types. From Figure 6 of [Kem+06]. Used with kind permission of Charles Kemp.

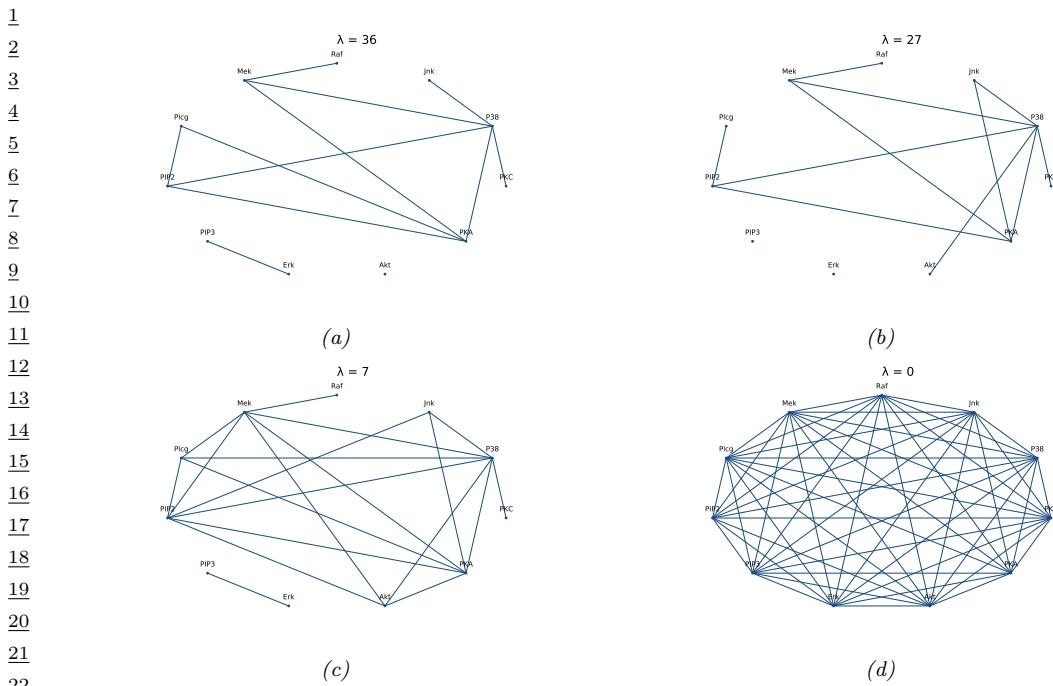
### 32.3 Graphical model structure learning

In this section, we discuss how to learn the structure of a probabilistic graphical model given sample observations of some or all of its nodes. That is, the input is an  $N \times D$  data matrix, and the output is a graph  $G$  (directed or undirected) with  $V$  nodes. (Usually  $V = D$ , but we also consider the case where we learn extra latent nodes that are not present in the input.)

#### 32.3.1 Applications

There are three main reasons to perform structure learning for PGMs: understanding, prediction, and causal inference (which involves both understanding and prediction), as we summarize below.

Learning sparse PGMs can be useful for gaining an understanding of multiple interacting variables. For example, consider a problem that arises in systems biology: we measure the phosphorylation status of some proteins in a cell [Sac+05] and want to infer how they interact. Figure 32.8 gives an example of a graph structure that was learned from this data, using a method called graphical lasso [FHT08; MH12], which is explained in the supplementary material. As another example, [Smi+06] showed that one can recover the neural “wiring diagram” of a certain kind of bird from multivariate time-series EEG data. The recovered structure closely matched the known functional connectivity of



23 *Figure 32.8: A sparse undirected Gaussian graphical model learned using graphical lasso applied to some flow*  
 24 *cytometry data (from [Sac+05]), which measures the phosphorylation status of 11 proteins. The sparsity level*  
 25 *is controlled by  $\lambda$ . (a)  $\lambda = 36$ . (b)  $\lambda = 27$ . (c)  $\lambda = 7$ . (d)  $\lambda = 0$ . Adapted from Figure 17.5 of [HTF09].*  
 26 *Generated by [gmm\\_lasso\\_demo.py](#).*

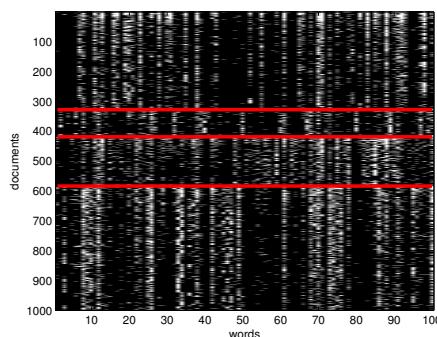
27  
28  
29  
30

31 this part of the bird brain.

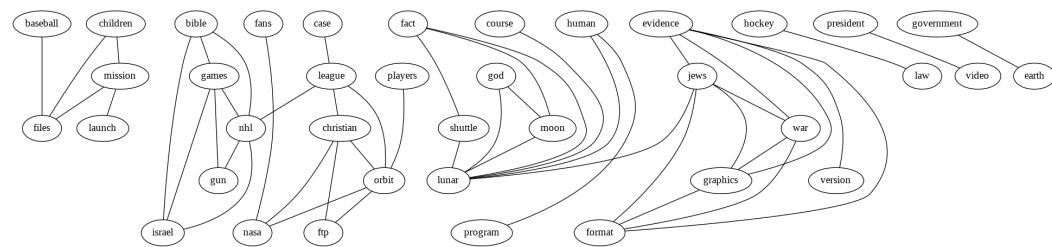
32 In some cases, we are not interested in interpreting the graph structure, we just want to use it to  
 33 make predictions. One example of this is in financial portfolio management, where accurate models of  
 34 the covariance between large numbers of different stocks is important. [CW07] show that by learning  
 35 a sparse graph, and then using this as the basis of a trading strategy, it is possible to outperform (i.e.,  
 36 make more money than) methods that do not exploit sparse graphs. Another example is predicting  
 37 traffic jams on the freeway. [Hor+05] describe a deployed system called JamBayes for predicting  
 38 traffic flow in the Seattle area, using a directed graphical model whose structure was learned from  
 39 data.

40 Structure learning is also an important pre-requisite for causal inference. In particular, to predict  
 41 the effects of interventions on a system, or to perform counterfactual reasoning, we need to know the  
 42 structural causal model (SCM), as we discuss in Section 4.6. An SCM is a kind of directed graphical  
 43 model where the relationships between nodes are deterministic (functional), except for stochastic  
 44 root (exogeneous) variables. Consequently one can use techniques for learning DAG structures as a  
 45 way to learn SCMs, if we make some assumptions about (lack of) confounders. This is called **causal**  
 46 **discovery**.

47



13 *Figure 32.9: Subset of size  $16242 \times 100$  of the 20-newsgroups data. We only show 1000 rows, for clarity.*  
 14 *Each row is a document (represented as a bag-of-words bit vector), each column is a word. The red lines*  
 15 *separate the 4 classes, which are (in descending order) comp, rec, sci, talk (these are the titles of USENET*  
 16 *groups). We can see that there are subsets of words whose presence or absence is indicative of the class. The*  
 17 *data is available from <http://cs.nyu.edu/~roweis/data.html>. Generated by newsgroupsVisualize.py.*



27 *Figure 32.10: Part of a relevance network constructed from the 20 newsgroup data. data shown in Figure 32.9.*  
 28 *We show edges whose mutual information is greater than or equal to 20% of the maximum pairwise MI.*  
 29 *For clarity, the graph has been cropped, so we only show a subset of the nodes and edges. Generated by*  
 30 *relevance\_network\_newsgroup\_demo.py.*

### 32.3.2 Relevance networks

35 If we are just interested in learning a graph structure that captures some properties of the data,  
 36 rather than learning a full generative model, a simple approach is to compute a **relevance network**,  
 37 in which we add an  $i - j$  edge if the pairwise mutual information  $\mathbb{I}(X_i; X_j)$  is above some threshold.  
 38 In the Gaussian case,  $\mathbb{I}(X_i; X_j) = -\frac{1}{2} \log(1 - \rho_{ij}^2)$ , where  $\rho_{ij}$  is the correlation coefficient, and the  
 39 resulting graph is called a **covariance graph** (Section 4.4.4.1). However, we can also apply it to  
 40 discrete random variables.

41 Relevance networks are quite popular in systems biology [Mar+06], where they are used to visualize  
 42 the interaction between genes. But they can also be applied to other kinds of datasets. For example,  
 43 Figure 32.10 visualizes the MI between words in the 20 newsgroup dataset shown in Figure 32.9. The  
 44 results seem intuitively reasonable.

45 However, relevance networks suffer from a major problem: the graphs are usually very dense, since  
 46 most variables are dependent on most other variables, even after thresholding the MIs. For example,

1 suppose  $X_1$  directly influences  $X_2$  which directly influences  $X_3$  (e.g., these form components of a  
2 signalling cascade,  $X_1 - X_2 - X_3$ ). Then  $X_1$  has non-zero MI with  $X_3$  (and vice versa), so there  
3  
4 will be a  $1 - 3$  edge as well as the  $1 - 2$  and  $2 - 3$  edges; thus the graph may be fully connected,  
5 depending on the threshold.

6 A solution to this is to learn a PGM, which represents conditional *independence*, rather than  
7 *dependence*. In the chain example, there will not be a  $1 - 3$  edge, since  $X_1 \perp X_3 | X_2$ . Consequently  
8 graphical models are usually much sparser than relevance networks.  
9

### 10 32.3.3 Learning sparse PGMs

11 The details on algorithms for learning sparse PGM structure can be found in the supplementary  
12 material, due to space constraints.  
13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

# 33 Non-parametric Bayesian models

*This chapter is written by Vinayak Rao.*

## 33.1 Introduction

A **stochastic process** is a probability distribution over a potentially infinite set of random variables. A simple example is a **Markov process** (Section 2.8), in which the variables are time indexed. This defines the distribution  $p(x_1, \dots, x_T)$  for any  $T$ , where  $x_t \in \mathcal{X}$ , using the form  $p(\mathbf{x}) = p(x_1) \prod_{t=2}^T p(x_t|x_{t-1})$ , where  $\mathbf{x} = (x_1, \dots, x_T)$ . Another example is a **Gaussian process**, specified by a mean function  $\mu$  and a positive definite kernel function  $\mathcal{K}$ . This defines a probability distribution over an unknown function  $f : \mathcal{X} \rightarrow \mathbb{R}$ , with the joint distribution  $p(f(\mathbf{x})) = \mathcal{N}(f(\mathbf{x})|\boldsymbol{\mu}(\mathbf{x}), \mathbf{K}(\mathbf{x}))$ , where  $f(\mathbf{x}) = [f(x_1), \dots, f(X_T)]$  evaluates this function at each input,  $\boldsymbol{\mu}(\mathbf{x}) = [\mu(x_1), \dots, \mu(X_T)]$  is the mean, and  $\mathbf{K}(\mathbf{x}, \mathbf{x}) = [\mathcal{K}(x_i), \mathcal{K}(x_j)]$  is the Gram matrix. (See Chapter 18 for details.)

In this chapter, we discuss other kinds of stochastic processes commonly used in a subfield of Bayesian statistics called **Bayesian nonparametrics**. The defining characteristic of a parametric model is that the objects being modeled, whether regression or classification functions, probability densities, or something more modern like graphs or shapes, are indexed by a finite-dimensional parameter vector. For instance, linear regression restricts itself to linear functions, and for scalar inputs, indexes these functions with two parameters, a slope and an intercept. In a parametric Bayesian model, a prior probability distribution on these parameters is used to define a prior distribution on the objects of interest. Bayesian nonparametric models directly place prior distributions on objects of interest, typically via some stochastic process, realizations of which have ‘infinite complexity’. For instance, unlike a straight line, a function drawn from a Gaussian process typically cannot be summarized by a finite number of parameters. This allows the statistical complexity of inferences and predictions to grow with the size of the training datasets, avoiding underfitting. By taking a Bayesian approach, the danger of overfitting is minimized: one maintains a full posterior distribution over the infinite parameters, rather than trying to fit them using a finite dataset. Despite involving infinite-parameter objects, practitioners are often only interested in inferences on a finite training dataset and predictions on a finite test dataset. This often allows these models to be surprisingly tractable. Nonparametric Bayesian models thus present an elegant synthesis of probabilistic, statistical and computational ideas.

## 33.2 Dirichlet process

Recall from Chapter 18 that a Gaussian process is a nonparametric probability distribution over functions  $f$ . We saw how in Bayesian settings, this can be used as a nonparametric prior for supervised learning tasks like regression and classification. By contrast, a **Dirichlet process** (DP) is a nonparametric probability distribution over probability distributions, and is useful as a flexible prior for unsupervised learning tasks like clustering and density modeling [Fer73]. We give more details below.

### 33.2.1 Definition

Let  $G$  be a probability distribution or a probability measure (we will use the latter terminology in this chapter) on some space  $\Theta$ . Recall that a probability measure is a function that assigns values to subsets  $T \subseteq \Theta$  satisfying the usual axioms of probability:  $0 \leq G(T) \leq 1$ ,  $G(\Theta) = \int_{\Theta} G(\theta) d\theta = 1$ , and for disjoint subsets  $T_1, \dots, T_K$  of  $\Theta$ ,  $G(T_1 \cup \dots \cup T_K) = \sum_{k=1}^K G(T_k)$ . Bayesian unsupervised learning now seeks to place a prior on the probability measure  $G$ .

We have already seen examples of *parametric* priors over probability measures. As a simple example, consider a Gaussian distribution  $\mathcal{N}(\theta|\mu, \sigma^2)$ : this is a probability measure on  $\Theta$ , and by placing priors on the parameters  $\mu$  and  $\sigma^2$ , we have a **parametric prior** on probability measures. Mixture models form more flexible priors, allowing multimodality and asymmetry, and are parametrized by the probabilities of the mixture components, as well as their parameters. DPs directly define a probability on probability measures  $G$ .

A Dirichlet Process is specified by a positive real number  $\alpha$ , called the **concentration parameter**, and a probability measure  $H$ , called the **base measure**. We write a random measure drawn from a DP as  $G \sim \text{DP}(\alpha, H)$ .  $H$  is typically a standard probability measure on  $\Theta$ , and forms the mean of the Dirichlet process. That is, if  $G \sim \text{DP}(\alpha, H)$ , then for any subset  $T$  of  $\Theta$ ,  $\mathbb{E}[G(T)] = H(T)$ . The parameter  $\alpha$  measures how concentrated the Dirichlet process is around  $H$ , with  $\mathbb{V}[G(T)] = \frac{H(T)(1-H(T))}{1+\alpha}$ . If  $\Theta$  is  $\mathbb{R}^2$ , then setting  $H$  to the bivariate normal  $\mathcal{N}(0, I_2)$  and  $\alpha$  to a large value implies a prior belief that  $G$  sampled from  $\text{DP}(\alpha, H)$  is close to the normal, whereas a small  $\alpha$  represents a relatively uninformative prior.

We now define the Dirichlet process more precisely. Let  $(T_1, \dots, T_K)$  be a finite partition of  $\Theta$ , that is,  $(T_1, \dots, T_K)$  are disjoint sets whose union is  $\Theta$ . For a probability measure  $G$ , let  $(G(T_1), \dots, G(T_K))$  be the vector of probabilities of the elements of this partition. Then  $\text{DP}(\alpha, H)$  is a prior over probability measures  $G$  satisfying the following requirement: for any finite partition, the associated vector of probabilities has the following joint Dirichlet distribution:

$$(33.1) \quad (G(T_1), \dots, G(T_K)) \sim \text{Dir}(\alpha H(T_1), \dots, \alpha H(T_K)).$$

Just like the Gaussian process, the DP is defined implicitly through a set of finite-dimensional distributions, in this case through the distribution of  $G$  projected onto any finite partition. The finite-dimensional distributions are **consistent** in the following sense: if  $T_{11}$  and  $T_{12}$  form a partition of  $T_1$ , then one can sample  $G(T_1)$  in two ways: directly, by sampling

$$(33.2) \quad (G(T_1), \dots, G(T_K)) \sim \text{Dir}(\alpha H(T_1), \dots, \alpha H(T_K))$$

or, indirectly, by sampling

$$(33.3) \quad (G(T_{11}), G(T_{12}), \dots, G(T_K)) \sim \text{Dir}(\alpha H(T_{11}), \alpha H(T_{12}), \dots, \alpha H(T_K))$$

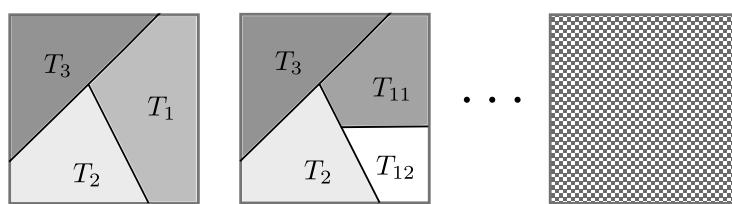


Figure 33.1: Partitions of the unit square. (left) One possible partition into  $K = 3$  regions, and (center) A refined partition into  $K = 4$  regions. In both figures, the shading of cell  $T_k$  is proportional  $G(T_k)$ , resulting from the same realization of a Dirichlet process. (right) An ‘infinite partition’ of the unit square. The Dirichlet process can informally be viewed as an infinite-dimensional Dirichlet distribution defined on this.

and then setting  $G(T_1) = G(T_{11}) + G(T_{12})$ . From the properties of the Dirichlet distribution,  $G(T_1)$  sampled either way follows the same distribution. This consistency property implies, via Kolmogorov’s extension theorem [Kal06], that underlying all finite-dimensional probability vectors for different partitions is a single infinite-dimensional vector that we could informally write as

$$G(d\theta_1), \dots, G(\theta_\infty) \sim \text{Dir}(\alpha H(d\theta_1), \dots, \alpha H(d\theta_\infty)). \quad (33.4)$$

Very roughly, this ‘infinite-dimensional Dirichlet distribution’ is the Dirichlet Process. Figure 33.1 sketches this out.

Why is the Dirichlet process, defined in this indirect fashion, useful to practitioners? The answer has to do with conjugacy properties that it inherits from the Dirichlet distribution. One of the simplest unsupervised learning problems seeks to learn an unknown probability distribution  $G$  from i.i.d. samples  $\{\bar{\theta}_1, \dots, \bar{\theta}_N\}$  drawn from it. Consider placing a DP prior on the unknown  $G$ . Then given the data, one is interested in the posterior distribution over  $G$ , representing the updated probability distribution over  $G$ . For a partition  $(T_1, \dots, T_K)$  of  $\Theta$ , an observation falls into the cell  $z$  following a multinoulli distribution:

$$z \sim \text{Cat}(G(T_1), \dots, G(T_K)). \quad (33.5)$$

Under a DP prior on  $G$ ,  $(G(T_1), \dots, G(T_K))$  follows a Dirichlet distribution (equation (33.1)). From the Dirichlet-multinomial conjugacy, the posterior for  $(G(T_1), \dots, G(T_K))$  given the observations is

$$(G(T_1), \dots, G(T_K)) | \{\bar{\theta}_1, \dots, \bar{\theta}_N\} \sim \text{Dir}(G(T_1) + \sum_{i=1}^N \mathbb{I}(\bar{\theta}_i \in T_1), \dots, G(T_K) + \sum_{i=1}^N \mathbb{I}(\bar{\theta}_i \in T_K)) \quad (33.6)$$

This is true for any finite partition, so that following our earlier definition, the posterior over  $G$  itself is a Dirichlet process, and it is easy to see that:

$$G | \bar{\theta}_1, \dots, \bar{\theta}_N, \alpha, H \sim \text{DP} \left( \alpha + N, \frac{1}{\alpha + N} \left( \alpha H + \sum_{i=1}^N \delta_{\theta_i} \right) \right). \quad (33.7)$$

Thus we see that the DP prior on  $G$  is a conjugate prior for i.i.d. observations from  $G$ , with the posterior distribution over  $G$  also a Dirichlet process with concentration parameter  $\alpha + N$ , and base

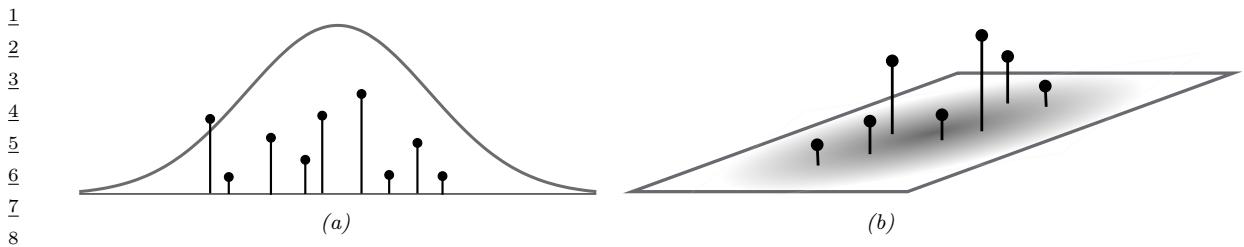
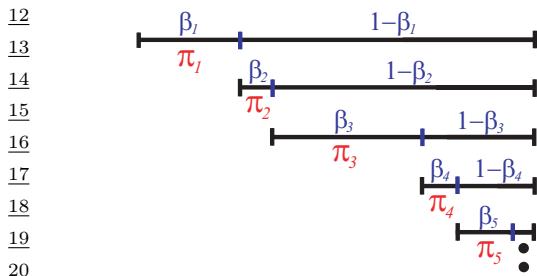
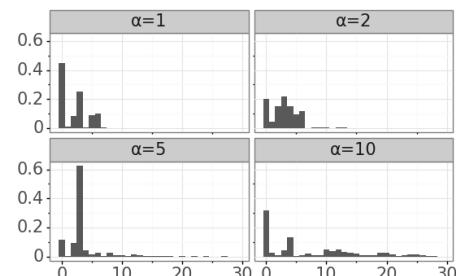


Figure 33.2: Realizations from a Dirichlet process when  $\Theta$  is (a) the real line, and (b) the unit square. Also shown are the base measures  $\alpha H$ . In reality, the number of atoms is infinite for both cases.

11

21  
22

(a)



(b)

Figure 33.3: Illustration of the stick breaking construction. (a) We have a unit length stick, which we break at a random point  $\beta_1$ ; the length of the piece we keep is called  $\pi_1$ ; we then recursively break off pieces of the remaining stick, to generate  $\pi_2, \pi_3, \dots$ . From Figure 2.22 of [Sud06]. Used with kind permission of Erik Sudderth. (b) Samples of  $\pi_k$  from this process for different values of  $\alpha$ . Generated by [stick\\_breaking\\_demo.py](#).

27

28

measure a convex combination of the original base measure  $H$  and the empirical distribution of the observations. Note that as  $N$  increases, the influence of the original base measure  $H$  starts to wane, and the posterior base measure becomes closer and closer to the empirical distribution of the observations. At the same time, the concentration parameter increases, suggesting that the posterior distribution concentrates around the empirical distribution.

34

### 33.2.2 Stick breaking construction of the DP

Our discussion so far has been very abstract, with no indication of how to either sample the random measure  $G$  or how to sample observations from  $G$ . We address the first question, giving a constructive definition for the DP known as the **stick-breaking construction** [Set94].

We first mention that probability measures  $G$  sampled from a DP are **discrete with probability one** (see Figure 33.2), taking the form

42

$$G(\theta) = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}(\theta). \quad (33.8)$$

45

Thus  $G$  consists of an infinite number of **atoms**, the  $k$ th atom located at  $\theta_k$ , and having weight  $\pi_k$ .

Informally, this follows from Equation (33.4), which represents the DP as an infinite-dimensional but infinitely-sparse Dirichlet distribution (recall that as its parameters become smaller, a Dirichlet distribution concentrates on sparse distributions that are dominated by a few components).

For a DP, the locations  $\theta_k$  of the atoms are drawn independently from the base measure  $H$ , whereas the concentration parameter  $\alpha$  controls the distribution of the weights  $\pi_k$ . Observe that the infinite sequence of weights  $(\pi_1, \pi_2, \dots)$  must add up to one, since  $G$  is a probability measure. The weights can be simulated by the following process sketched in Figure 33.3, and known as the **stick-breaking process**. Start with a stick of length 1 representing the total probability mass, and sequentially break off a random Beta( $1, \alpha$ ) distributed fraction of the remaining stick. The  $k$ th break forms  $\pi_k$ . In equations, for  $k = 1, 2, \dots$ ,

$$\beta_k \sim \text{Beta}(1, \alpha), \quad \theta_k \sim H, \quad (33.9)$$

$$\pi_k = \beta_k \prod_{l=1}^{k-1} (1 - \beta_l) = \beta_k \left(1 - \sum_{l=1}^{k-1} \pi_l\right) \quad (33.10)$$

Then, setting  $G(\theta) = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}(\theta)$ , one can show that  $G \sim \text{DP}(\alpha, H)$ . The distribution over the weights is often denoted by

$$\pi \sim \text{GEM}(\alpha), \quad (33.11)$$

where GEM stands for Griffiths, Engen and McCloskey (this term is due to [Ewe90]). Some samples from this process are shown in Figure 33.3.

We note that since the number of atoms is infinite, one cannot exactly simulate from a DP in finite time. However, the sequence of weights from the GEM distribution are **stochastically ordered**, having decreasing averages, and the truncation error resulting from terminating after a finite number of steps quickly becoming negligible [IJ01]. Nevertheless, we will see in the next section that it is possible to simulate samples and make predictions from a DP-distributed probability measure  $G$  without any truncation error. This exploits the conjugacy property of the DP.

### 33.2.3 The Chinese restaurant process (CRP)

Consider a single observation  $\bar{\theta}_1$  from a DP-distributed probability measure  $G$ . The probability that  $\bar{\theta}_1$  lies within a set  $T \in \Theta$ , marginalizing out the random  $G$ , is  $\mathbb{E}[G(T)] = H(T)$ , the equality following from the definition of the DP. This holds for arbitrary  $T$ , which implies that the first observation  $\bar{\theta}_1$  is distributed as the base measure of the DP:

$$p(\bar{\theta}_1 = \theta | \alpha, H) = H(\theta). \quad (33.12)$$

Given  $N$  observations  $\bar{\theta}_1, \dots, \bar{\theta}_N$ , the updated distribution over  $G$  is still a DP, but now modified as in Equation (33.7). Repeating the same argument, it follows that the  $(N + 1)$ st observation is distributed as the base measure of the posterior DP, given by

$$p(\bar{\theta}_{N+1} = \theta | \bar{\theta}_{1:N}, \alpha, H) = \frac{1}{\alpha + N} \left( \alpha H(\theta) + \sum_{k=1}^N N_k \delta_{\bar{\theta}_k}(\theta) \right) \quad (33.13)$$

1 where  $N_k$  is the number of observations equal to  $\bar{\theta}_k$ . The previous two equations form the basis of  
2 what is called the **Pólya urn** or **Blackwell-MacQueen** sampling scheme [BM+73]. This provides  
3 a way to exactly produce samples from a DP-distributed random probability measure.

4 It is often more convenient to work with discrete variables  $(z_1, \dots, z_N)$ , with  $z_i$  specifying which  
5 value of  $\theta_k$  the  $i$ th sample takes. In particular, for the  $i$ th observation,  $\bar{\theta}_i = \theta_{z_i}$ . This allows us to  
6 decouple the cluster or partition structure of the dataset (controlled by  $\alpha$ ) and the cluster parameters  
7 (controlled by  $H$ ). Let us assign the first observation to cluster 1, i.e.  $z_1 = 1$ . The second observation  
8 can either belong to the same cluster as observation 1, or belong to a new cluster, which we call  
9 cluster 2. In the former event,  $z_2 = 1$ , after which  $z_3$  can equal 1 or 2. In the latter event,  $z_2 = 2$ ,  
10 and  $z_3$  can equal 1, 2 or 3. Based on the Equation (33.13), we have

12

13

$$\underline{14} \quad p(z_{N+1} = z | \mathbf{z}_{1:N}, \alpha) = \frac{1}{\alpha + N} \left( \alpha \mathbb{I}(z = K + 1) + \sum_{k=1}^K N_k \mathbb{I}(z = k) \right), \quad (33.14)$$

15

16

17 assuming the first  $N$  observations have been assigned to  $K$  clusters. This is called the **Chinese**  
18 **restaurant process** or **CRP**, based on the following analogy: observations are customers in a  
19 restaurant with an infinite number of tables, each corresponding to a different cluster. Each table  
20 has a dish, corresponding to the parameter  $\theta$  of that cluster. When a customer enters the restaurant,  
21 they may choose to join an existing table with probability proportional to the number of people  
22 already sitting at this table (i.e. they join table  $k$  with probability proportional to  $N_k$ ); otherwise,  
23 with probability proportional to  $\alpha$ , they choose to sit at a new table, ordering a new dish by sampling  
24 from the base measure  $H$ .

25 The sequence  $Z = (z_1, \dots, z_N)$  of cluster assignments is **partition of the integers** 1 to  $N$ , and  
26 the CRP is a distribution over such partitions. The probability of creating a new table diminishes as  
27 the number of observations increases, but is always non-zero, and one can show that the number of  
28 occupied tables  $K$  approaches  $\alpha \log(N)$  as  $N \rightarrow \infty$  almost surely. The fact that currently occupied  
29 tables are more likely to get new customers is sometimes called a **rich get richer** phenomenon.  
30 It is important to recognize that despite being defined as a sequential process, the CRP is an  
31 **exchangeable process**, with partition probabilities that are independent of the observation indices.  
32 Indeed, it is easy to show that the probability of a partition of  $N$  integers into  $K$  clusters with  
33 sizes  $N_1, \dots, N_K$  is

34

35

$$\underline{36} \quad p(n_1, \dots, n_k) = \frac{\alpha^{K-1}}{[\alpha + 1]_1^N} \prod_{k=1}^K N_k!. \quad (33.15)$$

37

38

39 Here,  $[\alpha + 1]_1^N = \prod_{i=0}^{N-1} (\alpha + 1 + i)$  is the rising factorial. Equation (33.15) depends only on the  
40 cluster sizes, and is called the Ewens sampling formula [Ewe72]. Exchangeability implies that the  
41 probability that the first two customers sit at the same table is the same as the probability that the  
42 first and last sit at the same table. Similarly all customers have the same probability of ending up in  
43 a cluster of size  $S$ . The fact that the first customer can only belong to cluster 1 (i.e. that  $z_1 = 1$ )  
44 does not contradict exchangeability and reflects the fact that the cluster indices are chosen arbitrarily.  
45 This disappears if we index clusters by their associated parameter  $\theta_k$ .

47

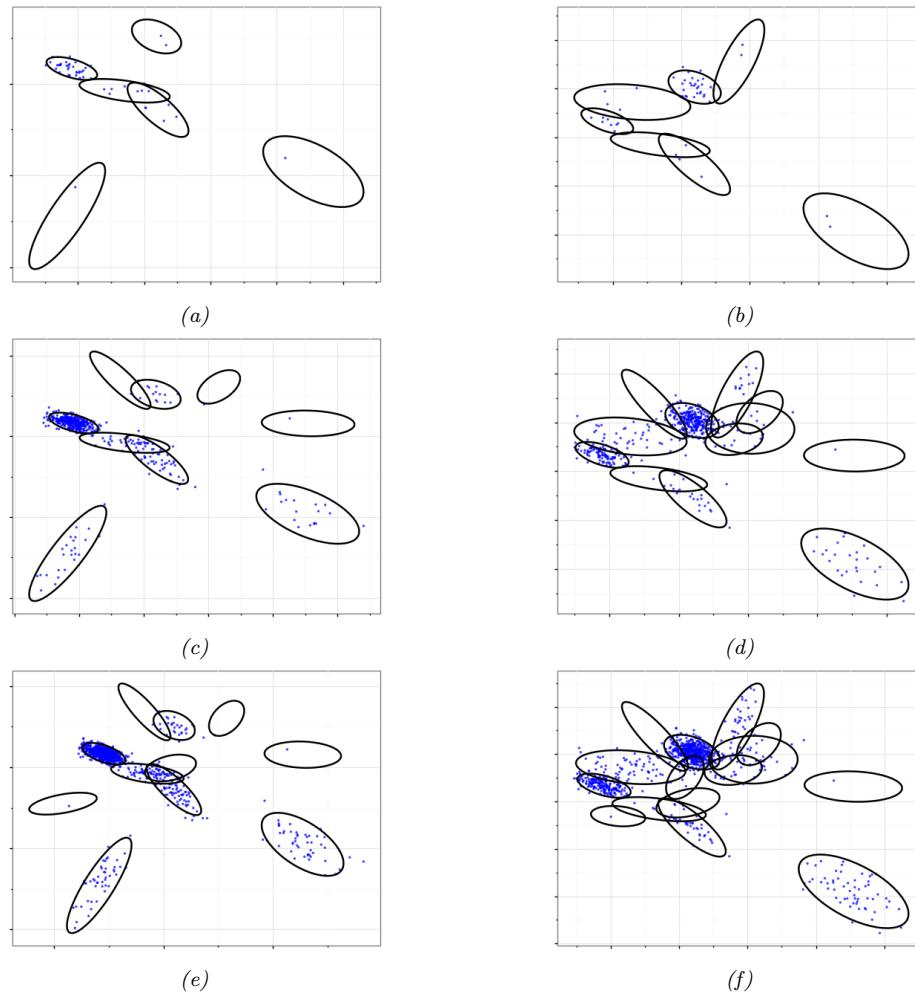
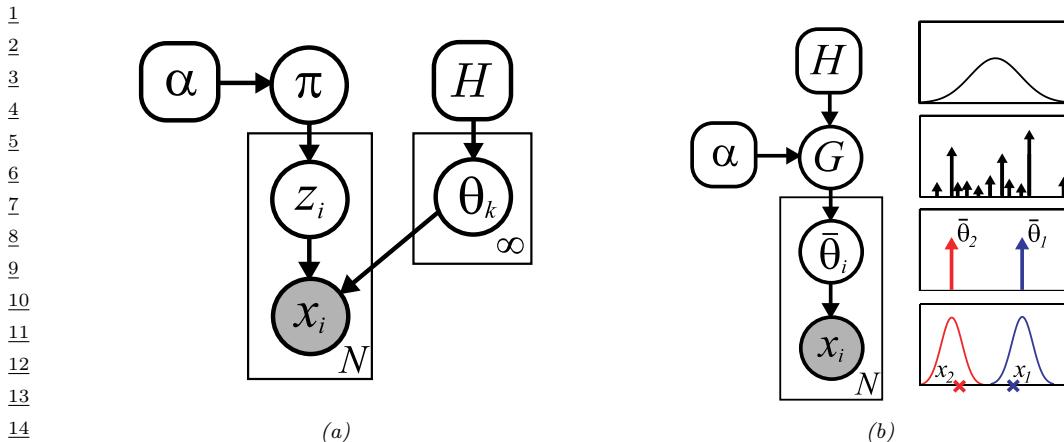


Figure 33.4: Some samples from a Dirichlet process mixture model of 2D Gaussians, with concentration parameter  $\alpha = 1$  (left column) and  $\alpha = 2$  (right column). From top to bottom, we show  $N = 50$ ,  $N = 500$  and  $N = 1000$  samples. We also show the model parameters as ellipses, which are sampled from a NIW base distribution. Generated by [dpmSampleDemo.py](#).

#### 33.2.4 Dirichlet process mixture models

Real-world datasets are often best modeled by continuous probability densities. By contrast, a sample  $G$  from a DP is discrete with probability one, and sampling observations from  $G$  will result in repeated values, making it inappropriate for many applications. However, the discrete structure of  $G$  is useful in clustering applications, as a prior for the latent variables underlying the observed datapoints. In particular,  $z_i$  and  $\bar{\theta}_i$  can represent the cluster assignment and cluster parameter of



16 Figure 33.5: Two views of  $N$  observations sampled from a DP mixture model. Left: representation where  
17 cluster indicators are sampled from the GEM-distributed distribution  $\pi$ . Right: representation where parameters  
18 are samples from the DP-distributed RPM  $G$ . The rightmost picture illustrates the case where  $N = 2$ ,  $\theta$  is  
19 real-valued with a Gaussian prior  $H(\cdot)$ , and  $F(x|\theta)$  is a Gaussian with mean  $\theta$  and variance  $\sigma^2$ . We generate  
20 two parameters,  $\bar{\theta}_1$  and  $\bar{\theta}_2$ , from  $G$ , one per data point. Finally, we generate two data points,  $x_1$  and  $x_2$ ,  
from  $\mathcal{N}(\bar{\theta}_1, \sigma^2)$  and  $\mathcal{N}(\bar{\theta}_2, \sigma^2)$ . From Figure 2.24 of [Sud06]. Used with kind permission of Erik Sudderth.

<sup>23</sup>the  $i$ th datapoint, whose value  $x_i$  is a draw from some parametric distribution  $F(x|\theta)$  indexed by  
<sup>24</sup> $\theta$ . The resulting model follows along the lines of a standard mixture model, but now is an **infinite**  
<sup>25</sup>**mixture model**, consisting of an infinite number of components or clusters, one for each atom in  $G$ .  
<sup>26</sup>We can write the model as follows:

$$\pi \circ GEM(\alpha) = \theta_1 \circ H \quad \quad k=1,2 \quad (33.16)$$

$$\frac{29}{\tilde{\gamma} \sim 2\pi} \quad x_i \sim E(\theta_{i-1}) \quad i = 1, 2, \dots, N \quad (33.17)$$

<sup>31</sup>Equivalently, we can write this as

$$\frac{32}{\pi^2} C = \text{DR}(\gamma, H) \quad (22.18)$$

$$\frac{\partial \phi}{\partial x} = \frac{1}{2} \left( \frac{\partial^2 \phi}{\partial x^2} \right)^{-1/2} \frac{\partial^2 \phi}{\partial x^2} \quad (33.13)$$

36  $G$  and  $F$  together define the infinite mixture model:  $G_F(x) \sim \sum_{k=1}^{\infty} \pi_k F(x|\theta_k)$ . If  $F(x|\theta)$  is  
37 continuous, then so is  $G_F(x)$ , and the Dirichlet process mixture model serves as a nonparametric prior  
38 over continuous distributions or probability densities. A very common setting when  $x_i \in \mathbb{R}^d$  is to set  
39  $F$  to be the multivariate normal distribution,  $\theta = (\mu, \Sigma)$ , and  $H$  to be the Normal-Inverse-Wishart  
40 distribution. Then, each of the infinite clusters has an associated mean and covariance matrix, and  
41 to generate a new observation, one picks cluster  $k$  with probability  $\pi_k$ , and simulates from a normal  
42 with mean  $\mu_k$  and covariance  $\Sigma_k$ . Figure 33.5 illustrates two graphical models that summarize this,  
43 corresponding to the two sets of equations above. The first generates the set of weights  $(\pi_1, \pi_2, \dots)$   
44 from the GEM distribution, along with an infinite collection of cluster parameters  $(\theta_1, \theta_2, \dots)$ . It  
45 then generates observations by first sampling a cluster indicator  $z_i$  from  $\pi$ , indexing the associated  
46 cluster parameter  $\theta_{z_i}$  and then simulating the observation  $x_i$  from  $F(\theta_{z_i})$ . The second graphical  
47

model simulates a random measure  $G$  from the DP. It generates observations by directly simulating a parameter  $\bar{\theta}_i$  from  $G$ , and then simulating  $x_i$  from  $F(\bar{\theta}_i)$ . The infinite mixture model can be viewed as the limit of a  $K$ -component finite mixture model with a  $\text{Dir}(\alpha/K, \dots, \alpha/K)$  prior on the mixture weights  $(\pi_1, \dots, \pi_K)$  and with mixture parameters  $\theta_1, \dots, \theta_K$ , as  $K \rightarrow \infty$  [Ras00; Nea00]. Producing exact samples  $(x_1, \dots, x_N)$  from this model involves one additional step to the Chinese restaurant process: after selecting a table (cluster)  $z_i$  with its associate dish (parameter)  $\theta_{z_i}$ , the  $i$ th customer now samples an observation from the distribution  $F(\theta_{z_i})$ .

### 33.2.4.1 Fitting a DP mixture model

Given a dataset of observations, one is interested in the posterior distribution  $p(G, z_1, \dots, z_N | x_1, \dots, x_N, \alpha, H)$ , or equivalently,  $p(\boldsymbol{\pi}, \theta_1, \theta_2, \dots, z_1, \dots, z_N | x_1, \dots, x_N, \alpha, H)$ . The most common way to fit a DPMM is via Markov chain Monte Carlo (MCMC), producing samples by constructing a Markov chain that targets this posterior distribution. We describe a **collapsed Gibbs sampler** based on the Chinese restaurant process that marginalizes out the infinite-dimensional random measure  $G$ , and that targets the distribution  $p(z_1, \dots, z_N | x_1, \dots, x_N, \alpha, H)$  summarizing all clustering information. It produces samples from this distribution by cycling through each observation  $x_i$ , and updating its assigned cluster  $z_i$ , conditioned on all other variables. Write  $\mathbf{x}_{-i}$  for all observations other than the  $i$ th observation, and  $\mathbf{z}_{-i}$  for their cluster assignments. Then we have

$$p(z_i = k | \mathbf{z}_{-i}, \mathbf{x}, \alpha, H) \propto p(z_i = k | \mathbf{z}_{-i}, \alpha) p(x_i | \mathbf{x}_{-i}, z_i = k, \mathbf{z}_{-i}, H) \quad (33.20)$$

By exchangeability, each observation can be treated as the last customer to enter the restaurant. Hence the first term is given by

$$p(z_i | \mathbf{z}_{-i}, \alpha) = \frac{1}{\alpha + N - 1} \left( \alpha \mathbb{I}(z_i = K_{-i} + 1) + \sum_{k=1}^{K_{-i}} N_{k,-i} \mathbb{I}(z_i = k) \right) \quad (33.21)$$

where  $N_{k,-i}$  is the number of observations in cluster  $k$ , and  $K_{-i}$  is the number of clusters used by  $\mathbf{x}_{-i}$ , both obtained after removing observation  $i$ , eliminating empty clusters, and renumbering clusters.

To compute the second term,  $p(x_i | \mathbf{x}_{-i}, z_i = k, \mathbf{z}_{-i}, H)$ , let us partition the data  $\mathbf{x}_{-i}$  into clusters based on  $\mathbf{z}_{-i}$ . Let  $\mathbf{x}_{-i,c} = \{x_j : z_j = c, j \neq i\}$  be the data points assigned to cluster  $c$ . If  $z_i = k$ , then  $x_i$  is conditionally independent of all the data points except those assigned to cluster  $k$ . Hence,

$$p(x_i | \mathbf{x}_{-i}, \mathbf{z}_{-i}, z_i = k) = p(x_i | \mathbf{x}_{-i,k}) = \frac{p(x_i, \mathbf{x}_{-i,k})}{p(\mathbf{x}_{-i,k})}, \quad (33.22)$$

where

$$p(x_i, \mathbf{x}_{-i,k}) = \int p(x_i | \theta_k) \left[ \prod_{j \neq i: z_j=k} p(x_j | \theta_k) \right] H(\theta_k) d\theta_k \quad (33.23)$$

is the marginal likelihood of all the data assigned to cluster  $k$ , including  $i$ , and  $p(\mathbf{x}_{-i,k})$  is an analogous expression excluding  $i$ . Thus we see that the term  $p(x_i | \mathbf{x}_{-i}, \mathbf{z}_{-i}, z_i = k)$  is the posterior predictive distribution for cluster  $k$  evaluated at  $x_i$ .

1 If  $z_i = k^*$ , corresponding to a new cluster, we have  
2

3  $p(x_i | \mathbf{x}_{-i}, \mathbf{z}_{-i}, z_i = k^*) = p(x_i) = \int p(x_i | \theta) H(\theta) d\theta$  (33.24)  
4

5 which is just the prior predictive distribution for a new cluster evaluated at  $x_i$ .

6 The overall sampler is sometimes called “Algorithm 3” (from [Nea00]). Algorithm 33 provides the  
7 pseudocode. The algorithm is very similar to collapsed Gibbs for finite mixtures except that we  
8 have to consider the case  $z_i = k^*$ . Note that in order to evaluate the integrals in Equation (33.23)  
9 and Equation (33.24), we require the base measure  $H$  to be conjugate to the likelihood  $F$ . In  
10 such conjugate settings, it is then also straightforward to sample cluster parameters given cluster  
11 assignments (if needed). Extensions to the case of non-conjugate priors are discussed in [Nea00].  
12

13

---

14 **Algorithm 33:** Collapsed Gibbs sampler for DPMM given data  $\mathcal{D} = \{x_1, \dots, x_N\}$

---

15 1 **for** each  $i = 1 : N$  in random order **do**  
16 2 Remove  $\mathbf{x}_i$ ’s sufficient statistics from old cluster  $z_i$  ;  
17 3 **for** each  $k = 1 : K$  **do**  
18 4    Compute  $p_k(\mathbf{x}_i) = p(\mathbf{x}_i | \mathbf{x}_{-i}(k))$ ;  
19 5    Set  $N_{k,-i} = \dim(\mathbf{x}_{-i}(k))$  ;  
20 6    Compute  $p(z_i = k | \mathbf{z}_{-i}, \mathcal{D}) = \frac{N_{k,-i}}{\alpha + N - 1}$ ;  
21 7    Compute  $p(z_i = * | \mathbf{z}_{-i}, \mathcal{D}) = \frac{\alpha}{\alpha + N - 1} p(\mathbf{x}_i)$ ;  
22 8    Normalize  $p(z_i | \cdot)$ ;  
23 9    Sample  $z_i \sim p(z_i | \cdot)$  ;  
24 10   Add  $\mathbf{x}_i$ ’s sufficient statistics to new cluster  $z_i$  ;  
25 11   If any cluster is empty, remove it and decrease  $K$ ;  
26

---

27

28 An example of this procedure in action is shown in Figure 33.6, where the sample clusterings,  
29 and the induced posterior over  $K$ , are reasonable looking. The MCMC algorithm tends to rapidly  
30 discover a good clustering. By contrast, Gibbs sampling (and EM) for a finite mixture model often  
31 gets stuck in poor local optima (not shown). This is because the DPMM is able to create extra  
32 redundant clusters early on, and to use them to escape local optima. The traceplots in Figure 33.7  
33 show that most of the time, the MCMC algorithm for the DPMM converges rapidly.  
34

35 An important issue is how to set the model hyper-parameters. These include the DP concentration  
36 parameter  $\alpha$ , as well as any parameters  $\lambda$  of the base measure  $H$ . For the DP, the value of  $\alpha$  does  
37 not have much impact on predictive accuracy, but has quite a strong affect the number of clusters.  
38 One approach is to put a Gamma prior for  $\alpha$ , and then form its posterior,  $p(\alpha | K, N, a, b)$  [EW95].  
39 Simulating  $\alpha$  given the cluster assignments  $\mathbf{z}$  is quite straightforward, and can be incorporated into  
40 the earlier Gibbs sampler. The same is the case with the hyper-parameters  $\lambda$  [Ras00]. Alternatively,  
41 one can use empirical Bayes [MBJ06] to fit rather than sample these parameters.

42 While Algorithm 33 is the simplest approach to posterior inference for DPMMs, a variety of  
43 other methods have been proposed as well. One popular class of MCMC samplers works with the  
44 stick-breaking representation of the DP instead of CRP, instantiating the random measure  $G$  [IJ01].  
45 The sampler then proceeds by sampling the cluster assignments  $\mathbf{z}$  given  $G$ , and then  $G$  given  $\mathbf{z}$ . An  
46 advantage of this is that the cluster assignments can be updated in parallel, unlike the CRP, where  
47

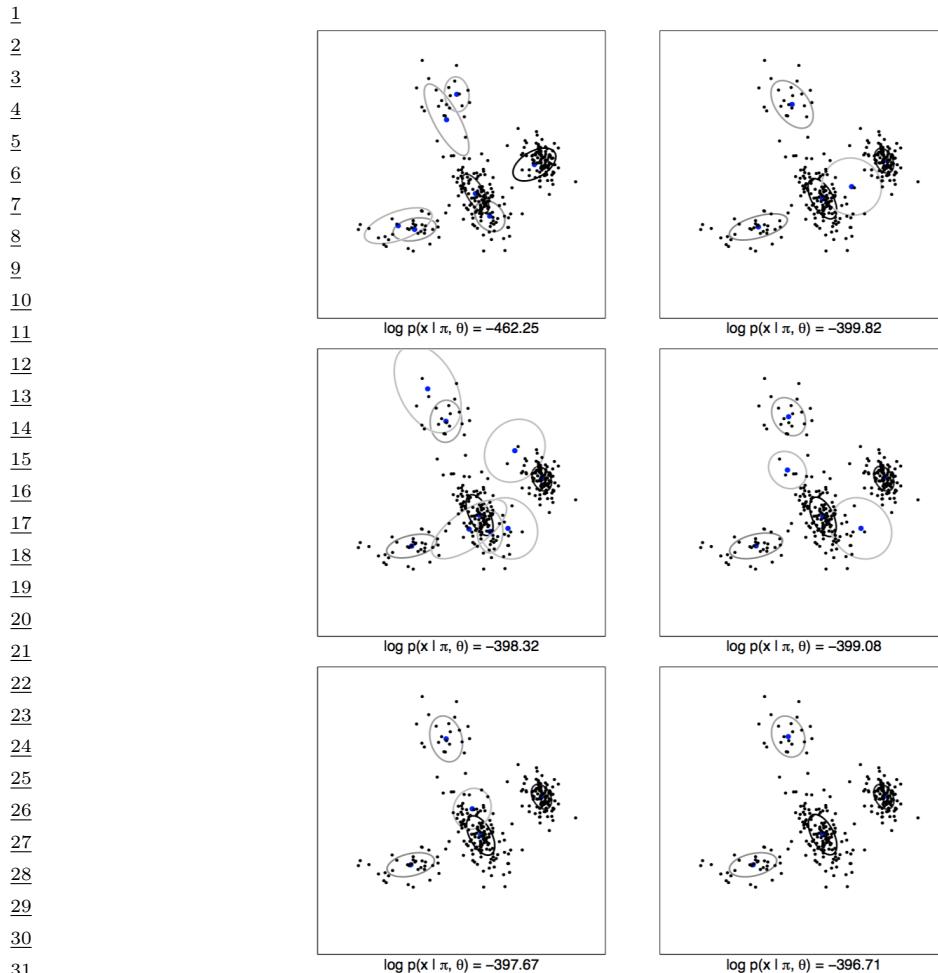


Figure 33.6: 300 data points in 2 dimensions are clustered using a DP mixture fit with collapsed Gibbs sampling. The two columns show different random initializations. The rows represent samples from the posterior over model parameters after  $T = 2$  (top),  $T = 10$  (middle) and  $T = 50$  (bottom) iterations. From Figure 2.26 of [Sud06]. Used with kind permission of Erik Sudderth.

they are updated sequentially. To be feasible however, these methods require truncating  $G$  to a finite number of atoms, though the resulting approximation error can be quite small. The posterior approximation error can be eliminated altogether by slice-sampling methods [KGW11], that work with random truncation levels. Alternatives to MCMC also exist. [Dau07] shows how one can use A\* search and beam search to quickly find an approximate MAP estimate. [Man+07] discusses how to fit a DPMM online using particle filtering, which is a like a stochastic version of beam search. This can be more efficient than Gibbs sampling, particularly for large datasets. A variety of variational approximation methods have been proposed as well, for example [BJ+06; KWW06; TKW08; Zob09;

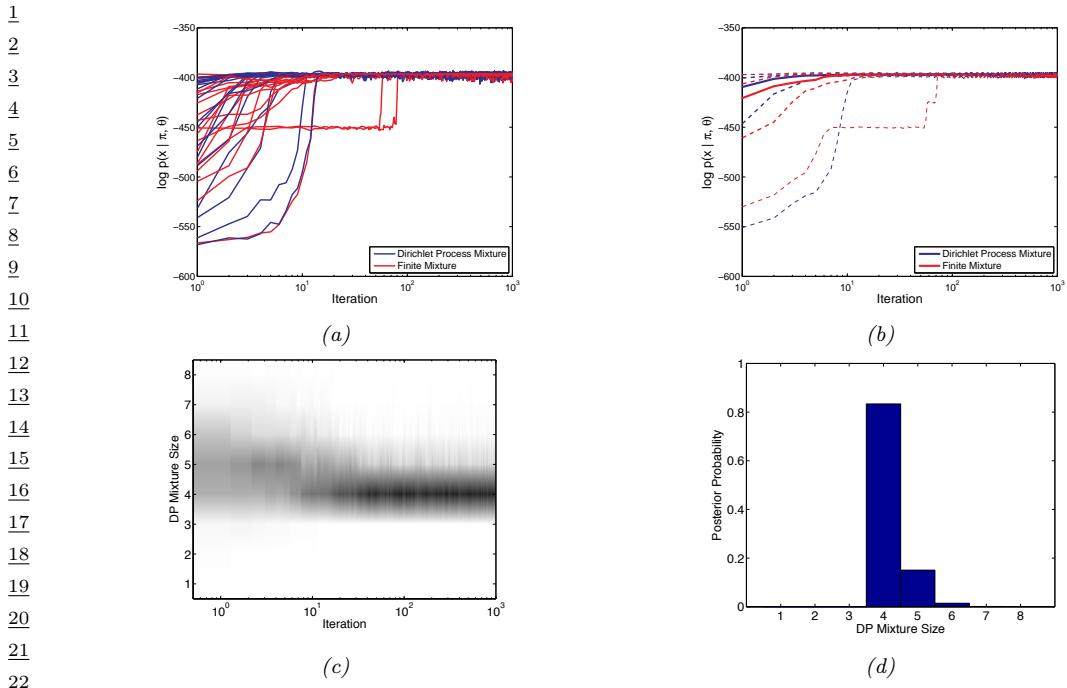


Figure 33.7: Comparison of collapsed Gibbs samplers for a DP mixture (dark blue) and a finite mixture (light red) with  $K = 4$  applied to the  $N = 300$  data points shown in Figure 33.6. (a) Log probability  $\log p(X|\pi, \theta)$  vs MCMC iteration for 20 different starting values. (b) Median (thick line) and quantiles (dashed lines) over 100 different starting values. (c) Number of components with at least 2% of the probability mass, vs iteration. (Intensity is proportional to posterior probability.) (d) Posterior over  $K$  based on last 900 samples. From Figure 2.27 of [Sud06]. Used with kind permission of Erik Sudderth.

WB12] among many others.

### 33.3 Generalizations of the Dirichlet process

Dirichlet process mixture models are flexible nonparametric models of continuous probability densities, and if set up with a little care, can possess important frequentist properties like **asymptotic consistency**: with more and more observations, the posterior distribution concentrates around the ‘true’ data generating distribution, with very little assumed about this distribution. Nevertheless, DPs still represent very strong prior information, especially in clustering applications. We saw that the number of clusters in a dataset of size  $N$  a priori is around  $\alpha \log N$ . As indicated by Equation (33.15), not just the number of clusters, but also the distribution of their sizes is controlled by a single parameter  $\alpha$ . The resulting clustering model is thus quite inflexible, and in many cases, inappropriate. Two examples from machine learning that highlight its limitations are applications involving text and image data. Here, it has been observed empirically that the number of unique words in a document, the frequency of word usage, the number of objects in an image, or the number of pixels in an object

follow power-law distributions. Clusterings sampled from the CRP do not produce this property, and the resulting model mismatch can result in poor predictive performance.

### 33.3.1 Pitman-Yor process

A popular generalization of the Dirichlet process is the Pitman-Yor process [PY97] (also called the two-parameter Poisson-Dirichlet process). Written at  $\text{PYP}(\alpha, d, H)$ , the Pitman-Yor process includes an additional **discount parameter**  $d$  which must be greater than 0. The concentration parameter can now be negative, but must satisfy  $\alpha \geq -d$ . As with the DP, a sample  $G$  from a PYP is a random probability measure that is discrete almost surely. It has a stick-breaking representation that generalizes that of the DP: again, we start with a stick of length 1, but now at step  $k$ , a random Beta( $1 - d, \alpha + kd$ ) fraction of the remaining probability mass is broken off, so that  $G$  is written as

$$G = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}, \quad (33.25)$$

$$\beta_k \sim \text{Beta}(1 - d, \alpha + kd), \quad \theta_k \sim H, \quad (33.26)$$

$$\pi_k = \beta_k \prod_{l=1}^{k-1} (1 - \beta_l) = \beta_k (1 - \sum_{l=1}^{k-1} \pi_l). \quad (33.27)$$

Because  $G$  is discrete, once again observations  $\bar{\theta}_1, \dots, \bar{\theta}_d$  sampled i.i.d. from  $G$  will possess a clustering structure. These can directly be sampled following a sequential process that generalizes the CRP. Now, when a new customer enters the restaurant, they join an existing table with  $N_k$  customers with probability proportional to  $(N_k - d)$ , and create a new table with probability proportional to  $\alpha + Kd$ , where  $K$  is the number of clusters.

Observe that the Dirichlet Process is a special instance of the Pitman-Yor process, corresponding to  $d$  equal to 0. Non-zero settings of  $d$  counteract the rich-get-richer dynamics of the DP to some extent, increasing the probability of creating new clusters. The more clusters there are present in a dataset, the higher the probability of creating a new cluster (relative to the Dirichlet process). This behavior means that a large number of clusters, as well as a few very large clusters are more likely under the PYP than the DP.

An even more general class of probability measures are the so-called Gibbs-type priors [DB+13]. Under such a prior, given  $N$  observations, the probability these are clustered into  $K$  clusters, the  $k$ th having  $n_k$  observations, is

$$p(n_1, \dots, n_k) = V_{N,K} \prod_{k=1}^K (\sigma - 1)_{N_k - 1}, \quad (33.28)$$

where  $(\sigma)_n = \sigma(\sigma + 1)\dots(\sigma + n - 1)$ , and for non-negative weights  $V_{N,K}$  satisfying  $V_{N,K} = (N - \sigma K)V_{N+1,K} + V_{N+1,K+1}$  and  $V_{1,1} = 1$ . This definition ensures that the probability over partitions is consistent, exchangeable and tractable, and includes the DP and PYP as special cases. Besides these two, Gibbs-type priors include settings where the number of components (or the number of atoms in the random measures) are random but bounded. Recall that DP and PYP mixture models are infinite mixture models, with the number of components growing with the number of observations. A sometimes undesirable feature of these models is that if a dataset is generated from

1 a finite number of clusters, these models will not this recover the true number of clusters [MH14].  
2 Instead, the estimated number of clusters will increases with the size of the dataset, resulting in  
3 redundant clusters that are located very close to each other. Gibbs-type priors with almost surely  
4 bounded number of components can learn the true number of clusters while still remaining reasonably  
5 tractable: so long as one can calculate the  $V_{N,K}$ s, MCMC for all these models can by carried out by  
6 modifications of the CRP-based sampler described earlier.  
7

8

### 9 33.3.2 Dependent random probability measures

10

11 Dirichlet processes, and more generally, random probability measures, have also been generalized  
12 from settings with a single set of observations to those involving grouped observations, or observations  
13 indexed by covariates. Consider  $T$  sets of observations  $\{\mathbf{X}^1, \dots, \mathbf{X}^T\}$ , each perhaps corresponding to  
14 a different country, a different year, or more abstractly, a different set of observed covariates. There  
15 are two immediate (though inadequate) ways to model such data: 1) by pooling all datasets into a  
16 single dataset, modeled as drawn from a DP or DPMM-distributed random probability measure  $G$ ,  
17 or 2) by treating each group as independent, having its own random probability measure  $G^t$ . The  
18 first approach fails to capture differences between the groups, while the second ignores similarities  
19 between the groups (e.g. shared clusters). Dependent random probability measures seek a compromise  
20 between these extremes, allowing statistical information to be shared across different groups.

21 A seminal paper in the machine learning literature that addresses this problem is the **hierarchical**  
22 **Dirichlet process** (HDP) [Teh+06]. The basic idea here is simple: each group has its own random  
23 measure drawn from a Dirichlet process  $DP(\alpha, H)$ . The twist now is that the base measure  $H$  itself  
24 is random, in fact it is itself drawn from a Dirichlet process. Thus, the overall generative process is

$$\begin{aligned} \text{25} \quad H &\sim DP(\alpha_0, H_0), \\ \text{26} \quad G^t &\sim DP(\alpha, H), \quad t \in 1, \dots, T \end{aligned} \tag{33.29}$$

$$\begin{aligned} \text{27} \quad \bar{\theta}_i^t &\sim G^t, \quad i \in 1, \dots, N_t, \quad t \in 1, \dots, T. \\ \text{28} \quad \theta_i^t &\sim F(\bar{\theta}_i^t), \quad i \in 1, \dots, N_t, \quad t \in 1, \dots, T. \end{aligned} \tag{33.30}$$

30 The  $\bar{\theta}_i^t$ s might be the observations themselves, or the latent parameter underlying each observation,  
31 with  $\mathbf{x}_i^t \sim F(\bar{\theta}_i^t)$ .

32 Recall that if a probability measure  $G^1$  is drawn from  $DP(\alpha, H)$ , its atoms are drawn independently  
33 from the base measure  $H$ . In the HDP,  $H$ , which is a draw from a DP, is discrete, so that some  
34 atoms of  $G^1$  will sit on top of each other, becoming a single atom. More importantly, all measures  
35  $G^t$  will share the same infinite set of locations: each atom of  $H$  will eventually be sampled to form a  
36 location of an atom of each  $G^t$ . This will allow the same clusters to appear in all groups, though they  
37 will have different weights. Moreover, a big cluster in one group is *a priori* likely to be a big cluster  
38 in another group, as underlying both is a large atom in  $H$ . Since the common probability measure  
39  $H$  itself is random, its components (both weights as well as locations) will be learned from data.  
40 Despite its apparent complexity, it is fairly easy to develop an analogue of the Chinese restaurant  
41 process for the HDP, allowing us to sample observations directly without having to instantiate any of  
42 the infinite-dimensional measures. This is called the Chinese restaurant franchise, and essentially  
43 amounts to each group having its own Chinese restaurant with the following modification: whenever  
44 a customer sits at a new table and orders a dish, that dish itself is sampled from an upper CRP  
45 common to all restaurants. It is also possible to develop stick-breaking representations of the HDP.  
46

47

The HDP has found wide application in the machine learning literature. A common application is document modeling, where the location of each atom is a **topic**, corresponding to some distribution over words. Rather than bounding the number of topics, there are an infinite number of topics, with document  $d$  having its own distribution over topics (represented by a measure  $G^d$ ). By tying all the  $G^d$ 's together through an HDP, different documents can share the same topics while emphasizing different topics. Another application involves infinite-state hidden Markov models. Recall a Markov chain is parametrized by a transition matrix, with row  $r$  giving the distribution over the next state if the current state is  $r$ . For an infinite-state HMM, this is an infinite-by-infinite matrix, with row  $r$  corresponding to a distribution  $G^r$  with an infinite number of atoms. The different  $G^r$ 's can be tied together by modeling them with an HDP, so that atoms from each correspond to the same states.

Hierarchical nonparametric models of this kind can be constructed with other measures such as the Pitman-Yor process. For certain parameter settings, the PYP possesses convenient marginalization properties that the DP does not. In particular, simulating an RPM in two steps as shown below

$$G_0|H_0 \sim \text{PYP}(\alpha d_0, d_0, H_0), \quad (33.32)$$

$$G_1|G_0 \sim \text{PYP}(\alpha, d_1, G_0) \quad (33.33)$$

is equivalent to directly simulating  $G_1$  without instantiating  $G_0$  as below:

$$G_1|H_0 \sim \text{PYP}(\alpha d_0, d_0 d_1, G_0). \quad (33.34)$$

This marginalization property (also called **coagulation**) allows deep hierarchies of dependent RPMs, with only a smaller, dataset-dependent subset of  $G$ 's having to be instantiated. In the **sequence memoizer** of [Woo+11], the authors model sequential data (e.g. text) with hierarchies that are *infinitely* deep, but with only a finite number of levels ever having to be instantiated. If needed, intermediate random measures can be instantiated by a dual **fragmentation** operator.

Deeper hierarchies like those described above allow more refined modeling of similarity between different groups. Under the original HDP, the groups themselves are exchangeable, with no subset of groups *a priori* more similar to each other than to others. For instance, suppose each of the measures  $G_1, \dots, G_T$  correspond to distributions over topics in different scientific journals. Modeling the  $G_t$ 's with an HDP allows statistical sharing across journals (through shared clusters and similar cluster probabilities), but does not regard some journals as *a priori* more similar than others. If one had further information, e.g. that some are physics journals and the rest are biology journals, then one might add another level to the hierarchy. Now rather than each journal having a probability measure  $G^t$  drawn from a DP( $\alpha, H$ ), physics and biology journals have their own base measures  $H_p$  and  $H_b$  that allow statistical sharing among physics and biology journals respectively. Like the HDP, these are draws from a DP with base measure  $H$ . To allow sharing *across* disciplines,  $H$  is again random, drawn from a DP with base measure  $H_0$ . Overall, if there are  $D$  disciplines,  $1, 2, \dots, D$ , the overall model is

$$H \sim \text{DP}(\alpha_0, H_0), \quad (33.35)$$

$$H^d \sim \text{DP}(\alpha_1, H), \quad d \in 1, \dots, D \quad (33.36)$$

$$G^{t,d} \sim \text{DP}(\alpha_2, H^d), \quad t \in 1, \dots, T_d \quad (33.37)$$

$$\theta_i^{t,d} \sim G^{t,d} \quad d \in 1, \dots, T_d, \quad t \in 1, \dots, N_t. \quad (33.38)$$

1 One might add further levels to the hierarchy given more information (e.g. if disciplines are  
2 grouped into physical sciences, social sciences and humanities).

4 Dependent random probability measures can also be indexed by covariates in some continuous  
5 space, whether time, space, or some Euclidean space or manifold. This space is typically endowed  
6 with some distance or similarity function, and one expects that measures with similar covariates  
7 are *a priori* more similar to each other. Thus,  $G^t$  might represent a distribution over topics in  
8 year  $t$ , and one might wish to model  $G^t$  evolving gradually over time. There is rich history of  
9 dependent random probability measures in statistics literature, starting from [Mac99a]. A common  
10 requirement in such models is that at any fixed time  $t$ , the marginal distribution of  $G^t$  follows some  
11 well specified distribution, e.g. a Dirichlet process, or a PYP. Approaches exploit the stick-breaking  
12 representation, the CRP representation or something else like the Poisson structure underlying such  
13 models (see Section 33.6).

14 As a simple and early example, recall the stick-breaking construction from Section 33.2.2, where a  
15 random probability measure is represented by an infinite collection of pairs  $(\beta_k, \theta_k)$ . To construct  
16 a family of RPMs indexed by some covariate  $t$ , we need a family of such sets  $(\beta_k^t, \theta_k^t)$  for each of  
17 the possibly infinite values of  $t$ . To ensure each  $G_t$  is marginally DP distributed with concentration  
18 parameter  $\alpha$  and base measure  $H$ , we need that for each  $t$ , the  $\beta_k^t$ 's are marginally i.i.d. draws from a  
19 Beta( $1, \alpha$ ), and the  $\theta_k^t$ 's i.i.d. draws from  $H$ . Further, we do not want independence across  $t$ , rather,  
20 for two times  $t_1$  and  $t_2$ , we want similarity to increase as  $|t_1 - t_2|$  decreases. To achieve this, define  
21 an infinite sequence of independent Gaussian processes on  $T$ ,  $f_k(t)$ ,  $k = 1, 2, \dots$ , with mean 0 and  
22 some covariance function. At any time  $t$ ,  $f_k(t)$  for all  $k$  are i.i.d. draws from a normal distribution.  
23 By transforming these Gaussian processes through the cdf of a Gaussian density (to produce a i.i.d.  
24 uniform random variables at each  $t$ ), and then through the inverse cdf of a Beta( $1, \alpha$ )), one has an  
25 infinite collection of i.i.d. Beta( $1, \alpha$ ) random variables at any time  $t$ . The GP construction means  
26 that each  $\beta_k^t$  varies smoothly with  $t$ . A similar approach can be used to construct smoothly varying  
27  $\theta_k^t$ 's that are marginally  $H$ -distributed, and together, these a family of gradually varying RPMs  $G^t$ ,  
28 with

$$\begin{aligned} G^t &= \sum_{i=1}^{\infty} \pi_k^t \delta_{\theta_k^t}, \quad \text{where } \pi_k^t = \beta_k^t \prod_{j=1}^{k-1} (1 - \beta_j^t) \end{aligned} \tag{33.39}$$

32  
33 Of course, such a model comes with formidable computational challenges, however the marginal  
34 properties allows standard MCMC methods from the DP and other RPMs to be adapted to such  
35 settings.

36

### 37 33.4 The Indian buffet process and the Beta process

38

39 The Dirichlet process is a Bayesian nonparametric model useful for clustering applications, where  
40 the number of clusters is allowed to grow with the size of the dataset. Under this generative model,  
41 each observation is assigned to a cluster through the variable  $z_i$ . Equivalently,  $z_i$  can be written as  
42 a one-hot binary vector, and the entire clustering structure can be written as a binary matrix  $Z$   
43 consisting of  $N$  rows, each with exactly one non-zero element (Figure 33.8(a)).

44 Clustering models that limit each observation to a single cluster can be overly restrictive, failing to  
45 capture the complexity of the real datasets. For instance, in computer vision applications, rather  
46 than assign an image to a single cluster, it might be more appropriate to assign it a binary vector of  
47

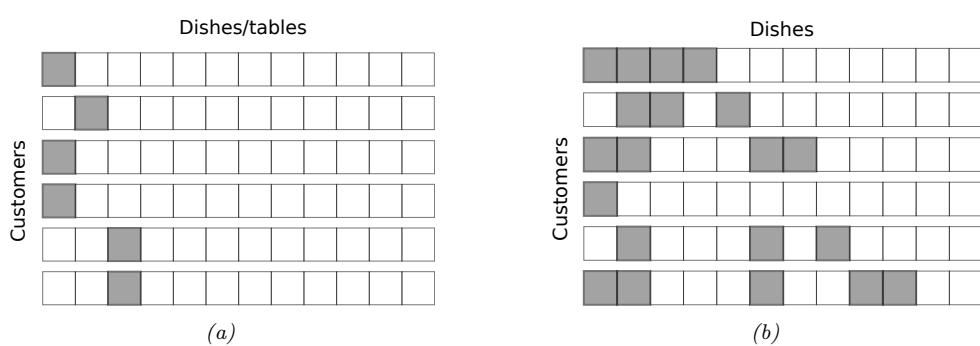


Figure 33.8: (a) A realization of the cluster matrix  $Z$  from the Chinese restaurant process (CRP) (b) A realization of the feature matrix  $Z$  from the Indian buffet process (IBP). Rows are customers and columns are dishes (for the CRP each table has its own dish). Both produce binary matrices, with the CRP assigning a customer to a single table (cluster), and the IBP assigning a set of dishes (features) to each customer.

features, indicating whether different object types are or are not present in the image. Similarly, in movie recommendation systems, rather than assign a movie to a single genre ('comedy' or 'romance' etc.), it is more realistic to assign to multiple genres ('comedy' AND 'romance'). Now, in contrast to all-or-nothing clustering (which would require a new genre 'romantic comedy'), different movies can have different but overlapping sets of features, allowing a partial sharing of statistical information.

Latent feature models generalize clustering models, allowing each observation to have multiple features. Nonparametric latent feature models allow the number of available features to be infinite (rather than fixed *a priori* to some finite number), with the number of active features in a dataset growing with a dataset size. Such models associate the dataset with an infinite binary matrix consisting of  $N$  rows, but now where each row can have multiple elements set to 1, corresponding to the active features. This is shown in Figure 33.8(b). As with the clustering models, each column is associated with a parameter drawn i.i.d. from some base measure  $H$ .

The Indian buffet process (IBP) is a Bayesian nonparametric analogue of the CRP for latent feature models. As with the CRP, the IBP is specified by a concentration parameter  $\alpha$ , and a base measure  $H$  on some space  $\Theta$ . The former controls the distribution over the binary feature matrix, whereas feature parameters  $\theta_k$  are drawn i.i.d. from the latter. Under the IBP, individuals enter sequentially into a restaurant, now picking a set of dishes (instead of a single table). The first customer samples a Poisson( $\alpha$ )-distributed random number of dishes, and assigns each of them values drawn from  $H$ . When the  $i$ th customer enters the restaurant, they first make a pass through all dishes already chosen. Suppose  $N_d$  of the earlier customers have chosen dish  $d$ : then customer  $i$  selects this with probability  $N_d/i$ . This results in a rich-get-richer phenomenon, where popular dishes (common features) are more likely to be selected in the future. Additionally, the  $i$ th customer samples a Poisson( $\alpha/i$ ) number of new dishes. The results in a non-zero probability of new dishes, that nonetheless decreases with  $i$ .

A key property of the Indian buffet process is that, like the CRP, it is exchangeable. In other words, its statistical properties do not change if its rows are permuted. For instance, one can show that the number of dishes picked by any customer is marginally Poisson( $\alpha$ ) distributed. Similarly, the distribution over the number of features shared by the first two customers is the same as that for

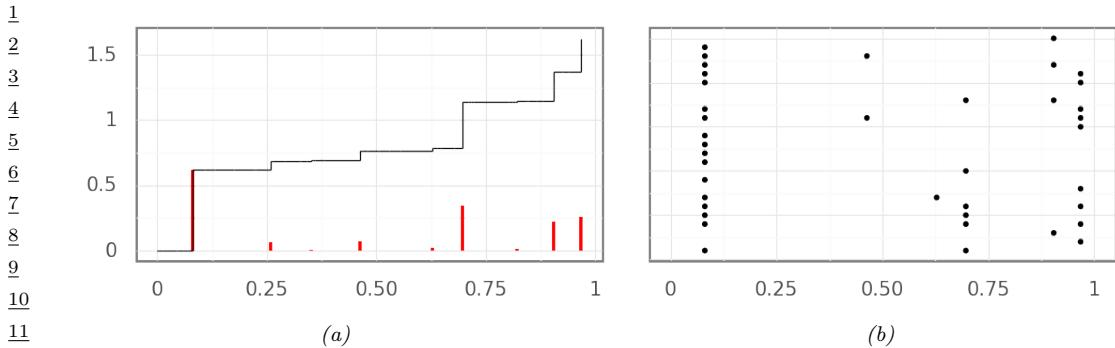


Figure 33.9: (a) A realization of a Beta process on the real line. The atoms of the random measure are shown in red, while the Beta subordinator (whose value at time  $t$  sums all atoms up to  $t$ ) is shown in black. (b) Samples from the Beta process

the first and last customer. We mention that like the CRP, the ordering of the dishes (or columns) is arbitrary and might appear to violate exchangeability. For instance, the first customer cannot pick the first and third dishes and not the second dish, while this is possible for the second customer. These artefacts disappear if we index columns by their associated parameters. Equivalently, after reordering rows, we can transform the feature matrix to be left-ordered (essentially, all new dishes selected by a customer must be adjacent, see [GG11]), and we can view the IBP as a prior on such left-ordered matrices.

The exchangeability of the rows of the IBP implies via de Finetti's theorem that there exists an underlying random measure  $G$ , conditioned on which the rows are i.i.d. draws. Just as the Chinese restaurant process represents observations drawn from a Dirichlet process-distributed random probability measure, the dishes of each customer in the IBP represent observations drawn from a **Beta process**. Like the DP, the Beta process is an atomic measure taking the form

$$G(\boldsymbol{\theta}) = \sum_{k=1}^{\infty} \pi_k \delta_{\boldsymbol{\theta}_k}(\boldsymbol{\theta}). \quad (33.40)$$

Each of the  $\pi_k$ 's lie between 0 and 1, but unlike the DP where they add up to 1, the  $\pi_k$ 's sum up to a finite random number. The Beta process is thus a **random measure**, rather than a random probability measure. One can imagine the  $k$ th atom as a coin located at  $\boldsymbol{\theta}_k$ , with probability of success equal to  $\pi_k$ . To simulate a row of the IBP, one flips each of the infinite coins, selecting the feature at  $\boldsymbol{\theta}_k$  if the  $k$  coin comes up heads. One can show that if the  $\pi_k$ 's sum up to a finite number, the number of active features will be finite. Of the infinite atoms in  $G$ , a few will dominate the rest, these will be revealed through the rich-gets-richer dynamics of the IBP as features common to a large proportion of the observations.

The Beta process has a construction similar to the stick-breaking representation of the Dirichlet process. As with the DP, the locations of the atoms are independent draws from the base measure, while the sequence of weights  $\pi_1, \pi_2, \dots$  are constructed from an infinite sequence of Beta variables: now these are  $\text{Beta}(\alpha, 1)$  distributed, rather than  $\text{Beta}(1, \alpha)$  distributed. The overall representation

of the Beta process is then

$$\beta_k \sim \text{Beta}(\alpha, 1), \quad \theta_k \sim H, \quad (33.41)$$

$$\pi_k = \beta_k \pi_{k1} = \prod_{l=1}^k \beta_l \quad (33.42)$$

It is not hard to see that under the IBP, the total number of dishes underlying a dataset of size  $N$  follows a  $\text{Poisson}(\alpha H_N)$  distribution, where  $H_N = \sum_{i=1}^N 1/i$  is the  $N$ th harmonic number. The Beta process and the IBP have been generalized to three-parameter versions allowing power-law behavior in the total number of dishes (features) in a dataset of size  $N$ , as well as in the number of customers trying each dish [TG09]. It has found application in tasks ranging from genetics, collaborative filtering, and in models for graph and graphical model structures. Just as with the DP, posterior inference can proceed via MCMC (exploiting either the IBP or the stick-breaking representation), particle filtering or using variational methods.

### 33.5 Small-variance asymptotics

Nonparametric Bayesian methods can serve as a basis to develop new and efficient discrete optimization algorithms that have the flavor of Lloyd's algorithm for the  $k$ -means objective function. The starting point for this line of work is the view of the  $k$ -means algorithm as the *small-variance asymptotic limit* of the EM algorithm for a mixture of Gaussians. Specifically, consider the EM algorithm to estimate the unknown cluster means  $\mu \equiv (\mu_1, \dots, \mu_k)$  of a mixture of  $k$  Gaussians, all of which have the same known covariance  $\sigma^2 I$ . In the limit as  $\sigma^2 \rightarrow 0$ , the E-step, which computes the cluster assignment probabilities of each observation given the current parameters  $\mu^{(t)}$ , now just assigns each observation to the nearest cluster. The M-step recomputes a new set of parameters  $\mu^{(t+1)}$  given the cluster assignment probabilities, and when each observation is hard-assigned to a single cluster, the cluster means are just the means of the assigned observations. The process of repeatedly assigning observations to the nearest cluster, and recomputing cluster locations by averaging the assigned observations is exactly Lloyd's algorithm for  $k$ -means clustering.

To avoid having to specify the number of clusters  $k$ , [KJ12b] considered a Dirichlet process mixture of Gaussians, with all infinite components again having the same known variance  $\sigma^2$ . The base measure  $H$  from which the component means are drawn was set to a zero-mean Gaussian with variance  $\rho^2$ , with  $\alpha$  the concentration parameter. The authors then considered a Gibbs sampler for this model, very closely related to the sampler in Algorithm 33, except that instead of collapsing or marginalizing out the cluster parameters, these are instantiated. Thus, an observation  $x_i$  as assigned to a cluster  $c$  with mean  $\mu_c$  with probability proportional to  $N_{c,-i} \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2\sigma^2} \|x_i - \mu_c\|^2)$ , while it is assigned to a new cluster with probability proportional to  $\alpha \frac{1}{\sqrt{2\pi(\sigma^2 + \rho^2)}} \exp(-\frac{1}{2(\sigma^2 + \rho^2)} \|x_i\|^2)$ . After cycling through all observations, one then resamples the parameters of each cluster. Following the Gaussian base measure and Gaussian likelihood, this too follows a Gaussian distribution, whose mean is a convex combination of the prior mean 0 and data-driven term, specifically the average of the assigned observations. The weight of the prior term is proportional to the inverse of the prior variance  $\rho^2$ , while the weight of the likelihood term is proportional to the inverse of the likelihood variance  $\sigma^2$ .

1 To derive a small-variance limit of the sampler above, we cannot just let  $\sigma^2$  go to 0, as that  
2 would result in each observation being assigned to its own cluster. To prevent the likelihood from  
3 dominating the prior in this way, one must also send the concentration parameter  $\alpha$  to 0, so that the  
4 DP prior enforces an increasingly strong penalty on a large number of clusters (recall that *a priori*,  
5 the average number of clusters underlying  $N$  observations is  $\alpha \log N$ ). [KJ12b] showed that with  $\alpha$   
6 scaled as  $\alpha = (1 + \rho^2/\sigma^2)^{d/2} \exp(-\frac{\lambda}{2\sigma^2})$  for some parameter  $\lambda$ , and taking the limit  $\sigma^2 \rightarrow 0$  with  $\rho$   
7 fixed, we get the following modification of the  $k$ -means algorithm:  
8

- 9 1. Assign each observation to the nearest cluster, *unless the distance to the nearest cluster exceeds  $\lambda$ ,*  
10    *in which case assign the observation to its own cluster.*  
11  
12 2. Set the cluster means equal to the average of the assigned observations.  
13  
14

---

15 **Algorithm 34:** DP-means hard clustering algorithm for data  $\mathcal{D} = \{x_1, \dots, x_N\}$

---

16 1 Initialize the number of clusters  $K = 1$ , with cluster parameter  $\mu_1$  equal to the global mean:

17     $\mu_1 = \frac{1}{N} \sum_{i=1}^N x_i;$

18 2 Initialize all cluster assignment indicators  $z_i = 1$ ;

19 3 **while** all  $z_i$ s have not converged **do**

20    4    **for** each  $i = 1 : N$  **do**

21       5    Compute distance  $d_{ik} = \|x_i - \mu_k\|^2$  to cluster  $k$  for  $k = 1, \dots, K$ ;

22       6    **if**  $\min_k d_{ik} > \lambda$  **then**

23          7    Increase the number of clusters  $K$  by 1:  $K = K + 1$  ;

24          8    Assign observation  $i$  to this cluster:  $z_i = K$  and  $\mu_K = x_i$ ;

25       9    **else**

26       10       Set  $z_i = \arg \min_c d_{ik}$  ;

27       11    **for** each  $k = 1 : K$  **do**

28          12    Set  $\mathcal{D}_k$  be the observations assigned to cluster  $k$ . Compute cluster mean

29          13        $\mu_k = \frac{1}{|\mathcal{D}_k|} \sum_{x \in \mathcal{D}_k} x;$

---

30

31

32

33

34 Like  $k$ -means, this is a hard-clustering algorithm, except that instead of having to specify the  
35 number of clusters  $k$ , one just specifies a penalty  $\lambda$  for introducing a new cluster. The actual number  
36 of clusters is determined by the data, [KJ12b] refer to this algorithm as DP-means. One can show that  
37 the iterates above converge monotonically to a local maximum of the following objective function:  
38

$$\sum_{k=1}^K \sum_{x \in \mathcal{D}_k} \|x - \mu_k\|^2 + \lambda K, \quad \text{where } \mu_k = \frac{1}{|\mathcal{D}_k|} \sum_{x \in \mathcal{D}_k} x. \quad (33.43)$$

39 The first term in the expression above is exactly the objective function of the  $k$ -means algorithm with  
40  $K$  clusters. The second term is a penalty term that introduces a cost  $\lambda$  for each additional cluster.  
41 Interestingly, the penalty term above corresponds to the so called Akaike Information Criterion  
42 (AIC), a well studied approach to penalizing model complexity.

43

It is also possible to derive hard-clustering algorithms for simultaneously clustering multiple datasets, while allowing these to share clusters. This is possible through the small-variance limit of a Gibbs sampler for the hierarchical Dirichlet process (HDP) and results in a clustering algorithm that now has two thresholding parameters, a local one  $\lambda_l$  and a global one  $\lambda_g$ . The algorithm proceeds by maintaining a set of global clusters, with the local clusters of each dataset assigned to a subset of the global clusters. It then repeats the following steps until convergence:

**Assign observations to local clusters** For  $x_{ij}$ , the  $i$ th datapoint in dataset  $j$ , compute the distance to all global clusters. For those global clusters not currently present in dataset  $j$ , add  $\lambda_l$  to their distance; this reflects the cost of introducing a new cluster into dataset  $j$ . Now, assign  $x_{ij}$  to the cluster with the smallest distance, unless the smallest distance exceeds  $\lambda_g + \lambda_l$ , in which case, create a new global cluster and assign  $x_{ij}$  to it. Observe that in the latter case, the distance of  $x_{ij}$  to the new cluster is 0, with  $\lambda_g + \lambda_l$  reflecting the cost of introducing a new global and then local cluster.

**Assign local clusters to global clusters** For each local cluster  $l$ , compute the sum of the distances of all its assigned observations to the cluster mean. Call this  $d_l$ . Also compute the sum of the distances of the assigned observations to each global cluster. For global cluster  $p$ , call this  $d_{l,p}$ . Then assign local cluster  $l$  to the global cluster with the smallest  $d_{l,p}$ , unless  $\min d_{l,p} > d_l + \lambda_g$  in which case we create a new global cluster.

**Recompute global cluster means** Set the global cluster means equal to the average of the assigned observations across all datasets.

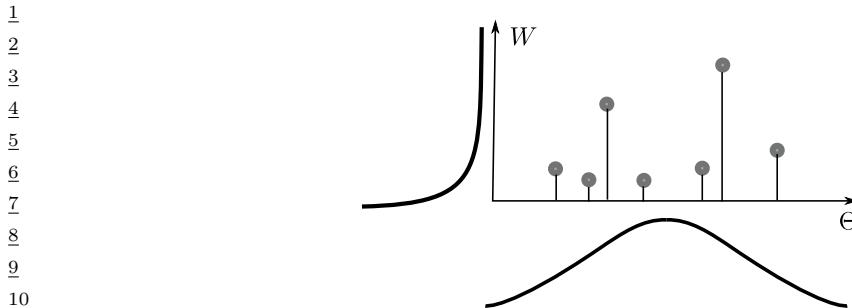
In [JKJ12], these ideas were extended from DP mixtures of Gaussians to DP mixtures of more general exponential family distributions. Briefly, the hard-clustering algorithms maintained the same structure, the only difference being that distance from clusters was measured using a **Bregman divergence** specific to that exponential family distribution. For Gaussians, the Bregman divergence reduces to the usual Euclidean distance.

In [Bkj13], the authors showed how such small-variance algorithms could be derived directly from a probabilistic model, and independent of any specific computational algorithm such as EM or Gibbs sampling. Their approach involves computing the MAP solution of the parameters of the model, and then taking the small-variance limit to obtain an objective function. This approach, called MAP-based Asymptotic Derivations from Bayes or MAD-Bayes allowed them to derive, among other things, an analog of the DP-means algorithm from feature-based models. They called this the BP-means algorithm, after the Beta process underlying the Indian Buffet process.

Write  $X$  for an  $N \times D$  matrix of  $N$   $D$ -dimensional observations,  $Z$  for an  $N \times K$  matrix of binary feature assignments, and  $A$  for a  $K \times D$  matrix of  $K$   $D$ -dimensional features, with one seeking a pair  $(A, Z)$  such that  $ZA$  approximates  $X$  as well as possible. [Bkj13] showed in the small-variance limit, finding the MAP solution for the IBP is equivalent to the following problem:

$$\operatorname{argmin}_{K,Z,A} \text{trace}[(X - ZA)^t(X - ZA)] + \lambda K, \quad (33.44)$$

where again  $\lambda$  is a parameter of the algorithm, governing how the concentration parameter scales with the variance, and specifying a penalty on introducing new features. This objective function is very intuitive: the first term corresponds to the approximation error from  $K$  features, while the second term penalizes model complexity resulting from a large number of features  $K$ . This objective function can be optimized greedily by repeating three steps for each observation  $i$  until convergence:



11 *Figure 33.10: Poisson process construction of a completely random measure  $G = \sum_i w_i \delta_{\theta_i}$ . The set of pairs*

12  $\{(w_1, \theta_1), (w_2, \theta_2), \dots\}$  *is a realization from a Poisson process with intensity  $\lambda(\theta, w) = H(\theta)\gamma(w)$ .*

13

14

15 1. Given  $A$  and  $K$ , compute the optimal value of the binary feature assignment vector  $z_i$  of observation

16  $i$  and update  $Z$ .

17

18 2. Given  $Z$  and  $A$ , introduce an additional feature vector equal to the residual for the  $i$ th observation,

19 namely,  $x_i - z_i A$ . Call  $A'$  the updated feature matrix, and  $Z'$  the updated feature assignment

20 matrix, where only observation  $i$  has been assigned this feature. If the configuration  $(K+1, Z', A')$

21 results in a lower value of the objective function than  $(K, Z, A)$ , set the former as the new

22 configuration. In words, if the benefit of introducing a new feature outweighs the penalty  $\lambda$ , then

23 do so.

24

25 3. Update the feature vectors  $A$  given the feature assignment vectors  $Z$ .

26 [BKT13] showed that this algorithm monotonically decreases the objective in Equation (33.44),

27 converging eventually to a local optima. Subsequent works have considered the small-variance

28 asymptotics of more structured models, such as topic models, hidden Markov models and even

29 continuous-time stochastic process models.

30

### 31 33.6 Completely random measures

32 The Dirichlet process is a example of a random probability measure, a class of random measures

33 which always integrate to 1. The Beta process, while not an RPM, belongs to another class of random

34 measures: **completely random measures** (CRM). A completely random measure  $G$  satisfies the

35 following property: for any two disjoint subsets  $T_1$  and  $T_2$  of  $\Theta$ , the values  $G(T_1)$  and  $G(T_2)$  are

36 independent random variables:

37

$$\text{38} \quad G(T_1) \perp\!\!\!\perp G(T_2) \quad \forall T_1, T_2 \subset \Theta \text{ s.t. } T_1 \cap T_2 = \emptyset. \quad (33.45)$$

40

41 Note that the Dirichlet process is not a CRM, since for disjoint sets  $T$  and its complement  $T^c = \Theta \setminus T$ ,

42 we have  $G(T) = 1 - G(T^c)$ , which is as far from independent as can be. More generally, under the

43 DP, for disjoint sets  $T_1$  and  $T_2$ , the measures  $G(T_1)$  and  $G(T_2)$  are negatively correlated. We will see

44 later though that the Dirichlet process is closely related to another CRM, the Gamma process. As a

45 side point, beyond the sum-to-one constraint,  $G(T_1)$  does not tell us anything about the distribution

46 of probability within  $G(T_1^c)$ , making the DP what is known as a **neutral process**.

47

The simplest example of a CRM is the **Poisson process** (see Section 33.8). A Poisson process with intensity  $\lambda(\theta)$  is a **point process** producing points or events in  $\Theta$ , with the number of points in any set  $T$  following a  $\text{Poisson}(\int_T \lambda(\theta)d\theta)$ -distribution, and with the counts in any two disjoint sets independent of each other. While it is common to think of this as a point process, one can also think of a realization from a Poisson process as an **integer-valued random measure**, where the measure of any set is the number of points falling within that set. It is clear then that the Poisson process is an example of a CRM.

It turns out that the Poisson process underlies all completely random measures in a fundamental way. For some space  $\Theta$ , and  $\mathbb{W}$  the positive real line, simulate a Poisson process on the product space  $W \times \Theta$  with intensity  $\lambda(\theta, w)$ . Figure 33.10 shows a realization from this Poisson process, write it as  $M = \{(\theta_1, w_1), \dots, (\theta_{|M|}, w_{|M|})\}$  where  $|M|$  is the number of events. This can be used to construct an atomic measure  $G = \sum_{i=1}^{|M|} w_i \delta_{\theta_i}$  as illustrated in Figure 33.10. From its Poisson construction, this is a completely random measure on  $\Theta$ , with set  $T \in \Theta$  having measure  $\sum_i w_i \mathbb{I}(\theta_i \in T)$ . Different settings of  $\lambda(\theta, w)$  give rise to different CRMs, and in fact, other than CRMs with atoms at some fixed locations in  $\Theta$ , this construction characterizes *all* CRMs.

For a CRM, the Poisson intensity is typically chosen to factor as  $\lambda(\theta, w) = H(\theta)\gamma(w)$ , with  $\int_\Theta H(\theta)d\theta = 1$ . Then,  $H(\theta)$  is the base measure controlling the locations of the atoms in the CRM, while the measure  $\gamma(w)$  controls the number of atoms, and the distribution of their weights. Setting  $\gamma(w) = w^{-1}(1-w)^{\alpha-1}$  gives the Beta process with base measure  $H(\theta)$ . Other choices include the Gamma process ( $\gamma(w) = \alpha w^{-1} \exp(-w)$ ), the stable process ( $\gamma(w) = \frac{1}{\Gamma(1+\sigma)} \alpha w^{-1-\sigma}$ ), and the generalized Gamma process ( $\gamma(w) = \frac{1}{\Gamma(1+\sigma)} \alpha w^{-1-\sigma} \exp(-\zeta w)$ ).

For all three processes described earlier, the  $\gamma(w)$  integrates to infinity, so that  $\int_\Theta \int_W \lambda(\theta, w)d\theta dw = \infty$ . Consequently, the number of Poisson events, and thus the number of atoms in the CRMs are infinite with probability one. At the same time, mass of the  $\gamma(w)$  function is mostly concentrated around 0 (see Figure 33.10), and for any  $\epsilon > 0$ ,  $\int_\epsilon^\infty \gamma(w)dw$  is finite. It is easy to show that the sum of the  $w$ 's is finite almost surely. Call this sum  $W$ , then the first condition ensures  $W$  greater than 0, while the second ensures it is finite. These two conditions make it sensible to divide a realization of a CRM by its sum, resulting in a random measure that integrates to 1: a random probability measure. Such RPs are called **normalized completely random measures**, or sometimes just **normalized random measures (NRMs)**. The Dirichlet process we saw earlier is an example of an NRM: it is a normalized Gamma process. This result mirrors the situation with the finite Dirichlet distribution: one can simulate from a  $d$ -dimensional  $\text{Dir}(\alpha_1, \dots, \alpha_d)$  distribution by first simulating  $d$  independent Gamma variables  $g_i \sim \text{Ga}(\alpha_i, 1), i = 1, \dots, d$ , and then defining the probability vector  $\frac{1}{\sum_{i=1}^d g_i}(g_1, \dots, g_d)$ . The Pitman-Yor process is not an NRM except for special settings of its parameters: like we saw,  $d = 0$  is a Dirichlet process or normalized Gamma process.  $\alpha = 0$  is a **normalized stable process**. The normalized generalized Gamma process is an NRM that includes the DP and the normalized stable process as special cases.

## 33.7 Lévy processes

Completely random measures are also closely related to **Lévy processes** and **Lévy subordinators** [Ber96]. A Lévy process is a continuous-time stochastic process  $\{L_t\}_{t \geq 0}$  taking values in some

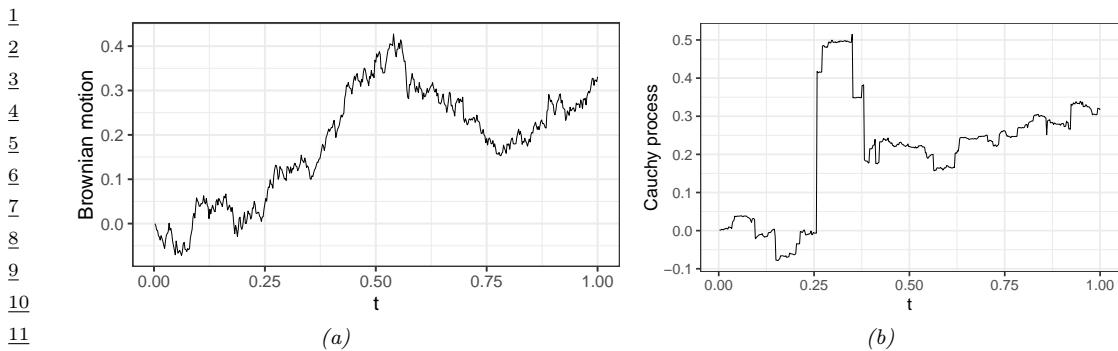


Figure 33.11: (a) A realization of a Brownian motion (b) A realization of Cauchy process (an  $\alpha$ -stable process with  $\alpha = 1$ ).

space (e.g.  $\mathbb{R}^d$ ) that satisfies two properties<sup>1</sup>:

**stationary increments:** for  $t, \Delta > 0$ , the random variable  $L_{t+\Delta} - L_t$  does not depend on  $t$

**independent increments:** for  $t, \Delta > 0$ ,  $L_{t+\Delta} - L_t$  is independent of values before  $t$ .

A Lévy subordinator is a real-valued, **nondecreasing** Lévy process. If we drop the stationarity condition, it should be clear that the increments of a Lévy subordinator are exactly the atoms of a completely random measure (see Figure 33.9). Common examples of Lévy subordinators are Beta subordinators (or Beta processes), Gamma subordinators, stable subordinators and generalized Gamma subordinators. CRMs generalize Lévy subordinators, by allowing them to be indexed by some general space  $\Theta$  (rather than by nonnegative reals), and by relaxing the stationarity condition to allow the atoms to follow some general base measure  $H$ .

Unlike Lévy subordinators, a general Lévy process can have negative changes as well, with the change  $L_{t+\Delta} - L_t$  belonging to an **infinitely divisible distribution**, scaled by  $\Delta$ . A random variable  $X$  follows an infinitely divisible distribution if, for any positive integer  $N$ , there exists some probability distribution such that the sum of  $N$  i.i.d. samples from that distribution has the same distribution as  $X$ . Examples of infinitely divisible distributions include the Poisson distribution, the Gamma distribution, the Cauchy distribution, the  $\alpha$ -stable distribution, and the inverse-Gaussian distribution (among many others), though by far the most well-known example is the Gaussian distribution. It is easy to see why the change in a Lévy process  $L_{t+\Delta} - L_t$  over some time interval  $\Delta$  follows an infinitely divisible distribution: just divide the interval into  $N$  equal-length segments. From the properties of the Lévy process, the changes over these segments are independent and identically distributed, and their sum equals  $L_{t+\Delta} - L_t$ . The **Lévy-Kintchine** formula shows that the converse is also true: any infinitely divisible distribution represents the change of an associated Lévy process. The Lévy process corresponding to the Gaussian distribution is the celebrated **Brownian motion** (or **Weiner process**). Brownian motion is a fundamental and widely applied stochastic process, whose mathematics was first studied by Louis Bachelier to model stock markets, and later, famously, by Albert Einstein to argue about the existence of atoms. For a Brownian motion, the

<sup>1</sup> There is also a technical continuity condition that we do not discuss here

increment  $L_{t_1+\Delta} - L_{t_1}$  follows a normal  $N(\mu\Delta, \sigma\Delta)$  distribution, where  $\mu$  and  $\sigma$  are the *drift* and *diffusion* coefficients. Setting  $\mu$  and  $\sigma$  to 0 and 1 gives **standard Brownian motion**. Paths sampled from the Weiner process are continuous with probability one, all other Lévy processes are jump processes. Figure 33.11 shows realizations from some processes. The jump processes have a Poisson construction related to the Lévy subordinators and completely random measures, and the **Lévy-Itô decomposition** shows that every Lévy process can be decomposed into Brownian and Poisson components. Levy processes have been widely applied in mathematical finance to model asset prices, insurance claims, stock prices and other financial assets.

## 33.8 Point processes with repulsion and reinforcement

In this section, we look more closely at the Poisson process, as well as other, more general point processes that allow inter-event interactions.

### 33.8.1 Poisson process

We have already briefly seen the Poisson process: this is a point process on some space  $\Theta$  that is parametrized by an intensity function  $\lambda(\theta) \geq 0$ , and that produces a  $\text{Poisson}(\int_T \lambda(\theta)d\theta)$ -distributed number of points of events in any set  $T \in \Theta$ , with the counts in any two disjoint sets independent of each other. Recall that if  $N_i \sim \text{Poisson}(\lambda_i)$  are independent, then a well-known property of the Poisson distribution is that  $N_1 + N_2 \sim \text{Poisson}(\lambda_1 + \lambda_2)$ . This relates to the infinite divisibility of the Poisson distribution. It is clear that the average number of points is large in areas where  $\lambda(\theta)$  is high, and small where  $\lambda(\theta)$  is small. When the intensity  $\lambda(\theta)$  is some constant  $\lambda$ , we have a **homogeneous Poisson process**, otherwise we have an **inhomogeneous Poisson process**.

Depending on whether  $\int_\Theta \lambda(\theta)d\theta$  is infinite or finite, a Poisson process will either produce an infinite number of points (recall the Poisson process underlying the CRMs of Section 33.6) or a finite number of points. The latter is more common in applications, such as modeling phone calls or financial shocks ( $\Theta$  is some finite time-interval), the locations of trees or forest fires ( $\Theta$  is some subset of the Euclidean plane), the locations of cells or galaxies ( $\Theta$  is a 3-dimensional space) or events in higher-dimensional spaces (for example, spatio-temporal activity). One way to simulate a finite Poisson process is to first simulate the number of total number of points  $N$ , which follows a  $\text{Poisson}(\int_\Theta \lambda(\theta)d\theta)$  distribution. One can then simulate the locations of these points by sampling  $N$  times from the probability density  $\frac{\lambda(\theta)}{\int_\Theta \lambda(\theta)d\theta}$ . For a homogeneous Poisson process, the locations are uniformly distributed over  $\Theta$ . One can also easily simulate a rate- $\lambda$  homogeneous Poisson on the real line by exploiting the fact that inter-event times follow an exponential distribution with mean  $1/\lambda$ .

If the integral  $\int_\Theta \lambda(\theta)d\theta$  is difficult to evaluate, one can also simulate a Poisson process using the **thinning theorem** [LS79]. Here, one needs to find a function  $\gamma(\theta)$  such that  $\gamma(\theta) \geq \lambda(\theta), \forall \theta$ , and such that it is easy to simulate from a rate- $\gamma(\theta)$  Poisson process. Suppose the result is  $\Psi = \{\psi_1, \dots, \psi_N\}$ . Since  $\gamma(\theta) \geq \lambda(\theta)$ ,  $\Psi$  is going to contain more events, and one *thins*  $\Psi$  by keeping each element  $\psi_i$  in  $\Psi$  with probability  $\lambda(\psi_i)/\gamma(\psi_i)$  (otherwise one discards it). Once can show that the set of surviving points is then a realization of a Poisson process with rate  $\lambda(\theta)$ .

The defining feature of the Poisson process is the assumption of independence among events. In many settings, this is inappropriate and unrealistic, and the knowledge of an event at some location  $\theta$  might suggest an reduced or elevated probability of events in neighboring areas. Point process

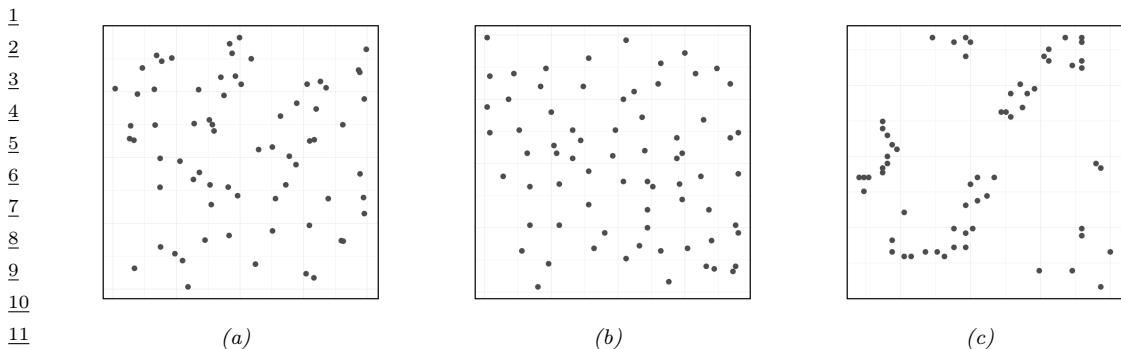


Figure 33.12: A realization of (a) a homogeneous Poisson process. (b) an underdispersed point process (Swedish pine sapling locations [Rip05]). (c) A realization of an overdispersed point process (California redwood tree locations [Rip77]).

satisfying the former property are called underdispersed (Figure 33.12(b)), while point processes satisfying the latter property are called overdispersed (Figure 33.12(c)). Examples of underdispersed point processes include the locations of trees or train stations, which tend to be more spread out than Poisson because of limited resources. An example of an overdispersed point process is earthquake locations, where aftershocks tend to occur in the vicinity of the main shock.

A simple approach to modeling overdispersed point processes is through hierarchical extensions of the Poisson process that allow the Poisson intensity function  $\lambda(\cdot)$  to be a random variable. Such models are called doubly-stochastic Poisson processes or Cox processes [CI80], and a common approach is to model the intensity  $\lambda(\cdot)$  via a Gaussian process. Note though that the Poisson process intensity function must be nonnegative, so that  $\lambda$  is often a transformed Gaussian process:

$$\lambda(\theta) = g(\ell(\theta)), \quad \ell(\theta) \sim \text{GP}. \quad (33.46)$$

Common examples for  $g$  include exponentiation, sigmoid transformation or just thresholding. Because of the smoothness of the unknown  $\lambda(\cdot)$ , observing an event at some location suggests events are likely in the neighborhood. Such models still do not capture direct interactions between point process events, rather, these are mediated through the unknown intensity functions, making them inappropriate for many applications. For instance, in neuroscience a neuron's spiking can be driven directly by past activity, and activity of other neurons in the network, rather than just through some shared stimulus  $\lambda(t)$ . Similarly, social media activity has a strong reciprocal component, where, for instance, emails sent out by a user might be in response to past activity, or activity of other users. The next few subsections show how one might explicitly model such interactions.

39

#### 40 33.8.2 Renewal process 41

42 **Renewal processes** are one class of models of repulsion and reinforcement for point processes defined on the real line (typically regarded as time). Recall that for a homogeneous Poisson process, 43 inter-event times follow an exponential distribution. The exponential distribution has the property of 44 **memorylessness**, where the time until the next event is independent of how far in the past the last 45 event occurred. That is, if  $\tau$  follows the exponential distribution, then for any  $\delta$  and  $\Delta > 0$ , we have 46

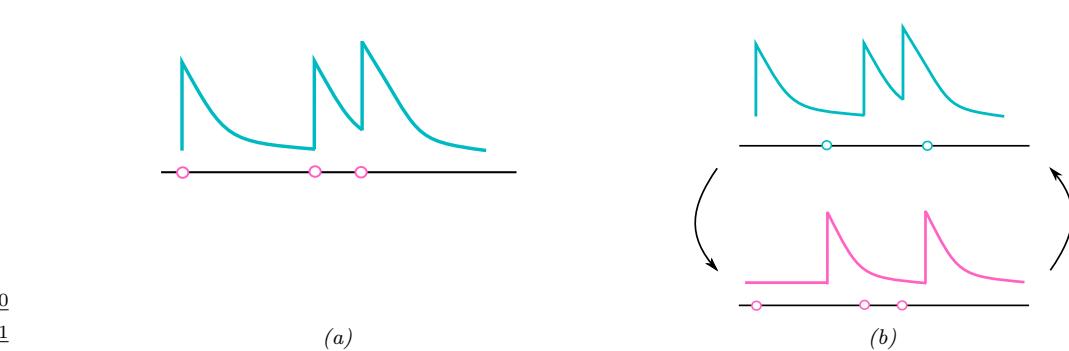


Figure 33.13: (a) A self-exciting Hawkes process. Each event causes a rise in the event rate, which can lead to burstiness. (b) A pair of mutually exciting Hawkes processes, events from each cause a rise in the event rate of the other.

$p(\tau > \Delta + \delta | \tau \geq \delta) = p(\tau > \Delta)$ . Renewal processes incorporate memory by allowing the interevent times to follow some general distribution on the positive reals. Examples include Gamma renewal processes and Weibull renewal processes, where interevent times follow the Gamma and Weibull distribution respectively. Both these processes include parameter settings that recover the Poisson process, but also allow **burstiness** and **refractoriness**. Burstiness refers to the phenomenon where, after an event has just occurred, one is more likely to see more events than a longer time afterwards. This is useful for modeling email activity for instance. Refractoriness refers to the opposite situation, where an event occurring implies new events temporarily less likely to occur. This is useful to model neural spiking activity, for instance, where after spiking, a neuron is depleted of resources, and requires a recovery period before it can fire again.

### 33.8.3 Hawkes process

Hawkes processes [Haw71] are another class of reinforcing point processes that have attracted much recent attention. Hawkes processes provide an intuitive framework for modeling reinforcement in point processes, through self-excitation when a single point process is involved, and through mutual excitation (sometimes called reciprocity) when collections of point processes are under study. The former, shown in Figure 33.13(a), is relevant when modeling bursty phenomena like visits to a hospital by an individual, or purchases and sales of a particular stock. The latter, displayed in Figure 33.13(b) is useful to characterize activity on social or biological networks, such as email communications or neuronal spiking activity. In both examples, each event serves as a trigger for subsequent bursts of activity. This is achieved by letting  $\lambda(t)$ , the event rate at time  $t$ , be a function of past activity or *event history*. Write  $\mathcal{H}(t) = \{t_k : t_k \leq t\}$  for the set of event times up to time  $t$ . Then the rate at time  $t$ , called the *conditional intensity function* at that time, is given by

$$\lambda(t|\mathcal{H}(t)) = \gamma + \sum_{t_k \in \mathcal{H}(t)} \phi(t - t_k), \quad (33.47)$$

where  $\gamma$  is called the **base-rate** and the function  $\phi(\cdot)$  is called the **triggering kernel**. The latter characterizes the excitatory effect of a past event on the current event rate. Figure 33.13 shows

1 the common situation where  $\phi(\cdot)$  is an exponential kernel  $\phi(\Delta) = \beta e^{-\Delta/\tau}$ ,  $\Delta \geq 0$ . Here, a new  
2 event causes a jump of magnitude  $\beta$  in the intensity  $\lambda(t)$ , with its excitatory influence decaying  
3 exponentially back to  $\gamma$ , with  $\tau$  the time-scale of the decay. For a multivariate Hawkes process, there  
4 are  $m$  point processes  $(N_1(t), N_2(t), \dots, N_m(t))$  associated with  $m$  users or nodes. Write  $\mathcal{H}_i(t)$  for  
5 the event history of the  $i$ th process at time  $t$ . Its conditional intensity can depend on all  $m$  event  
6 histories, taking the form  
7

$$\lambda_i(t|\{\mathcal{H}_j(t)\}_{j=1}^m) = \gamma_i + \sum_{j=1}^m \sum_{t_k \in H_j(t)} \phi_{ij}(t - t_k). \quad (33.48)$$

8  
9 More typically, the conditional intensity of user  $i$  can depend only on the event histories of those  
10 nodes ‘connected’ to it, with connectivity specified by a graph structure, and with  $\phi_{ij}(\cdot) = 0$  if there  
11 is no edge linking  $i$  and  $j$ . Alternately, events can have marks indicating whom they are sent to (e.g.  
12 recipients of an email), with each event only updating the conditional intensities of its recipients.

13 Simulating from a Hawkes process is a fairly straightforward extension of simulating from an  
14 inhomogeneous Poisson process. Consider the univariate Hawkes process, and suppose the last event  
15 occurred at time  $t_i$ . Then, until the next event occurs, at any time  $t > t_i$ , we have that  $\mathcal{H}(t) = \mathcal{H}(t_i)$ ,  
16 so that the conditional intensity  $\lambda(t|\mathcal{H}(t)) = \lambda(t|\mathcal{H}(t_i))$ . The next event time,  $t_{i+1}$ , is then just the  
17 time of the first event of a Poisson process with intensity  $\lambda(t|\mathcal{H}(t_i))$  on the interval  $[t_i, \infty)$ . With most  
18 choices of the kernel  $\phi$ ,  $t_{i+1}$  can easily be simulated, either by integrating  $\lambda(t|\mathcal{H}(t_i))$ , or by Poisson  
19 thinning. At time  $t_{i+1}$ , the history is updated to incorporate the new event, and the conditional  
20 intensity experiences a jump. The updated intensity is used to simulate the next event at some time  
21  $t_{i+2} > t_{i+1}$  from a rate- $\lambda(t|\mathcal{H}(t_{i+1}))$  Poisson process, and the process is repeated until the end of the  
22 observation interval. For the case of multivariate Hawkes processes, one has a collection of competing  
23 intensities  $\lambda_i(t|\{\mathcal{H}_j(t)\}_{j=1}^m)$ . The next event is the first event among all events produced by these  
24 intensities, after which the intensities are updated and the process is repeated. A realization  $\Psi$  from  
25 a Hawkes process has log-likelihood

$$\ell(\Psi) = \sum_{t^* \in \Psi} \log \lambda(t^*|\mathcal{H}(t^*)) - \int \lambda(t|\mathcal{H}(t)) dt. \quad (33.49)$$

26 This can typically be evaluated quite easily, so that maximum likelihood estimates of parameters like  
27 the base-rate as well as parameters of the excitation kernel can be obtained straightforwardly.

28 The Hawkes process as described is a fairly simple model, and there have been a number of  
29 extensions enriching its structure. An early example is [BBH12], where the authors considered the  
30 multivariate Hawkes process, now with an underlying clustering structure. Instead of each individual  
31 point process having its own conditional intensity function, each cluster has an intensity function  
32 shared by all point process assigned to it. The interaction kernels are also defined at the cluster level,  
33 with an event in process  $i$  causing a jump  $\phi_{c_i c}(\tau)$  in the intensity function of cluster  $c$  (where  $c_i$  is the  
34 cluster point process  $i$  belongs to). The cluster structure was modeled through a Dirichlet process,  
35 allowing the authors to learn the underlying clustering, as well as the inter-cluster interaction kernels  
36 from interaction data. In [Tan+16], the authors considered **marked** point processes, where event  $i$  at  
37 time  $t_i$  has some associated content  $y_i$  (for example, each event is a social media post, and  $y_i$  is the  
38 text associated with the post at time  $t_i$ ). The authors allowed the jump in the conditional intensity  
39 to depend on the associated mark, with the Hawkes kernel taking the form  $\phi(\Delta) = f(y_i) \exp(-\Delta/\tau)$ .

40

In that work, the authors modeled the function  $f$  with a Gaussian process, though other approaches, such as ones based on neural networks, are possible.

The neural Hawkes process [ME17] is a more fleshed out approach to modeling point processes using neural networks. This models event intensities through the state of a continuous-time LSTM, a modification the more standard discrete-time LSTMs from Section 16.3.3. Central to an LSTM is a memory cell  $\mathbf{c}_i$  to store long-term memory, summarizing the past until time step  $i$ . Continuous-time LSTMs include two long-term memory cells,  $\mathbf{c}_i$  and  $\tilde{\mathbf{c}}_i$ , summarizing the history until the  $i$ th Hawkes event. The first cell represents the starting value to which the intensity jumps after the  $i$ th event, and the second represents a baseline rate after the  $i$  event. These are both updated after each event, with  $\mathbf{c}(t)$ , the instantaneous rate at any intermediate time determined by  $\mathbf{c}_i$  decaying exponentially towards the baseline  $\tilde{\mathbf{c}}_i$ . This mechanism allows the intensity at any time to be influenced not just by the number of events in the past, but also the waiting times between them. It can be extended to marked point processes, where each event is also associated with a mark  $\mathbf{y}$ : now both a learned embedding of the mark as well as the time since the last event is used to update state and long-term memory. For more details, see also Du et al. [Du+16].

### 33.8.4 Gibbs point process

Gibbs point processes [MW07] from the statistical physics and spatial statistics literature provide a general framework for modeling interacting point processes on higher-dimensional spaces. Such spaces are more challenging than the real line, since there is now no ordering of points, and thus no natural notion of history affecting future activity. Instead, Gibbs point processes use an **energy function**  $E$  to quantify deviations from a Poisson process with rate 1. Specifically, under a Gibbs process, the probability density of any configuration  $\Psi$  with respect to a rate-1 Poisson process takes the form

$$P_\beta(\Psi) = \frac{1}{Z_\beta} \exp(-\beta E(\Psi)) \quad (33.50)$$

where  $\beta$  is the inverse-temperature parameter, and  $Z_\beta = \mathbb{E}_\pi[\exp(-\beta E(\Psi))]$  is the normalization constant (the expectation is with respect to  $\pi$ , the unit-rate Poisson process). Under some conditions on the energy function  $E$ , the expectation  $Z_\beta$  is finite, and  $P_\beta$  is a well-defined density, whose integral with respect to  $\pi$  is 1. While the above equation resembles a Markov random field, the domain of  $\Psi$  is much more complicated, and evaluating  $Z_\beta$  now involves solving an infinite dimensional integral.

Equation (33.50) states that configurations  $\Psi$  for which  $E(\Psi)$  is small are more likely than under a Poisson process, with  $\beta$  controlling how peaked this is. The most common energy functions are **pairwise potentials**, taking the form

$$E(\Psi) = \sum_{(s,s') \in \Psi} \phi(\|s - s'\|), \quad (33.51)$$

where the summation is over all pairs of events in  $\Psi$ , and  $\phi : \mathbb{R}^+ \rightarrow \mathbb{R} \cup \infty$ . The Strauss process is a specific example, with energy function specified by positive parameters  $a$  and  $R$  as

$$E(\Psi) = \sum_{s,s' \in \Psi} a \cdot \mathbb{I}(\|s - s'\| \leq R). \quad (33.52)$$

<sup>1</sup> This is a repulsive process that penalizes configurations with events separated by distance less than  
<sup>2</sup>  $R$ , and as  $a \rightarrow \infty$  becomes a **hardcore repulsive process**, forbidding configurations with two  
<sup>3</sup> points separated by less than  $R$ . More generally, the energy function can be piecewise-constant,  
<sup>4</sup> parametrized by a collection of pairs  $(a_1, R_1), \dots, (a_n, R_n)$ :  
<sup>5</sup>

$$\frac{6}{7} E(\Psi) = \sum_{s, s' \in \Psi} \sum_{i=1}^n a_i \cdot \mathbb{I}(\|s - s'\| \leq R_i). \quad (33.53)$$

$$\frac{8}{9}$$

<sup>10</sup> Another natural option is to use smooth functions like a squared exponential kernel. Gibbs point  
<sup>11</sup> processes can also involve higher-order interactions, examples being Geyer's triplet point process  
<sup>12</sup> (which penalizes occurrences of 3 events that are all within some distance  $R$ ), or area-interaction point  
<sup>13</sup> processes (that center disks of radius  $R$  on each event, calculate the area of the union of these disks,  
<sup>14</sup> and define  $E$  as some function of this area).

<sup>15</sup> While Gibbs point processes are flexible and interpretable point process models, the intractable  
<sup>16</sup> normalization constant  $Z_\beta$  makes estimating parameters like  $\beta$  a formidable challenge. In practice,  
<sup>17</sup> these models have to be fit using approximate approaches such as maximizing a pseudo-likelihood  
<sup>18</sup> function (instead of Equation (33.50)).  
<sup>19</sup>

### <sup>20</sup> <sup>21</sup> 33.8.5 Determinantal point process

<sup>22</sup> **Determinantal point processes** [Mac75; Bor; LMR15] or DPPs are another approach to modeling  
<sup>23</sup> repulsion, and have seen considerable popularity in the machine learning literature. Like any point  
<sup>24</sup> process, a DPP is a probability distribution over subsets of a fixed set  $\mathcal{S}$ , and in the DPP literature  
<sup>25</sup> this is often called the **ground set**. Point process applications typically have  $\mathcal{S}$  with uncountably  
<sup>26</sup> infinite cardinality (for example,  $\mathcal{S}$  could be the real line), although machine learning applications  
<sup>27</sup> of DPPs often focus on  $\mathcal{S}$  with a finite number of elements. For instance,  $\mathcal{S}$  could be a database of  
<sup>28</sup> images, a collection of news articles, or a group of individuals. A sample from a DPP is a random  
<sup>29</sup> subset of  $\mathcal{S}$ , produced, for instance, in response to a search query. The repulsive nature of DPPs  
<sup>30</sup> ensures diversity and parsimony in the returned subset. This could be useful, for example, to ensure  
<sup>31</sup> responses to a search query are not minor variations of the same image. Another application is  
<sup>32</sup> clustering, where a DPP serves as a prior over the number of clusters and their locations, with the  
<sup>33</sup> repulsiveness discouraging redundant clusters that are very similar to each other.

<sup>34</sup> DPPs are parametrized by a similarity kernel  $K$ , whose element  $K_{ij}$  gives the similarity between  
<sup>35</sup> elements  $i$  and  $j$  of the ground set. For finite ground sets,  $K$  is just a similarity matrix. DPPs require  
<sup>36</sup>  $K$  to be positive definite, with largest eigenvalue less than 1, that is  $0 \preceq K \preceq I$ . For simplicity,  $K$  is  
<sup>37</sup> often assumed symmetric, though this is not necessary. Given a kernel  $K$ , the associated DPP is  
<sup>38</sup> defined as follows: if  $Y$  is the random subset of  $\mathcal{S}$  drawn from the DPP, then the probability that  $Y$   
<sup>39</sup> contains any subset  $A$  of  $\mathcal{S}$  is given by  
<sup>40</sup>

$$\frac{41}{42} p(A \subseteq Y) = \det(K_A). \quad (33.54)$$

<sup>43</sup> Here  $K_A$  is the submatrix obtained by restricting  $K$  to rows and columns in  $A$ , and we define  
<sup>44</sup>  $\det(K_\phi) = 1$  for the empty set  $\phi$ . Observe that this probability is specified exactly, and not just up  
<sup>45</sup> to a normalization constant. We immediately see that  $Y$  contains the empty set with probability one  
<sup>46</sup> (this is trivially true since the empty set is a subset of every set). We also see that the probability  
<sup>47</sup>

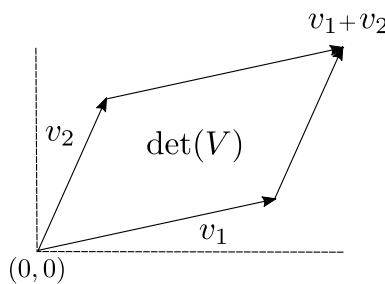


Figure 33.14: The determinant of a matrix  $V$  is the volume of the parallelogram spanned by the columns of  $V$  and the origin.

that element  $i$  is selected in  $Y$  is the  $i$ th diagonal element of  $K$ :

$$p(i \in Y) = \det(K_{ii}) = K_{ii}, \quad (33.55)$$

so that the diagonal of  $K$  gives the inclusion probabilities of the individual elements in  $\mathcal{S}$ . More interestingly, the probability a pair of elements  $\{i, j\}$  are both contained in  $Y$  is given by

$$p(\{i, j\} \subseteq Y) = K_{ii}K_{jj} - K_{ij}^2. \quad (33.56)$$

The first term  $K_{ii}K_{jj}$  is the probability of including  $i$  and  $j$  if they were independent, and this probability is adjusted by subtracting  $K_{ij}^2$ , a measure of similarity between  $i$  and  $j$ . This is the source of repulsiveness or diversity in DPPs. More generally, the determinant of a set of vectors is the volume of the parallelogram spanned by them and the origin (see Figure 33.14), and by making the probability of inclusion proportional to the volume, DPPs encourage diversity. We mention for completeness that for uncountable ground sets like a Euclidean space, the determinant  $\det(K_A)$  now gives the **product density function** of a realization  $A$ . This generalizes the intensity of a Poisson process to account for interactions between point process events, and we refer the interested reader to Lavancier, Møller, and Rubak [LMR15] for more details.

Equation (33.54) characterizes the marginal probabilities of a determinantal point process: what is the probability that a realization  $Y$  contains some subset of  $\mathcal{S}$ . More often, one is interested in the probability that the realization  $Y$  equals some subset of  $\mathcal{S}$ . The latter is of interest for simulation, inference and parameter learning with DPPs. For this, it is typical to work with a narrower class of DPPs called **L-ensembles** [Bor; KT+12]. Like general DPPs, such a process is characterized by a positive semidefinite matrix  $L$ , with the probability that  $Y$  equals some configuration  $A$  given by:

$$p(Y = A) \propto \det(L_A). \quad (33.57)$$

Note that this probability is specified only upto a normalization constant, and so we do not need to upperbound eigenvalues of  $L$ . In fact, it is not hard to show that the normalizer is given by  $(I + \det(L))$ , so that

$$p(Y = A) = \frac{\det(L_A)}{I + \det(L)}. \quad (33.58)$$

1 A similar calculation can be used to show that  $p(A \subseteq Y) = \det(K)$ , where  $K = L(I + L)^{-1} =$   
2  $I - (I + L)^{-1}$ , showing that L-ensembles are indeed special kind of DPP. Equation (33.58) and  
3 Equation (33.54) allow parameters of the DPP to estimated from realizations of a point process,  
4 typically by gradient descent. Without additional structure, naively calculating determinants is  
5 cubic in the cardinality of  $\mathcal{S}$ , and this represents a substantial saving when one considers the number  
6 of possible subsets of  $\mathcal{S}$ . When even cubic scaling is too expensive, a number of approximation  
7 approaches can be adopted, and these are often closely related to approaches to solve the cubic cost  
8 of Gaussian processes.

10 So far, we have only discussed how to calculate the probability of samples from a DPP. Simulating  
11 from a DPP, while straightforward, is a bit less intuitive, and we refer the reader to [LMR15; KT+12].  
12 At a high level, these approaches use the eigenstructure of  $K$  to express a DPP as a mixture of  
13 **determinantal projection point processes**. The latter are DPPs whose similarity kernel has  
14 binary eigenvalues (either 0 or 1) and are easier to sample from. Observe that any eigenvalue  $\lambda_i$  of  
15 the similarity kernel  $K$  must lie in the interval  $[0, 1]$ . This allows us to generate a random similarity  
16 kernel  $\hat{K}$  with the same eigenvectors as  $K$  but with binary eigenvalues as follows: for each  $i$ , replace  
17 eigenvalue  $\lambda_i$  of  $K$  with a binary variable  $\hat{\lambda}_i$  simulated from a  $\text{Bernoulli}(\lambda_i)$  distribution. One can  
18 show that the DPP simulated from  $\hat{K}$  after simulating  $\hat{K}$  from  $K$  in this fashion is distributed  
19 exactly as a DPP with kernel  $K$ . We refer the reader to [LMR15] for details on how to simulate a  
20 determinantal projection point process with similarity kernel  $\hat{K}$ .

212223242526272829303132333435363738394041424344454647

# 34 Representation learning (Unfinished)

*This chapter was written by Ben Poole and Simon Kornblith.*

## 34.1 CLIP



# 35 Interpretability

*This chapter was written by Been Kim and Finale Doshi-Velez.*

## 35.1 Introduction

As machine learning models become increasingly commonplace, there exists increasing pressure to ensure that these models' behaviors align with our values and expectations. It is essential that models that automate even mundane tasks—from paperwork processing and serving information to flagging potential fraud—don't cause harms to their users or to society at large; for models that are used to inform potentially life-changing decisions, such as autonomous cars or health advice, the stakes are even higher.

However, while some harms—such as costs due to the model's false positive and false negative rate on training data—are easy to quantify—it is not easy to ensure that models are not dangerous, unfair, or too narrowly focused on a single objective in complex environments. Objectives can often be easily exploited (e.g., reward hacking). The value of any metric (whether it is the objective or a false negative rate) can be hard to estimate precisely in complex settings e.g., with limited data or a distribution shift. Modern machine learning methods—while they have accomplished much—still often learn shortcuts that do not generalize to new situations even as users start attributing notions like language or visual understanding to them [Gei+20b]. As a result, not only might unexpected, irreversible harms may occur (e.g., an incorrect medical procedure), but biases and more subtle safety issues may go unnoticed for long periods of time until sufficient reporting data accrues [Amo+16]. While it may not be possible to fully eliminate these harms, a broad, holistic approach to validation that includes expert inspection (via interpretability) and empirical analysis (e.g., more traditional statistical measures) will be critical in mitigating them.

Finally, even when we have satisfactory models for our tasks(e.g., models that make good predictions or compress the data well) we may be interested in understanding *why* they work to gain scientific and operational insights that go beyond the ML original task or to predict unseen failure cases. For example, one might gain insights in language acquisition and learning by asking why a language model performs so well; understanding why patient data cluster along particular axes may result in a better understanding of disease and the common treatment pathways. Ultimately, interpretation help humans to communicate better with machines to accomplish our tasks better in many metrics we care by communicating to or teaming up with machines.

In this chapter, we lay out the role and terminologies in interpretable ML before introducing methods, properties and evaluation of interpretability methods.

---

1 2 **35.1.1 The Role of Interpretability**

3

4 As noted above, ensuring that models have the behaviors that we want is a challenging task that  
5 requires a holistic approach to design and validation. In some cases, the desired behavior can be  
6 guaranteed by design, such as certain notions of privacy via differentially-private learning algorithms  
7 or some chosen mathematical metric of fairness. In other cases, tracking various metrics, such  
8 as adverse events or subgroup error rates, may be the appropriate and sufficient way to identify  
9 concerns and demonstrate that they have been resolved. However, in many cases, the goal may be  
10 fundamentally impossible to fully specify and thus formalize. In such cases, human inspection of the  
11 machine learning model may be necessary. Below we describe several examples.

12

13 **Blindspot Discovery.** Inspection may reveal **blindspots** in our modeling, objective, or data  
14 [Bad+18; Zec+18; Gur+18] For example, suppose a company has trained a machine learning system  
15 for credit scoring. The model was trained on a relatively affluent, middle-aged population, and now  
16 the company is considering using it on a college population—a new population, on which they have  
17 no data. Suppose that inspection of the model reveals that it relies heavily on the properties of the  
18 applicant’s home and car. Not only might this suggest that the model might not transfer well to  
19 the new college population, but it might encourage us to check for bias in the existing application  
20 because we know historical biases have prevented certain populations from achieving home ownership  
21(something that a purely quantitative definition of fairness may not be able to recognize). Indeed,  
22 the most common application of interpretability in industry settings is for engineers to debug models  
23 and make deployment decisions [Pai].

24

25 **Novel Insights.** Inspection may catalyze **novel insights**. For example, suppose an algorithm  
26 determines that surgical procedures fall into three clusters, and the surgeries in one of the clusters  
27 of patients seem to consistently take longer than expected. A human inspecting these clusters may  
28 determine that a key factor in the cluster with the worst delays is that those surgeries happen in a  
29 more distant part of the hospital (a feature not in the original dataset), and then hypothesize that  
30 transit time for clinical staff may be affecting performance.

31

32 **Human+ML Teaming.** Inspectability may empower effective **human+ML interaction and**  
33 **teaming**. For example, suppose an anxiety treatment recommendation algorithm reveals that one of  
34 the key factors determining the recommendation was that the patient has insomnia. The patient  
35 reports that they no longer have trouble sleeping. Then they could re-run the algorithm with that  
36 input changed to get a better recommendation. More broadly, if a human can provide feedback to  
37 the model (e.g., correcting an incorrect input or assumption) or if they can incorporate information  
38 from the ML model into their own decision-making (e.g., having an accurate sense of the risk of a  
39 poor outcome, while trying to optimize for a good one) the human+ML team may be able to produce  
40 better combined performance than either alone (e.g. [Ame+19; Kam16]).

41

42 **Individual-Level Recourse.** A common setting under the law is that one needs to demonstrate  
43 that a specific harm or error happened in a specific context. A local explanation can help provide  
44 information needed for an individual to seek recourse. For example, if a loan applicant knows what  
45 features were used to deny them a loan, they have a starting point to argue that an error might have  
46 been made, or that the algorithm denied them unjustly. For this reason, inspectability is sometimes  
47

1 a legal requirement [Zer+19; GF17; Cou16].  
2

3 As we look at the examples above, we see that one common element is that **interpretability**  
4 is needed when we need to combine human insights with the ML algorithm to get to  
5 the ultimate goal.<sup>1</sup> However, looking at the list above also emphasizes that beyond this very  
6 basic commonality, **each application and task represents very different needs**. One would  
7 not expect a scientist seeking to glean insights from a clustering on molecules (a case in which we  
8 want to understand global patterns—such as all molecules with certain loop structures are more  
9 stable—and are not under time pressure) to require the same kind of information as a clinician  
10 seeking to make a specific treatment decision for a specific patient (a case in which we need to make  
11 a local decision and are under time pressure). This brings us to our most important point: The best  
12 form of explanation depends on the context; interpretability is a means to an end.  
13

### 14 35.1.2 Terminology and Framework

15 In broad strokes, “to interpret means to explain or present in understandable terms,”[Mer] [to a  
16 human], and understanding involves an alignment of mental models. In interpretable machine  
17 learning, that alignment is between what (perhaps part of) the machine learning model is doing and  
18 what the user thinks the model is doing.  
19

20 Specifically, not only does the interpretable machine learning ecosystem include standard machine  
21 learning (e.g., a prediction task), it also crucially includes what information is provided to the  
22 human user, in what context, and the user’s ultimate goal. The broader *socio-technical system*—the  
23 collection of interactions between human, social, organizational, and technical (hardware and software)  
24 factors—in which the machine learning system is situated cannot be ignored [Sel+19]. The goal of  
25 interpretable machine learning is to help a user do *their* task, with *their* cognitive strengths and  
26 weaknesses, with *their* focus and distractions [Mil19]. Below we define the key terms of this expanded  
27 ecosystem and describe how they relate to each other. Before continuing, however, we note that the  
28 field of interpretable machine learning is relatively new, and a consensus around terminology is still  
29 evolving. Thus, it is always important to be precise about what one means in one’s usage.  
30

31 Two key **social** or **human-factors** concepts in interpretable machine learning are the *context* and  
32 the *end-task*.

33 **Context.** We use the term *context* to describe the setting in which an interpretable machine  
34 learning system will be used. What is the setting? Who is the user? What information do they  
35 already have about the setting? What constraints are present on their time, cognition, or attention?  
36 We will use the terms context and application interchangeably [Sta].  
37

38 **End-task.** We use the term *end-task* to refer to the user’s ultimate goal. What are they ultimately  
39 trying to achieve? Are they trying to help human experts to be more efficient or to do a recourse?  
40 We use end-task and downstream task interchangeably in this chapter.  
41

42 Three key **technical** concepts in interpretable machine learning are the *method*, the *metrics*, and  
43 the *properties* of the methods.  
44

---

45 1. We emphasize that interpretability is different from manipulation or persuasion: interpretability assumes that the  
46 model will not intentionally deceive nor aim to convince users with a goal in mind.  
47

1

2

3   **Method.** What is the *method* used to provide interpretability? We use the term *explanation*  
4 to mean whatever is provided by the machine learning system to the user—that is, *interpretable*  
5 *machine learning* involves *providing explanations*. If the explanation is the model itself, we call the  
6 method *inherently interpretable* or *interpretable by design*. In other cases, the model may be too  
7 complex for a human to inspect in the required context: perhaps it is a large neural network that  
8 no human could expect to understand; perhaps it is a medium-sized decision tree that could be  
9 inspected if one had twenty minutes but not if one needs to make a decision in two. In such cases,  
10 the explanation may be a *partial view* into the model, one that is ideally suited for performing the  
11 end-task in the given context. We emphasize that even inherently interpretable models do not reveal  
12 everything: one might be able to fully inspect the function (e.g. a two-node decision tree) but not  
13 know what data it was trained on or which data points were most influential.

14

15   **Metrics.** How is the interpretability method evaluated? Evaluation is one of the most essential  
16 and challenging aspects of interpretable machine learning, because we are interested in the **End-task**  
17 performance of the *human*, when explanation is provided. We call this the *downstream performance*.  
18 Just as different goals in ML require different metrics (e.g., positive predictive value, log likelihood,  
19 AUC), different **contexts** and **end-tasks** will have different metrics. For instance, the model with  
20 the best predictive performance (on some standard ML loss, such as log likelihood) may not be the  
21 model that results in the best downstream performance.

22

23   **Properties.** What characteristics do the explanation have in relation to the model and the  
24 end-tasks? For example, an explanation might have the property that it correctly describes of the  
25 impact of modifying an input on the predictions, but only for models of low curvature (that is,  
26 models that are relatively slowly varying). Different **contexts** and different **end-tasks** might require  
27 different properties. For example, in a credit risk assessment scenario, the above property about  
28 correctly describing the impact of modifying inputs might be critical for checking for effects of bias or  
29 errors. Some other scenarios may require highly sparsity in explanations. In this way, properties serve  
30 as a glue between interpretability methods and end-tasks: properties allow us to specify and quantify  
31 aspects relevant to our ultimate end-task goals, and then we can make sure that our interpretability  
32 method has those properties.

33

34   **How they all relate.** Formulating an interpretable machine learning problem generally starts by  
35 specifying the context and the end-task. Together the context and the end-task imply what metrics  
36 are appropriate to evaluate the downstream performance on the end-task and, ideally, suggest what  
37 properties will be important in the explanation. Meanwhile, the context also determines the data  
38 and training metric to train the ML model. The appropriate choice of explanation methods will  
39 depend on the model and properties desired, and it will be evaluated with respect to the end-task  
40 metric to determine the downstream performance. Figure 35.1 shows these relationships.

41   While there are many challenges in interpretable machine learning, including computing expla-  
42 nations and optimizing interpretable models, creating explanations with certain properties, and  
43 understanding the associated human factors, a grand challenge in interpretable machine learning is to  
44 1) develop a general understanding of what properties are needed for different contexts and end-tasks,  
45 and 2) identify and create interpretable machine learning methods that have those properties.

46

47

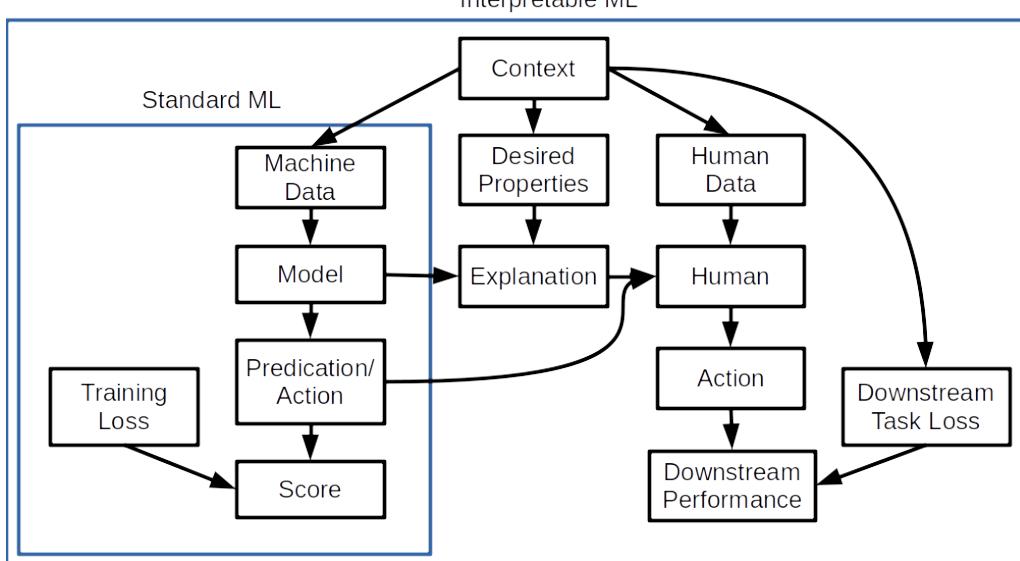


Figure 35.1: The Interpretable Machine Learning ecosystem. While standard machine learning can often abstract away elements of the context and consider only the process of learning models given a data distribution and a loss, interpretable machine is inextricably tied to a socio-technical context.

**A Simple Example.** In the following sections, we will expand upon methods for interpretability, metrics for evaluation, and types of properties. First, however, we provide a simple example connecting all of the concepts we discussed above.

Suppose our context is that we have a lemonade stand, and our end-task is to understand when the stand is most successful in order to prioritize which days it is worth setting it up. (We have heard that sometimes machine learning models latch on to incorrect mechanisms and want to check before using the model to guide our lemonade stand business strategy.) Our metric for the downstream performance is whether we make the correct decision about whether to use this model to decide when to open our lemonade stand; this could be quantified as the amount of profit that we make by opening on busy days and being closed on quiet days.

To train our model, we collect data on two input features—the average temperature for the day (measured in degrees Fahrenheit) and the cleanliness of the sidewalk near our stand (measured as a proportion of the sidewalk that is free of litter, between 0 and 1)—and the output feature of whether the day was profitable. We realize that two models seem to fit the data approximately equally well:

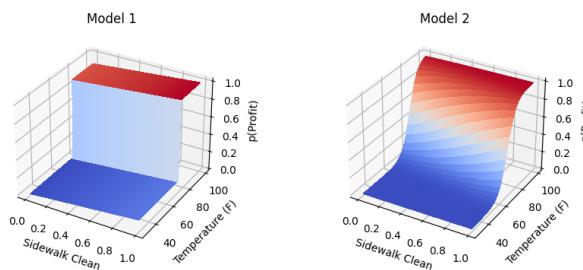
Model 1:

$$p(\text{profit}) = .9 * (\text{temperature} > 75) + .1(\text{howCleanSidewalk}) \quad (35.1)$$

Model 2:

$$p(\text{profit}) = \sigma(.9(\text{temperature} - 75)/\text{maxTemperature} + .1(\text{howCleanSidewalk} - .5)) \quad (35.2)$$

1  
2  
3  
4  
5  
6  
7  
8  
9  
10



11 *Figure 35.2: Models described in the simple example. Both of these models have the same qualitative*  
12 *characteristics, but different explanation methods will describe these models quite differently—and in ways*  
13 *that could potentially confuse someone unfamiliar with the explanation is computed.*

14  
15  
16

17 These models are illustrated in Figure 35.2. Both of these models are inherently interpretable in  
18 the sense that they are easy to inspect and understand. Both also rely mostly on the temperature,  
19 which seems reasonable. Also note that these are not causal models; while causal model would work  
20 for our end-task of determining our business strategy, we decided that we are not confident that  
21 there will not be unknown confounding factors (common assumption in causal models).

22 For the sake of this example, suppose that the models above were blackboxes, and we could only  
23 request partial views of it. We decide to ask the model for the most important features. Let us see  
24 what happens when we consider two different ways of computing important features.<sup>2</sup>

25 Our first (feature-based) explanation method computes, for each training point, whether individually  
26 changing each feature to its max or min changes the prediction. Important features are those that  
27 change the prediction for many training points. One can think of this explanation method as a  
28 variant of computing feature importance based on how important a feature is to the coalition that  
29 produces the output (i.e., prediction). In this case, both models will report temperature to be the  
30 dominating feature. If we used this explanation, we would correctly conclude that both models use  
31 the features in a sensible way (and thus may be worth considering for deciding when to open our  
32 lemonade stand).

33 Our second (feature-based) explanation method computes, for each training point, the magnitude  
34 of the derivatives of the output with respect to the inputs. Important features are those that have a  
35 large sum of absolute derivatives across the training set. One can think of this explanation method  
36 as a variant of computing feature importances based on local geometry. In this case, Model 2 will  
37 still report that temperature has higher derivatives. However, Model 1, which has very similar  
38 behavior to Model 2, will report that sidewalk cleanliness is the dominating feature because the  
39 derivative with respect to temperature is zero nearly everywhere. If we used this explanation to  
40 vet our models, we would incorrectly conclude that Model 1 over-relied on a feature we believe is  
41 relatively unimportant—and more generally, incorrectly conclude that Model 1 and Model 2 rely on  
42 different features.

43 What happened? The key is that it's not about a method being right or wrong; it's about whether  
44 the property of a chosen method is right or wrong for the context and the end-task. Here, the first

45  
46  
47

46 2. In the remainder of the chapter we will describe many other ways of creating and computing explanations.

explanation had the property of fidelity with respect to identifying features that, if changed, will affect the prediction, whereas the second explanation had the property of correctly identifying features have the most curvature. In this example, since our goal is to determine our business strategy, the property that affects the prediction when feature is modified (first method) was a better fit than measuring sensitivity of the feature (the second method). (Note: Other properties in addition to fidelity may be important for this end-task. This example is just a simplest one.)

## 35.2 Methods for Interpretable Machine Learning

There exist many methods for interpretable machine learning. Each method has different properties and the right choice will depend on context and end-tasks; as we noted in Section 35.1.2, the grand challenge in interpretable machine learning is determining what kinds of properties are needed for what contexts, and what explanation methods satisfy those properties. Thus, one should consider this section a high-level snapshot of the rapidly changing options of methods that one may want to choose for interpretable machine learning.

### 35.2.1 Inherently Interpretable Models: The Model is its Explanation

There are certain classes of models that we can consider inherently interpretable: a person can inspect the full model and its internals in the finest granularity, and with reasonable effort, understand how inputs—typically input features—become outputs.<sup>3</sup> Specifically, we define inherently interpretable models as those that require no additional process or proxies in order for them to be used as explanation for the end-task. For example, a model using sparse rules for prediction may itself (i.e., rules) be the explanation without any extra processing. However, the "inherence"—the absence of any additional process or proxies between the model and the user—alone does not guarantee improvement of downstream tasks—inherently interpretable models must still be evaluated to determine whether they help the user achieve their end-task.

Inherently interpretable models fall into two main categories: sparse (or otherwise compact) models and logic-based models.

**Compact** or **sparse** feature-based models include various kinds of sparse regressions. Basic models in this category include LASSO or other L1-regularized regressions. More advanced models in this category include super-sparse linear integer models and other checklist models [DMV15; UTR14]. While simple functionally, sparsity has its drawbacks when it comes to inspection and interpretation: for example, if features are correlated, then the model may be forced to pick one and not the other, or even pick both with different signs. To handle these issues, as well as to express more complex functions, some models in this category impose hierarchical or modular structures in which each component is still relatively compact and can be inspected. Examples include topic models (e.g. [BNJ03b]), (small) discrete time series models (e.g. [FHDV20]), and generalized additive models (e.g. [HT17]).

<sup>3</sup>. There may be other questions, such as how training data influenced the model, which may still require additional computation or information.

<sup>1</sup> **Logic-based** models use logical statements as basis; such as decision-trees [Bre+17], are often  
<sup>2</sup> most well-known interpretable models. Other logic-based models include decision lists [Riv87; WR15;  
<sup>3</sup> Let+15a; Ang+18; DMV15] , decision tables, and decision sets [Hau+10; Wan+17a; LBL16; Mal+17;  
<sup>4</sup> Bén+21] and logic programming [MDR94]. A broader discussion, as well as a survey of user studies  
<sup>5</sup> on these methods, can be found in [Fre14]. Logic-based models have the advantage of being able to  
<sup>6</sup> easily model non-linear functions, but they often have trouble with handling continuously changing  
<sup>7</sup> values (e.g. expressing a linear function vs. a step-wise constant function). Like the compact models,  
<sup>8</sup> hierarchies and other forms of modularity can be used to extend the expressivity of the model while  
<sup>9</sup> keeping it human-inspectable. For example, one can define a new concept as a formula based on  
<sup>10</sup> some literals, and then use that concept in the next formula.

<sup>11</sup> When using inherently interpretable models, three key decisions need to be made: the choice of  
<sup>12</sup> the model class, how to manage uninterpretable input features, and the choice of optimization method.

<sup>13</sup>

<sup>14</sup> **Decision: Model Class.** Since the model is its own explanation, the decision on the model  
<sup>15</sup> class becomes the decision on the form of explanation. Thus, we need to consider both whether the  
<sup>16</sup> model class is a good choice for modeling the data as well as providing the necessary information  
<sup>17</sup> to the user in their context. For example, if one chooses to use a linear model to describe one's  
<sup>18</sup> data, then it is important that the intended users can understand or manipulate the linear model  
<sup>19</sup> in the desired context. A user may interact differently with, for example, a linear model versus  
<sup>20</sup> a decision tree. Moreover, if the fitting process produces a model that is too large to human-  
<sup>21</sup> inspectable, then it is no longer inherently interpretable, even if it belongs to one of the model  
<sup>22</sup> classes described above. Finally, a learnt model might be inherently interpretable for one context  
<sup>23</sup> and end-tasks but not another: For example, a decision tree with even a relatively “small” depth  
<sup>24</sup> of 10 nodes may be hard for a human to inspect, though perhaps they may be able to follow a sim-  
<sup>25</sup> ple path down it to try to understand whether a specific decision was made for an appropriate purpose.

<sup>26</sup>

<sup>27</sup> **Decision: Optimization Methods for Training.** The kinds of model classes that are typically  
<sup>28</sup> inherently interpretable often require more advanced methods for optimization: compact, sparse, and  
<sup>29</sup> logic-based models all involve discrete parameters that must be determined. Fortunately, there is a  
<sup>30</sup> long and continuing history of research for optimizing such models, including directly via various  
<sup>31</sup> optimization programs, via various relaxation and rounding techniques, and various search-based  
<sup>32</sup> approaches. Another popular optimization approach is via distillation or mimics: one first trains  
<sup>33</sup> a complex model (e.g., neural network) and then use complex model's output to train a simpler  
<sup>34</sup> model (sometimes called student model); the simpler model is trained to mimic the behavior of  
<sup>35</sup> the complex model. Of course, in this case, only the simpler model is inherently interpretable; the  
<sup>36</sup> complex model is only used for producing training data for the simpler model before being discarded.  
<sup>37</sup> These optimization techniques are beyond the scope of this chapter but covered in other chapters  
<sup>38</sup> and optimization textbooks.

<sup>39</sup>

<sup>40</sup> **Decision: How to Manage Uninterpretable Input Features.** Sometimes the input features  
<sup>41</sup> themselves are not directly interpretable (e.g. pixels of an image or individual amplitudes spectrogram);  
<sup>42</sup> only collections of inputs have semantic meaning for human users. This situation challenges our  
<sup>43</sup> ability to create inherently interpretable models, but also explanations in general.

<sup>44</sup>

<sup>45</sup> To address this issue, more advanced methods attempt to add a “concept” layer that first converts  
<sup>46</sup> the uninterpretable raw input to a set of human-interpretable concept features, and then relate those  
<sup>47</sup>

concepts to a model’s decision (e.g., prediction)[Kim+18a; Bau+17]; the latter stage can still be inherently interpretable. A Concept can be built from a set of features or input patterns. For example, one could map a pattern of spectrogram of semantically meaningful sound (e.g., people chatting, cups clinking), and then from those sound to a sound classification (e.g., cafe). While promising, one must ensure that the initial data-to-concept mapping truly maps the raw data to concepts as the user understands them, no more and no less (e.g. sneaking in additional signals). Creating and validating that machine-derived concepts to correspond to a semantically meaningful human concepts is an open research challenge.

**When might we want to consider inherently interpretable models? When not?** Inherently interpretable models have several advantages over other approaches. When the model is its explanation, one need not worry about whether the explanation is faithful to the model or whether it provides the right partial view of the model for the intended task. Thus, an inherently interpretable model can likely be used for a greater variety of interpretation tasks. Relatedly, if a person vets the model and finds nothing amiss, they might feel more confident about avoiding surprises. For all these reasons, inherently interpretable models have been advocated for in high-stakes scenarios, as well as generally being the first go-to to try[Rud19].

That said, these models do have their drawbacks. They typically require more specialized optimization approaches. With appropriate optimization, inherently interpretable models can often match the performance of more complex models, but there are domains—in particular, images and text—in which deep models or other more complex models typically give significantly higher performance. Trying to fit complex behavior with a too-simple function may result not only with high bias in the trained model, but the use of features that people may still find stories to explain even if they are irrelevant [Lun+20]. Finally, in an industry setting, one may not have the opportunity to change an already-implemented model (e.g., a legacy, business critical model that has been tuned over decades would run into some resistance).

Lastly, we note that just because a model is inherently interpretable, it does not guard against all kinds of surprises: as noted in Section 35.1, interpretability is just one form of validation mechanism. For example, if the data distribution shifts, then one may start observing model behavior that no one expected.

### 35.2.2 Semi-Inherently Interpretable Models: Example-Based Methods

Example-based models uses examples (e.g., instances from training set) as basis for explanation. For example, they classify a new input by finding similar instances or representative instances in the training set with that decision, based on a chosen distance metric. They may then apply voting scheme amongst those similar training instances. Like logic-based models, example-based models can describe highly non-linear boundaries.

K-nearest neighbors is one of the best known model in this class, but there have been many extensions, including methods to identify exemplars for predicted classes/clusters (e.g. [KRS14; KL17b; JL15a; FD07b][RT16; Arn+10]), generating exemplars (e.g. [Li+17c]), sophisticated embedding and distance metric methods to define similarity between instances (e.g.[PM18a]), and parts-based approaches that first decompose an instance into parts and can find neighbors or exemplars between the parts (e.g. [Che+18b]).

On one hand, individual decisions made by example-based methods seem fully inspectable: one

1 can provide the user with exactly the training instances (including their labels) that were used to  
2 classify a particular input in a particular way. However, it may be difficult to convey a potentially  
3 complex distance metric, and the user may extrapolate the reasons that the examples are rated as  
4 similar incorrectly (e.g., what features made the examples similar). It is also often difficult to convey  
5 the intuition behind the global decision boundary using examples.  
6

7

### 8 35.2.3 Post-hoc or Joint training: The Explanation gives a Partial View of the 9 Model

10

11 While there exist many approaches for creating inherently interpretable models, inherently inter-  
12 pretabile models are clearly only a subset of all machine learning models. Various circumstances  
13 may require working with a model that is not inherently interpretable: as noted above, large neural  
14 models have demonstrated large performance benefits for certain kinds of data (e.g. images and text);  
15 one might have to work with a legacy, business critical model that has been tuned for decades; one  
16 might be trying to understand a system of interconnected models.

17 In these cases, explanations can still be extracted; however, the view that the explanation gives into  
18 the model will necessarily be partial: the explanation may only be an approximation of the model or  
19 be otherwise incomplete. Thus, more decisions have to be made about what the explanation should  
20 contain—and how it should be computed—so that it provides the information that is necessary for  
21 the specific context and end-task. It is crucial that one understands the abilities and limitations of  
22 these partial explanation methods [Sla+20; Yeh+19a; Kin+19; Ade+20a].

23 Below, we split these (interconnected) decisions into two broad categories—what the explanation  
24 consists of (which also requires knowing whom it is for, the context) and how the explanation is  
25 computed given the trained model.

26

#### 27 35.2.3.1 What does the explanation consist of?

28

29 One set of decisions center around what the explanation consists of, including what properties it has  
30 given a context (includes whom it is for) and an end-task. One key decision is the form: Will the  
31 explanation be a list of important features? A local model? Another is formalizing what properties  
32 the explanation must have—for example, in what sense should that list of key features reflect the  
33 true model? Finally, one must decide the scope of the explanation: Global, local, or somewhere in  
34 between? We expand on each of these below; the right choice, as always, will depend on user—whom  
35 the explanation is for—and their end-task.

36

37 **Decision: Form of the Explanation.** In the case of inherently interpretable models, the choice  
38 of the model class used to fit the data was also the choice of the form of the explanation. Now, the  
39 model class and the explanation are two different entities. Thus, regardless of what model class  
40 one uses to fit the data, one also needs to determine the form of the explanation, that is, the way  
41 in which the information will be delivered. For example, the model could be a deep network; the  
42 explanation in the form of a decision tree.

43 Works in interpretable machine learning have used a large variety of forms of explanations. The  
44 form could be a list of “important” input features [RSG16b; Lun+20; STY17; Smi+17; FV17] or  
45 “important” concepts [Kim+18a; Bau+20; Bau+18]. Or it could be a simpler model that approximates  
46 the complex model (e.g. a local linear approximation, an approximating rule set)[FH17; BKB17;  
47

1 2 3 4 5 Aga+21b; Yin+19c]. Another choice could be a set of similar or prototypical examples [KRS14;  
AA18; Li+17c; JL15a; JL15b; Arn+10]. Finally, one can choose whether the explanation should  
include a contrast against an alternative (also sometimes described as a counterfactual explanation)  
[Goy+19; WMR18; Kar+20a] or include or influential examples [KL17b].

6 Different forms of explanations will facilitate different tasks in different contexts. For example, a  
7 contrastive explanation of why treatment A is better than treatment B may help a clinician determine  
8 whether to choose treatment A over treatment B—but that same contrast between treatments A and  
9 B may not help if the question is deciding between treatments A and C. Similarly, an explanation  
10 that provides the most important features for a prediction may not provide information about how  
11 those features interact. Given the large number of choices, observations or literature on what people  
12 communicate in the desired context may provide some guidance. For example, if the domain is one  
13 that involves making quick, high-stakes decisions, one might turn to the literature on recognition-  
14 primed decision making [Kle17] which discusses how trauma nurses and firefighters explain their  
15 decisions.

16 **Decision: Determining what Properties the Context Needs.** Each of the forms of expla-  
17 nations above have different levels of expressivity (e.g., consider the different expressivity between a  
18 local linear model and a local decision tree). For each form, there will also be many ways to compute  
19 an explanation of that form (more on this in Section 35.2.3.2). How do we choose amongst all of these  
20 different ways to compute the explanation? Rather than jump to particular forms of computation  
21 (e.g., gradients vs. local perturbations to determine feature importances) the first step should be to  
22 determine what properties are needed from the explanation; the properties provide the specification  
23 for the computation.

24 Specifying properties is especially important because different forms of explanation may not only  
25 have different intrinsic properties, but they may have different properties depending on the underlying  
26 model (that they are trying to explain). For example, if the function that represents the underlying  
27 model is relatively smooth, then a feature-based explanation relying on local gradients may be fairly  
28 faithful to the original model; however, if the function has strong spikes, the same feature-based  
29 explanation (with the same computation) may no longer be faithful to the model; producing con-  
30 flicting results at best. Similarly, whether the data lie on a low-dimensional manifold or not will  
31 determine the properties of an approach that identifies key features based on local perturbations.  
32 Once the desired properties are determined, one can determine what kind of computation is necessary  
33 to achieve them. We will list commonly desirable properties in Section 35.3.

35 **Decision: Scope of the Explanation: Global or Local.** The final major decision regarding  
36 the parameters of the explanation is its scope: global or local.

38 **Local explanation:** In some cases, we may only need the explanation to interrogate an existing  
39 model about a specific decision: For example, why was this image predicted as a bird? Why was  
40 this patient predicted to have disease X? Local explanations are useful if one needs to dig into a  
41 particularly important decision that a model made, for example to see if an error was made or  
42 determine what could have been done differently to produce a different outcome (recourse).

44 Local explanations can take many forms. They may be a locally-fit simpler model in the neighbor-  
45 hood of the data of interest (e.g. LIME [RSG16b]). A family of methods called saliency maps or  
46 attribution maps [STY17; Smi+17; ZF14; Sel+17; Erh+09; Spr+14; Shr+16] use similar approach

<sup>1</sup> by estimating feature importances via first-order derivatives. A local explanation may also consist  
<sup>2</sup> of representative examples, including identifying which data points e.g., from training set, were  
<sup>3</sup> most influential for a particular decision [KL17b] or identifying nearby data points with different  
<sup>4</sup> predictions (e.g., counterfactual examples) [MRW19; LHR20; Kar+20a]. With all local explanation  
<sup>5</sup> methods, one needs to be cautious not to overgeneralize beyond the scope of the explanation, as well  
<sup>6</sup> as not overfit user's mental model of the model based on a few local explanations; again, post-hoc  
<sup>7</sup> explanations are necessarily partial.  
<sup>8</sup>

<sup>9</sup>  
<sup>10</sup> **Global explanation:** In other cases, we may desire insight into the model as a whole or for a  
<sup>11</sup> collection of data points (e.g., all inputs predicted to one class). Such a broader scope may be needed,  
<sup>12</sup> for example, if the end-task is to make a deployment decision; then it's not just the decision process  
<sup>13</sup> for a few inputs that matter, one wants to vet the model's overall decision process. That said, it is  
<sup>14</sup> important to remember that a global explanation is still a partial view of the underlying model; there  
<sup>15</sup> could still be qualities of the model—either highly local, or simply lost in the approximation—that  
<sup>16</sup> the global explanation misses.

<sup>17</sup> Global explanations can take many forms. One choice to fit a simpler model (e.g. an inherently  
<sup>18</sup> interpretable model) that somehow approximate the original model (e.g. [HVD14]). One can identify  
<sup>19</sup> concepts or features that affect decision across many inputs (e.g. [Kim+18b]). Another approach  
<sup>20</sup> is to provide a carefully set of representative examples[Yeh+18]. These examples might be chosen  
<sup>21</sup> to be somehow characteristic of, or providing coverage of, a class (e.g. [AA18]), to draw attention  
<sup>22</sup> to decision boundaries (e.g. [Zhu+18]), or to identify inputs particularly influential in training the  
<sup>23</sup> model.

<sup>24</sup>

### <sup>25</sup> 35.2.3.2 How the explanation is computed

<sup>26</sup>

<sup>27</sup> Another set of decisions have to do with how the explanation is computed.

<sup>28</sup>

<sup>29</sup> **Decision: Computation of Explanation.** Decisions that we discussed so far (form of explana-  
<sup>30</sup>tions, properties, local vs. global) are our decisions/desiderata for our contexts and end-task. In  
<sup>31</sup>order to produce the explanation, we now need to decide how to compute it. Together with other  
<sup>32</sup>decisions, this decision finalizes the definition of "explanation" in each method. Therefore, it is crucial  
<sup>33</sup>to iterate to optimize this decision for the context and end-task in mind.

<sup>34</sup> For example, suppose one is seeking to identify the most "important" input features that changes a  
<sup>35</sup>prediction, the computation would differ depending on their definition of importance. One definition  
<sup>36</sup>of importance (and therefore computation decision) might be the smallest region in an image when  
<sup>37</sup>changed, prediction changes—a perturbation-based analysis. In case of local perturbations alone, we  
<sup>38</sup>need to then decide how much to perturb (while remained within training distribution) and in what  
<sup>39</sup>segments (e.g., perturb each pixels, a set of pixels at a time) [SVZ13; DG17; FV17; DSZ16; Adl+18;  
<sup>40</sup>Bac+15]. Another way to compute "importance" is by measuring how often the input feature is  
<sup>41</sup>part of a "winning coalition" that drives the prediction, e.g. a Shapley or Banzaf score[LL17]. Now  
<sup>42</sup>suppose one is now seeking to identify the most "important" input features in terms of sensitivity  
<sup>43</sup>(e.g., largest gradients of the output with respect to the input feature). Even then, there are many  
<sup>44</sup>other computational decisions one can make such as done in [STY17; Smi+17; Sel+17; Erh+09;  
<sup>45</sup>Shr+16]. Each of these choices of how to compute the explanation will have different properties, as  
<sup>46</sup>well as require different amounts of computation.

<sup>47</sup>

Similar issues come up with other forms of explanations. For example, if an example-based form is chosen, then one has to determine what it needs to be ‘similar’ (e.g. cosine similarity between activations? a uniform L2 ball of a certain size between inputs?) or otherwise ‘representative.’ In another example, there are many different ways to obtain counterfactuals; by defining a distance function and loss terms to describe what “counterfactual example” is [WMR17; LHR20], or formulating it as a SAT problem[Kar+20a], or by following causal inference framework [Kus+18].

**Decision: Joint Training vs. Post-hoc Application.** So far, we have described our partial explanation techniques as extracting some information from an already-trained model. This approach is called deriving a post-hoc explanation. As noted above, post-hoc, partial explanations may have some limitations: for example, an explanation based on a local linear approximation may be great if the model is generally smooth, but provide little insight if the model has high curvature—this is not because the partial explanation is wrong, but because the view that local gradients provides isn’t sufficient for curvy true decision boundary.

Another approach to getting explanations to have the properties we desire is to train the model and the explanation jointly. For example, A regularizer that penalizes violations of desired properties can help steer the overall optimization process towards learning models that both perform well and are amenable to the kind of explanation that we desire [Plu+20]. It is often possible to find such a model because most complex model classes have multiple high-performing optima [Bre01].

The choice of regularization will depend on the desired properties, the form of the explanation, and its computation. For example, we may know of the kinds of features that people are likely to be using themselves for a task (e.g., lower frequency vs. higher frequency features in image classifiers [Wan+20a]) to make machine learned classification processes match those used by people. We may want the local explanation to use or not use certain input dimensions or to be sparse or otherwise compact (e.g. a small decision tree) while still being faithful to the underlying model [RHDV17; Shu+19; Vel+17; Nei+18; Wu+19b; Plu+20]; this category may also include attention models [JW19; WP19] depending on what properties are desired. We may also have constraints on the properties of concepts or other intermediate features [AMJ18b; Koh+20; Hen+16; BH20; CBR20; Don+17b].

When choosing between a post-hoc explanation or joint training, one key consideration is that joint training assumes that one can re-train the model or the system of interest. In many cases in practice, this may not be possible. Replacing a complex and well-validated system in deployment for a decade may not be possible or take prohibitively long time or not allowed. In that case, one can still extract approximated explanations using post-hoc methods. Finally, a joint optimization, even when it can be performed, is not a panacea: optimization for some properties may result in unexpected violations of other (unspecified but desired) properties. For this reason, explanations from jointly trained model is still often partial.

**When might we want to consider post-hoc methods, and when not?** The advantage of post-hoc interpretability methods is that they can be applied to any model. This family of methods is especially useful in real-world scenarios where one needs to work with a system that contains many models as its parts, where one cannot expect to replace the whole system with one model. These approaches can also provide at least some broader knowledge about the model to identify unexpected concerns.

That said, post-hoc explanations, as approximations of the true model, may not be fully faithful to

1 the model nor do they cover the model completely. As such, an explanation method tailored for one  
2 context may not be transferable in another; even in the intended context, there may be blindspots  
3 about the model that the explanation misses completely. For these reasons, in high stakes situations,  
4 one should attempt to use an inherently interpretable model first if possible [Rud19]. In all situations  
5 when post-hoc explanations are used, one must keep in mind that they are only one tool in a broader  
6 accountability toolkit and warn users appropriately.  
7

8

#### 9 **35.2.4 Transparency and Visualization**

10

11 The scope of interpretable machine learning is usually centered around methods that expose the  
12 process by which a trained model makes a decision. However, the behavior of a model closely depends  
13 on the training data, how the training data were collected and processed, and how the model was  
14 trained and tested. Conveying to a human these other aspects of what goes into the creation of  
15 a model can be as important as explaining the trained model itself. While a full discussion of  
16 transparency and visualization is outside the scope of this chapter, we provide a brief discussion here  
17 to describe these important adjacent concepts.

18

19 Transparency is an umbrella term for the many things that one could expose about the modeling  
20 process and its context. Interpreting models is one aspect. However, one could also be transparent  
21 about other aspects, such as the data collection process or the training process (e.g. [Geb+21;  
22 Mit+19; Dnp]). There are also situations in which a trained model is released (whether or not it is  
23 inherently interpretable), and thus the software can be inspected and run directly.

24

25 Visualization is one way to create transparency. One can visualize the data directly, various  
26 aspects of the model’s process (e.g. [Str+17]), and create interactive/static visualizations that can  
27 convey more than text or code descriptions [ZF14; OMS17; MOT15; Ngu+16; Hoh+20]. Finally,  
28 in the specific context of interpretable machine learning, how the explanation is presented—the  
29 visualization—can make a large difference in how easily users can consume it. Even something as  
30 simple as a rule list has many choices of layout, highlighting, and other organization.

31

32 **When might we want to consider transparency and visualization? When not?** In many  
33 cases, the trouble with a model comes not from the model itself, but parts in its training pipeline. For  
34 example, the data it was trained on could cause problems—if policing data contain historical bias, then  
35 predictions of crime hot spots based on that data will be biased. Similarly, if clinicians only order  
36 tests when they are concerned about a patient’s condition, then a model trained to predict risk based  
37 on tests ordered will only recapitulate what the clinicians already know. Transparency—knowing  
38 about the global properties of data, and how training and testing were performed can help identify  
39 these issues, which would cause problems regardless of which model was used.

40 Of course, inspecting the data and the model generation process is something that takes time and  
41 attention. Thus, visualizations and other descriptions to increase transparency are best-suited to  
42 situations in which a human inspector wants to understand complex patterns and potential sources  
43 of trouble without much time pressure (e.g., prior to starting a project). These methods are not  
44 well-suited for situations in which a specific decision must be made in a relatively short amount of  
45 time, e.g. providing decision-support to a clinician at the bedside.

46 Finally, transparency in the form of making code available can potentially assist in understanding  
47

1 how a model works, identifying bugs, and allow independent testing by third party (e.g, testing with  
2 new set of inputs, evaluating counterfactuals in different testing distributions) However, if a model is  
3 sufficiently complex, as many modern models are, then simply having access to the code may not be  
4 enough for a human to gain sufficient understanding for their task.  
5

### 7 35.3 Properties: The Abstraction Between Context and Method 8

9 Recall from the Terminology and Framework in Section 35.1.2 that the context and end-task determine  
10 what properties of the explanation. For example, in a high-stakes setting—such as advising on  
11 interventions for an unstable patient—it may be important that the explanation completely and  
12 accurately reflects the model (fidelity). In contrast, in a discovery-oriented setting, it might be more  
13 important for any explanation to allow for efficient iterative refinement, revealing different aspects  
14 of the model in turn (interactivity). From these examples, it should be clear that not all contexts  
15 and end-tasks need all properties, and the lack of a key property may result in poor downstream  
16 performance.

17 While the research is still evolving, there exists a growing informal understanding about how  
18 properties may work as an abstraction between methods and contexts. Many interpretability methods  
19 from Section 35.2 share the same properties, and methods with the same properties may have similar  
20 downstream performance in a specific end-task and context. If two contexts and end-tasks require  
21 the same properties, then a method that works well for one may work well for the other. A method  
22 with properties optimal for one downstream could miserably fail in another context.  
23

24 **How to find desired properties?** Of course, identifying what properties are important for a  
25 particular context and end-task is not trivial. Recall that identifying what properties are important  
26 for what contexts, end-tasks, and downstream performance metrics is one facet of the grand challenge  
27 of interpretable machine learning. For the present, the process of identifying the correct properties  
28 will likely require iteration via user studies. However, iterating over properties is still a much smaller  
29 space than iterating over methods. In particular, knowing what the currently focused properties are  
30 can aid user study design. For example, if one wants to test whether the sparsity of the explanation  
31 is key to good downstream performance, one could intentionally create explanations of varying levels  
32 of sparsity to test that hypothesis. This is a much more precise knob than exhaustively trying out  
33 different explanation methods across hyperparameter space.

34 Below, we first describe examples of properties that have been discussed in the interpretable  
35 machine learning literature. Many of these properties are purely computational—that is, they can be  
36 determined purely from the model and the explanation—while a few have some user-centric elements.  
37 Next we list examples of properties of explanation from cognitive science (on human to human  
38 explanations) and human-computer interaction (machine to human explanations). Some of these  
39 properties have current analogs in the machine learning list, while others may serve as inspiration for  
40 areas to formalize.  
41

#### 42 35.3.1 Properties of Explanations from Interpretable Machine Learning 43

44 Many lists of potentially-important properties of interpretable machine learning models have been  
45 compiled, sometimes using different terms for similar concepts and sometimes using the similar terms  
46 for different concepts. Below we list some commonly-described properties of explanations, knowing  
47

1  
2 that this list will evolve over time as the field advances.

3  
4 **Compactness, Sparsity** (e.g. as described in [Lip18; Mur+19]). In general, an explanation  
5 must be small enough such that the user can process it within the constraints of the task (e.g. how  
6 quickly a decision must be made). Sparsity generally corresponds to some notion of smallness (a  
7 few features, a few parameters,  $L1$  norm etc.), whereas compactness generally carries an additional  
8 notion of not including anything irrelevant (that is, even if the explanation is small enough, it could  
9 be made smaller). Each must be formalized for the context.

10  
11 **Faithfulness, Fidelity** (e.g. as described in [JG20; JG21]). When the explanation is only a  
12 partial view of the model, how well does it match the model? There are many ways to make this  
13 notion precise. For example, suppose a mimic (simple model) is used to provide a global explanation  
14 of a more complex model. One possible measure of faithfulness could be whether the mimic gives the  
15 same outputs as the original. Another could be whether the mimic has the same first derivatives  
16 (local slope) as the original. In the context of a local explanation consisting of the key features for a  
17 prediction, one could measure faithfulness by whether the prediction changes if the supposedly impor-  
18 tant features are flipped. Another measure could check to make sure the prediction does not change if  
19 a supposedly unimportant feature is flipped. The appropriate formalization will depend on the context.

20  
21 **Completeness** (e.g. as described in [Yeh+19b]). If the explanation is not the model, does it still  
22 include all of the relevant elements? For example, if an explanation consists of important features  
23 for a prediction, does it include all of them, or leave some out? Moreover, if the explanation uses  
24 derived quantities that are not the raw input features—for example, some notion of higher-level  
25 concepts—are they expressive enough to explain all possible directions of variation that could change  
26 the prediction? Note that one can have faithful explanation but not complete (e.g., while flipping  
27 each unimportant features do not change prediction—faithful, the explanation may fail to include  
28 that flipping a selected set of them do change the prediction).

29  
30 **Stability** (e.g. as described in [AMJ18a]) To what extent are the explanations similar for similar  
31 inputs? Note that the underlying model will naturally affect whether the explanation can be stable  
32 and faithful. For example, if the underlying model has high curvature and the explanation has limited  
33 expressiveness, then it may not be possible to have an explanation that is stable and faithful.

34  
35 **Actionability** (e.g. as described in [Kar+20b; Poy+20]). Actionability implies filtering the  
36 content of the explanation to focus on only aspects of the model that the user might be able  
37 to intervene on. For example, if a patient is predicted to be at high risk of heart disease, an  
38 actionable explanation might only include mutable factors such as exercise, and not immutable  
39 factors such as age or genetics. The notion of recourse corresponds to actionability in a justice context.

40  
41 **Modularity** (e.g. as described in [Lip18; Mur+19]). Modularity implies that the explanation can  
42 be broken down into understandable parts. While modularity does not guarantee that the user can  
43 explain the system as a whole, for more complex models, modular explanations—where the user can  
44 inspect each part—could be the most effective way to provide a reasonable level of insight into the  
45 model’s workings.

46  
47

**Interactivity.** (e.g., [Ten+20]) Does the explanation allow the user to ask questions, such as how the explanation changes for a related input, or how an output changes given a change in input? In some contexts, providing everything that a user might want or need to know from the start might be overwhelming, but it might be possible to provide a way for the user to navigate the information about the model in a way that they choose.

**Translucence** (e.g. as described in [SF20; Lia+19]). Is the explanation clear about its limitations? For example, if a linear model is locally fit around a particular input of interest and used to explain a deep model, is there a mechanism that reports that this explanation may be limited if there are strong feature interactions around that input? We emphasize that translucence is about exposing limitations in the explanation, rather than the model. The goal of the explanation—and all our other accountability methods—is to expose the limitations of the model.

**Simulability** (e.g. as described in [Lip18; Mur+19]). A model is simulable if a user can, given the model and an input, compute the output (within any constraints of time and cognition). In the context of explanations, the explanation must be a "simulable model". For example, a list of features would not be simulable by itself, because a list of features alone does not imply a computation from features to prediction. A user would have to make some assumptions (e.g. by assuming a linear model) to predict the output of the model. However, an explanation in the form of a decision tree does include a computation process—following the logic of the tree—and might be simulable as long as the tree was not too deep. The latter examples also point out an important difference between compactness and simulability: if an explanation is too large, it may not be simulable. However, just because an explanation is compact—such as a short list of features—does not mean that a person can compute the model's output with it.

It may seem that simulability is different from the properties we have listed so far because its definition involves human input. However, in practice, we may often know what kinds of explanations are easy for people to simulate (e.g. decision trees with short path lengths, rule lists with small formulas, etc.). This knowledge can then be turned into a purely computational training constraint where we seek explanations that are known to be simulable. In this sense, computational formalizations of simulability are no different than computational formalizations of other properties such as compactness, where there is a human-centric element that requires us to determine how compact is compact enough.

**Alignment to the User's Vocabulary and Mental Model.** (e.g., as described in [Kim+18a]) Is the content of the explanation designed for the user? For example, an explanation in terms of parameter variances and influential points may be comprehensible to an engineer debugging a lending model but not to a lay user trying to get a loan. Similarly, an explanation given in the semantics a user knows—such as in terms of medical conditions vs. raw sensor readings to a clinician—help the explanations to be placed in the context of user's expert knowledge and decision-making guidelines (modified quote from [Clo+19]).

Like simulability, this property is more human-centric. However, just as before, we can imagine an abstraction between eliciting vocabulary and mental models from users—that is, determining how they define their terms and how to think—and ensuring that an explanation is provided in alignment with whatever that elicited user vocabulary and mental model is.

Once desired properties are identified, we need to operationalize them. For example, if sparsity is a

1 desired property, would using L1 norm enough? Or something more sophisticated loss term needs to  
2 be designed? This decision will necessarily be human-centric: how small an explanation needs to be,  
3 or in what ways it needs to be small, is a decision that needs to consider how people will be using the  
4 explanation. Once operationalized, most properties can be optimized for computationally. That said,  
5 the properties should be evaluated with context and end-task, model and the chosen explanation  
6 methods. Once evaluation returns, one may revisit the method and models decision depending on  
7 the importance of the properties to the performance of the user on the end-task.

8 Finally, we emphasize that the ability to achieve a particular property will depend on the *intrinsic*  
9 complexity of the model. For example, the behavior of a highly nonlinear model with interactions  
10 between the inputs will, in general, be harder to understand than a linear model. No matter how we  
11 try to explain it, if we are trying to explain something complicated, then users will have a harder  
12 time understanding it.

13

### 14 15 35.3.2 Properties of Explanations from Cognitive Science

16 While work in interpretable machine learning have developed lists of properties largely from a  
17 computational perspective, in particular, the relationship between the model and the explanation, the  
18 fields of cognitive science and human-computer interaction have examined what people consider good  
19 properties of an explanation for a long time, that is the relationship between the explanation and the  
20 human. These more human-centered properties may be ones that researchers in machine learning  
21 may be less aware of, yet essential for the explanation to do its job of communicating information to  
22 a human.

23 Below we list several of these properties. Just as with the previous list, different properties may be  
24 important in different contexts—unsurprisingly, the literature on human explanation concurs that  
25 the explanation must fit the context [VF+80], therefore there is no one optimal explanation for all  
26 contexts. Finally, we emphasize that human explanations are also social constructs that happen as  
27 part of a conversation, and that they often include post-hoc rationalizations and other biases. In  
28 other words, one must carefully decide on desired properties for context and end-task in mind and  
29 weigh in potential human biases, rather than deciding "because humans sometimes do it". We focus  
30 on properties of explanations that help users achieve their goals.

31

32 **Soundness** (e.g., as described in [Kul+13]). Explanations should contain nothing but the truth  
33 with respect to whatever they are describing. Soundness corresponds to notions of *compactness* and  
34 *faithfulness* above.

35

36 **Completeness** (e.g., as described in [Kul+13]). Explanations should contain the whole truth  
37 with respect to whatever they are describing. Completeness corresponds to notions of *completeness*  
38 and *faithfulness* above.

39

40 **Contrastiveness** (e.g., as described in [Mil19]). Contrastive explanations provide information of  
41 how something differs from an alternate decision or prediction. For example, instead of providing a  
42 list of features for why a particular drug is recommended, it might provide a list of features that for  
43 why one drug is recommended over another. Contrastiveness relates to notions of *actionability* above,  
44 and more generally explanation types that include *counterfactuals*.

45

46

47

1

2     **Generality.** Overall, people understand that an explanation for one context may not apply in  
3 another. That said, there is an expectation that an explanation should reflect some underlying  
4 mechanism or principle and will thus apply to similar cases—for whatever notion of similar is in  
5 the person’s mental model. Explanations that do not generalize to similar cases may be misinterpreted.

6

7

8     **Simplicity.** All of the above being equal, simpler explanations are generally preferred. Simplicity  
9 relates to notions of *sparsity* and *complexity* above.

10

11

12

13

14

15

16

17

18 Finally, the cognitive science literature also notes that explanations are often goal directed. This  
19 matches notion of explanation in ML as information that helps a person use a prediction better, give  
20 understanding by providing appropriate features, or otherwise do well on their end-task. Different  
21 kinds of content or forms may help with different goals, and thus human explanations take many forms.  
22 Forms of explanations include deductive-nomological (i.e. a logical proofs) [HO48], providing some  
23 sense of underlying mechanism [BA05; Gle02; CO06], and conveying understanding [Kei06]. Knowing  
24 these forms can help us consider what options might be best among different sets of interpretable  
25 machine learning methods.

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

## 35.4 Evaluation of Interpretable Machine Learning Models

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

1 also serve as intermediate evaluations and sanity checks to identify undesirable explanation behavior  
2 prior to a more expensive user study-based evaluation.  
3

4  
5 **User studies with people.** The second category of evaluation involves user studies with  
6 humans. Rigorous, carefully-designed, quantitative and qualitative user studies measure how well  
7 an interpretable machine learning method enables the user to complete their end-task in a given  
8 context. We emphasize that performing a rigorous, well-designed user study in a real context is  
9 significant work—much more so than computing a test likelihood on benchmark datasets. It requires  
10 significant asks of not only the researchers but also the target users. Methods for different contexts  
11 will also have different evaluation challenges: while a system designed to assist with optimizing  
12 music recommendations might be testable on a wide population, a system designed to help a particle  
13 physicist identify new kinds of interactions might only be tested with one or two physicists because  
14 people with that expertise are hard to find. In all cases, the evaluation can be done rigorously with  
15 proper attention to the experimental design.  
16

#### 17 **35.4.1 Computational Evaluation: Does the Method have Desired Properties?**

18  
19 While the ultimate measure of interpretability is whether the method successfully empowers the  
20 user to perform their task, properties can serve as a valuable abstraction. For example, checking  
21 whether an explanation has the right computational and desired properties can ensure that the  
22 method works as expected (e.g., no obviously odd behaviors, no implementation errors) and iterate to  
23 make computational improvement (e.g., better optimization methods given quantitative metric of a  
24 property) before conducting expensive human experiments. Checking for specific properties can also  
25 help pinpoint in what way an explanation is falling short, which may be less clear from a user study  
26 due to confounding . For example, if we know from the context that the explanation must be faithful  
27 in a particular way, we can computationally check whether an explanation achieves that property,  
28 as well as whether one explanation scores better on that property than another. Computational  
29 checks can also to ensure whether properties that held for one model continue to hold when applied  
30 to another model.

31 There may also be cases where user studies are not required to demonstrate the value of a new  
32 method in interpretable machine learning. For example, there may be situation in which an existing  
33 method has already been demonstrated to have certain properties—such as a faithful, compact  
34 decision set—such that they are useful in certain contexts. However, this method may have the  
35 computational challenge of computing (e.g., computational efficiency, or reliability across different  
36 datasets or model classes) the explanation. In this case, one can demonstrate methodological  
37 improvements purely computationally.

38 In some cases, one might be able to mathematically prove that an explanation has certain prop-  
39 erties, while in others the test must be empirical. For empirical testing, one strategy is to use a  
40 hypothesis-based sanity check; if we think a phenomenon X should never occur (hypothesis), we  
41 can try to test whether we can create situations where X may occur. If it does, then the method  
42 fails this sanity check. Another umbrella strategy is to create datasets with known characteristics or  
43 ground truth. These could be purely synthetic constructions (e.g. generated tables with intentionally  
44 correlated features), semi-synthetic approaches (e.g. intentionally changing the labels on an image  
45 dataset), or taking slices of a real dataset (e.g. introduce intentional bias by only selecting real  
46 image, label pairs that are of outdoor environments). Note that these tests only reflect lower-  
47

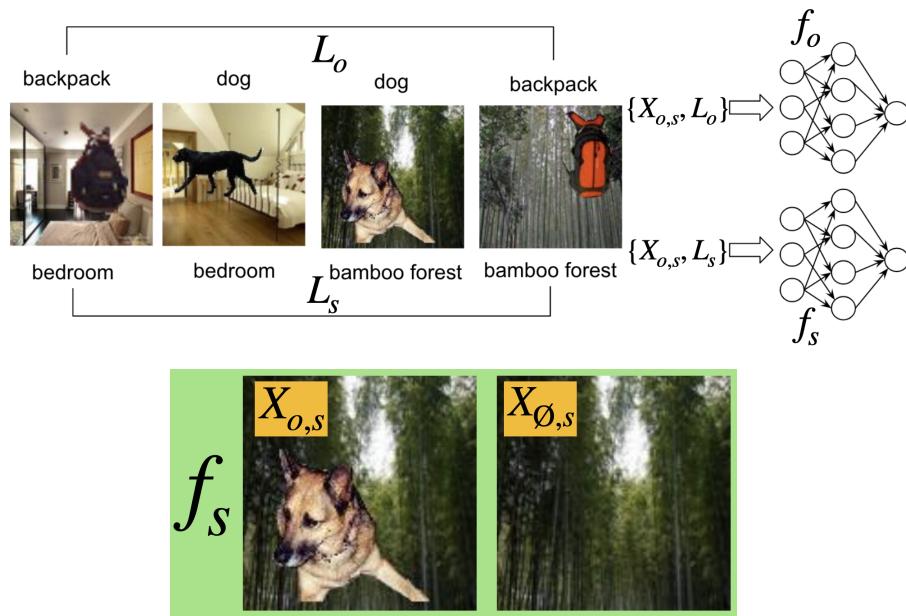


Figure 35.3: An example of computational evaluation using (semi-)synthetic datasets from [YK19]: foreground images (e.g. dogs, backpacks) are placed on different backgrounds (e.g. indoors, outdoors) to test what an explanation is looking for.

bounds: If a method finds what it should find in these constructed datasets or sanity checks, then there may still be missing blindspots, but if it fails in these settings, we know that something is wrong.

**Examples of Sanity Checks.** One strategy is to do hypothesis-based sanity check; if we think a phenomenon X should not occur (hypothesis), we can try to test whether we can create situations where X may occur. If it does, then the method fails this sanity check. This type of strategy often services as a lower-bound, bare minimum check by asking outside-of-the-box questions, and often to reveal some surprising failure modes of explanation methods.

For example, [Ade+20a] operates under the assumption that a faithful explanation should be function of a model's prediction. The hypothesis is that the explanation should significantly change when comparing a trained model to an untrained model (where prediction is random). They show that many existing methods fail to pass this sanity check (Figure 35.4).

In another example, [Kin+19] hypothesize that a faithful explanation should invariant to input transformations that do not affect model predictions or weights, such as constant shift of inputs (e.g., all inputs are added by 10). This hypothesis can be seen as testing both faithfulness and stability properties. This work shows that some methods fail this sanity check.

In a way, adversarial attacks [GAZ19] on explanation methods could be seen as a sanity check; one that would be harder for many methods to pass. [GAZ19] shows that two perceptively indistinguishable inputs with the same predicted label can be assigned very different interpretations.

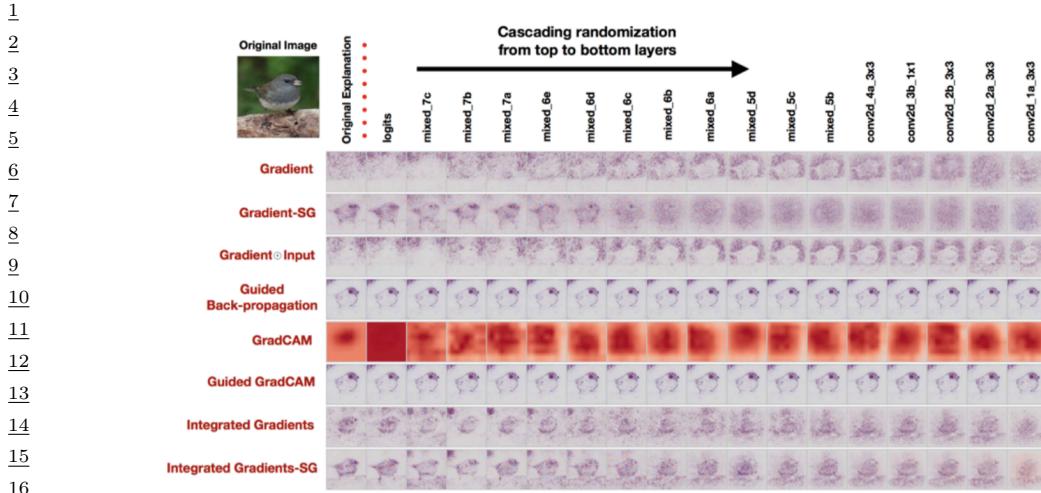


Figure 35.4: Interpretability methods (each row) and their explanations as we randomize layers starting from the logits, and cumulatively to the bottom layer (each column), in the context of image classification task. The rightmost column is showing a completely randomized network. Most methods output similar explanations for the first two columns; one predicts the bird, and the other predicts random. This sanity check tests the hypothesis that the explanation should significantly change (quantitatively and qualitatively) when comparing a trained model and an untrained model [Ade+20a].

**Examples using (semi-)Synthetic Datasets.** While sanity checks may often lead to binary conclusion, using (semi-)synthetic datasets for computational evaluation come put quantitative metrics that can be compared across methods.

We use the work of [YK19] as an example. Here, the authors were interested in explanations with the properties of compactness and faithfulness: it should not identify features as important if they are not (i.e., explanations should have low false positive). The "importance" is defined as a set of features that are correlated with prediction (not causation). To test fidelity under this definition, authors generate a set of image dataset with intentional correlation; an object often co-occur with particular background (an object is a image patch from a real picture, pasted onto another real background picture, see Figure 35.3). They generate a few sets of dataset with varying level of the correlation between objects and backgrounds. Note that each dataset comes with two labels per image: for an object and a background.

Note that correlation of features do not imply causal relationship—that those features 'caused' the prediction. What we do know with certainty is *relative* importance: for example, we can compare two models one trained to classify objects and one trained to classify backgrounds (left, Figure 35.3). If a model is trained to classify objects and they all happen to have the same background, the background should be less important than in a model trained to classify backgrounds (Model Contrast Score). We can also deduce similar relative importance given one model: a model is given two images with (image A) and without (image B) unimportant set of features (e.g., a model trained to predict backgrounds. All images contain a dog. Two images are given to the model one with and without a dog) (see right

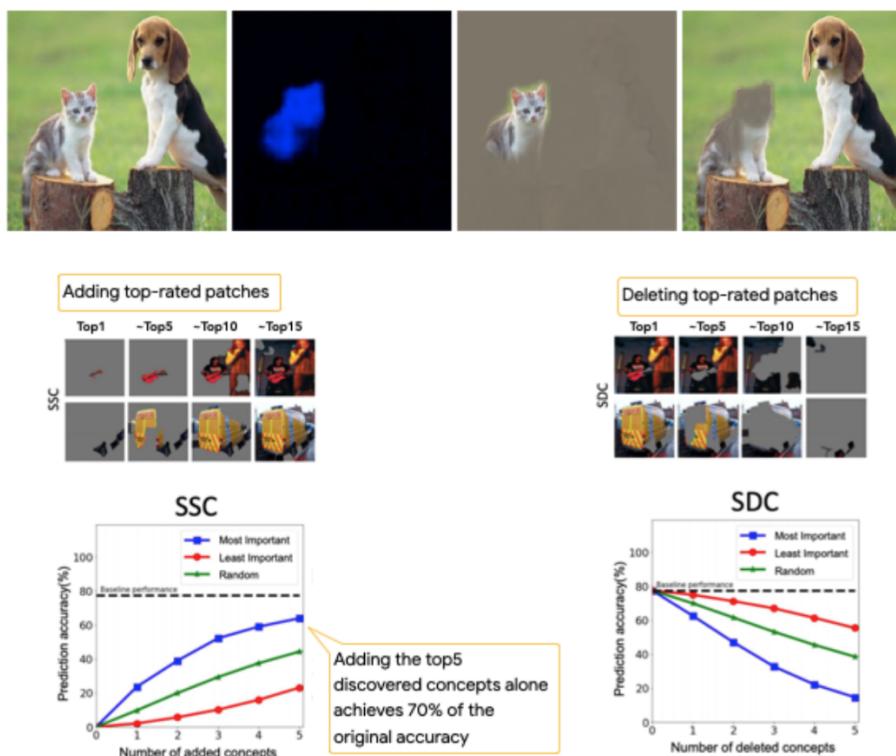


Figure 35.5: Examples of computational evaluation with real datasets from [DG17; Gho+19]. For example, one would expect that adding or deleting patches rated as most relevant for an image classification would have a large effect on the classification compared to patches not rated as important.

Figure 35.3). When comparing attributions in image A and B, a more faithful method must attribute less the unimportant part of the image (e.g., a dog part of the image) in image A than in image B. Similar idea was explored in [Kim+18b], except they used a surrogate metric (model's accuracy) as a source of ground truth explanations in combination with synthetic data.

Other works using similar strategies include [Wil+20b; Gha+21; PMT18; KPT21; Yeh+19b].

**Examples with Real Datasets.** While more difficult, it is possible to at least partially check for certain kinds of properties on real datasets that have no ground-truth.

For example, suppose a method returns a ranking of features that are most important to least, and we want to check its faithfulness to the underlying model. "Importance" of features here is defined as a set of minimum features triggering model's prediction. One can collect a model's correct prediction only with the top-1 most important feature (provided an interpretability method), then only the top-2 features, etc., then check if the delta in prediction accuracy between top- $n$  and top- $n + 1$  decreases as  $n$  increases. Note that this evaluation assumes that there is no significant interacting between features; if features A, B, C are the top-3 features, but C is only important if feature set B

1 exists, this would over-estimate the importance of the feature set C.  
2

3 Figure35.5 shows an example of this from [Gho+19]. The method aims to output a set of image  
4 patches (e.g., a set of connected pixels) that correlates (if not causes) prediction. They adds top-n  
5 image patches provided by an interpretability method, and observe the trend in accuracy backs  
6 up their claim. Similar experiment in reverse direction (i.e., deleting top-n most important image  
7 patches) provides additional evidence. Similar experiments are also conducted in [FV17; RSG16a].

8 Instead of using model's prediction accuracy as a proxy measure, one can define a faithfulness  
9 metric that best fits their desired property. For example, in [DG17], authors define properties  
10 in plain English first: Smallest sufficient region (smallest region of the image that alone allows a  
11 confident classification) and Smallest destroying region (smallest region of the image that when  
12 removed, prevents a confident classification), followed by careful operationalization of these properties  
13 such that they become the objective for optimization. Then, separately, a evaluation metric of  
14 saliency is defined to be " the tightest rectangular crop that contains the entire salient region and  
15 to feed that rectangular region to the classifier to directly verify whether it is able to recognise the  
16 requested class.". While the "rectangular" constraint may introduce artifacts, it is a neat trick to  
17 make evaluation possible. By checking expected behavior as described above, authors confirm that  
18 methods's behavior on the real data is aligned with the defined property compared to baselines.

19  
20 **Evaluating the Evaluations.** As we have seen so far, there are many ways to formalize a given  
21 property and many empirical tests to determine whether a property is present. Each empirical test  
22 will have different qualities. As an illustration, in [Tom+20], the authors ask whether popular saliency  
23 metrics give consistent results across literature. They tested whether different metrics for assessing  
24 the quality of a saliency-based explanations (that is, explanations that identify important pixels or  
25 regions in images) is evaluating similar properties. In other words, this work tests consistency and  
26 stability property of *metrics*. They show many metrics are statistically unreliable and inconsistent.  
27 While each metric may still have a particular use [Say+19], knowing this inconsistency between  
28 metrics helps us better understand the landscape and limitations of our evaluation approaches.  
29 Developing good evaluations for computational properties is an ongoing area of research.

30  
31 **35.4.2 User Study-based Evaluation: Does the Method Help a User Perform a**  
32 **Task?**  
33

34 User study-based evaluations measure whether an interpretable machine learning method helps a  
35 human perform some task. This task could be the ultimate end-task of interest (e.g. does a method  
36 help a doctor make better treatment decisions) or a synthetic task that mirrors contexts of interest  
37(e.g. a simplified situation with artificial diseases and symptoms). In both cases, careful and rigorous  
38 experimental design is critical to ensuring that the experiment measures what we want it to measure.  
39

40 **35.4.2.1 User Studies in Real Contexts.**  
41

42The gold standard for testing whether an explanation is useful is to test it in the intended context:  
43 Do clinicians make better decisions with a certain kind of decision support? Do programmers debug  
44 code faster with a certain kind of explanation about model errors? Do developers improve fairer  
45treatment of their customers?

46 A complete guide on how to design and conduct user studies is out of scope for this chapter; below  
47

1 we point out some very basic considerations and emphasize that understanding experimental design  
2 for user studies is essential for research in interpretable machine learning.  
3

4

#### 5 35.4.2.2 Basic elements of user studies 6

7 Performing a high-quality user study is a nuanced and non-trivial endeavor. There are many sources  
8 of bias, some obvious (e.g. learning and fatigue effects during a study) and some less obvious (e.g.  
9 participants willing to work with us are more optimistic about AI technology than those we could  
10 not recruit, different participants may have different needs for cognition).

11

12 **Interface Design.** The explanation must be presented to the user, and as noted in Section 35.3.  
13 Unlike the *intrinsic* difficulty of explaining a model (i.e., in general, complex models will necessarily  
14 be harder to explain than simpler ones), the design of the interface is an *extrinsic* source of difficulty  
15 (or ease) that can confound the experimental results. For example, it may be easier, in general, to  
16 scan a list of important features if they are ordered by importance rather than ordered alphabetically.

17 When we perform an evaluation with respect to an end-task, intrinsic and extrinsic difficulties  
18 can get conflated. Does one explanation type work better because it does a better job of explaining  
19 the complex system? Or does it work better simply because it was presented in way that was  
20 easier for people to understand and use? Especially if the goal is to test the difference between one  
21 explanation and another in the experiment, it is important that the interface for each is designed to  
22 tease out the effect from the explanations and their presentations. (Note that good presentations  
23 and visualization are an important but different object of study.) Moreover, if using the interface  
24 requires training, it is important to deliver the training in a way that is neutral in each testing  
25 conditions; in general, how the end-task and goals of the study are described and confirmed (e.g.  
26 with practice questions to test comprehension) will have a large impact on how users approach the task.

27

28 **Baselines.** Simply the presence of an explanation may change the way in which people interact  
29 with an ML system. Thus, depending on the experiment goals and setting, important baselines may  
30 include: how a human performs with no ML system; an ML system with no explanation; an ML  
31 system with a placebo explanation (an explanation that provides no information); an ML system with  
32 hand-crafted explanations (manually generated by humans who are presumably good communicators).

33

34 **Experimental Design and Hypothesis Testing.** Independent and dependent variables, hy-  
35 potheses, and inclusion and exclusion criteria must be clearly defined prior to the start of the study.  
36 For example, suppose that one hypothesizes that a particular explanation will help a developer debug  
37 an image classifier. In this case, the independent variable would be a form of assistance: various  
38 explanations, perhaps to be compared to no explanation at all. The dependent variable would be  
39 whether the developer can identify bugs. Inclusion and exclusion criteria might include a requirement  
40 that the developer has sufficient experience training image classifiers (as determined by an initial  
41 survey, or even a pre-test), demonstrates engagement (as measured by a base level of performance on  
42 some initial practice rounds), and does not have prior experience with the particular explanation  
43 types (as determined by an initial survey). Other exclusion criteria could be not using data from  
44 any participant that takes an unusually long or short time to perform the end-task (as proxies for  
45 insufficient engagement).

46 As noted in Section 35.2, there are many decisions that go into any interpretable machine learning  
47

<sup>1</sup> method, and the specific context may have many nuances. Studies of the form “Does explanation  
<sup>2</sup>  $X$  (computed via some pipeline  $Y$ ) help users in context  $Z$  compared to explanation  $X'?$ ” may not  
<sup>3</sup> provide much insight as to *why* that particular explanation is better or worse—making it harder not  
<sup>4</sup> only to iterate on a particular explanation but also to generalize to other explanations or contexts.  
<sup>5</sup> There are many, many sources of potential variation in the results, ranging from the properties of  
<sup>6</sup> the explanation and its presentation to the randomness and difficulty of the task.  
<sup>7</sup>

<sup>8</sup> To reduce this variance, and to get more useful and generalizable insights, one may want to, at  
<sup>9</sup> least in initial stages, manipulate some of these factors directly. For example, if the research question  
<sup>10</sup> is whether complete explanations are better than incomplete explanations in a particular context,  
<sup>11</sup> then one might write out, by hand, explanations that are complete in what features they implicate,  
<sup>12</sup> explanations in which one important feature is missing, and explanations in which several important  
<sup>13</sup> features are missing—being careful to choose the missing features either at random or in a specific  
<sup>14</sup> way. Doing so ensures even coverage of the different experimental regimes of interest, which may  
<sup>15</sup> not occur if the explanations were simply output from a pipeline. As another example, one might  
<sup>16</sup> intentionally create an image classifier with known bugs, or simply pretend to have an image classifier  
<sup>17</sup> that makes certain predictions, so one knows exactly what should be found and the classifier’s error  
<sup>18</sup> rate (as done in [Ade+20b]). These kinds of studies are called *wizard-of-oz* studies, and they can  
<sup>19</sup> help us more precisely uncover the science of why an explanation is useful (e.g. as done in [Jac+21]).

<sup>20</sup> Once the independent and dependent variables, hypotheses, and participant criteria (including  
<sup>21</sup> how the independent and dependent variables may be manipulated) are determined, the next step  
<sup>22</sup> is setting up the study design itself. Broadly speaking, randomization—whether it is in assigning  
<sup>23</sup> subjects to end-tasks or ordering end-tasks—marginalizes over various potential confounds (e.g.  
<sup>24</sup> subject prior knowledge or learning effects). Matching (e.g., asking the same subject to perform the  
<sup>25</sup> same or equivalent end-tasks with two different explanations) reduces variance. Repeated measures  
<sup>26</sup>(e.g., asking the subject to perform the end-task for several different inputs) also reduces variance.

<sup>27</sup> In the kinds of studies that one is likely to be performing for user studies in interpretable machine  
<sup>28</sup> learning, block randomized designs, or more generally, Latin square designs, can be used, for example,  
<sup>29</sup> to randomize the order of explanation types while keeping tasks associated with each explanation type  
<sup>30</sup> grouped together, can be used to marginalize over the effects of learning and fatigue. Careful consid-  
<sup>31</sup> eration should be given to what is tested within subjects (a form of matching that produces lower  
<sup>32</sup> variance, but adds potential bias from learning effects from the first task to the second) and across sub-  
<sup>33</sup> jects (higher variance, potential bias from population shift during experiment recruitment). Finally,  
<sup>34</sup> each of these study designs, as well as the choice of independent and dependent variables, will imply  
<sup>35</sup> an appropriate significance test (e.g., ANOVA, bonferroni correction). It is essential to choose the  
<sup>36</sup> right test and multiple hypothesis correction to avoid inflated significance values while retaining power.  
<sup>37</sup>

<sup>38</sup> **Qualitative Studies.** So far, we have described the standard approach for the design of a  
<sup>39</sup> quantitative user study—one in which the dependent variable is numerically measured (e.g. time  
<sup>40</sup> taken to correctly identify a bug, % bugs detected). While quantitative studies provide value by  
<sup>41</sup> demonstrating that there is a consistent, quantifiable effect across many users, they usually do not  
<sup>42</sup> tell us *why* a certain explanation worked. In contrast, qualitative studies, often performed with a  
<sup>43</sup> “think-aloud” or other discussion-based protocol in which users expose their thought process as they  
<sup>44</sup> perform the experiment, can help identify why a particular form of explanation seems to be useful  
<sup>45</sup>(or not), as the experimenter can gain insights by hearing how the user was using the information,  
<sup>46</sup> and depending on the protocol, can ask for clarifications.

<sup>47</sup>

1

2 For example, suppose one is interested in how people use an example-based explanation to  
3 understand a video-game agent’s policy. The idea is to show a few video clips of an automated  
4 agent in the video game, and then ask the user what the agent might do in novel situations. In a  
5 think-aloud study, the user would perform this task while talking through how they are connecting  
6 the videos they have seen to the new situation. By hearing these thoughts, a researcher might not  
7 only gain deeper insight into how users make these connections—e.g. users might see the agent  
8 collect coins in one video and presume that the agent will always go after coins (a goal directed  
9 viewpoint, vs. the agent will go left or right in the presence of a coin)—to refine the explanation, but  
10 they might also identify surprising bugs: for example, a user might see the agent fall into a pit and  
11 attribute it to a one-off sloppy fingers, not internalizing that an automated agent might make that  
12 mistake every time.

13

14 While a participant in a think-aloud study is typically more engaged in the study than the might  
15 be otherwise (because they are describing their thinking), knowing their thoughts can provide insight  
16 into the causal process between what information is being provided by the explanation and the  
17 action that the human user takes, ultimately helping advance the science of how people interact with  
18 machine-provided information.

19

20 **Pilot Studies:** The above descriptions are just a very high-level overview of the many factors  
21 that must be designed properly for a high-quality evaluation. In practice, one do not typically get all  
22 of these right the first time. Small scale pilot studies are essential to checking whether participants  
23 attend to the provided information in unexpected ways (or not at all), whether instructions are  
24 clear and well-designed, etc. Modifying the experiments after iterative small scale pilot studies can  
25 save a lot of time and energy down the road. In these pilots, one should collect not only the usual  
26 information about users and the dependent variables, but also discuss with the participants how they  
27 approached the study tasks and whether elements were confusing. These discussions will lead to  
28 insights and confidence that the study is testing what it is intended to test. The results from pilot  
29 studies should be not included in the final results.

30

31 Finally, as the number of factors to test increases (e.g., baselines, independent variables), the study  
32 design becomes more complex and may require more participants and longer participation times  
33 to determine if the results are significant—which can in turn increase costs and effects of fatigue.  
34 Pilots, think-aloud studies, and careful thinking about what aspects of the evaluation require user  
35 studies and what can be completed computationally (e.g. if one explanation has significantly worse  
36 properties than another) can all help distill down a user-based evaluation to the most important  
37 factors.

38

39

### 35.4.2.3 User Studies in Synthetic Contexts

40

41 It is not always appropriate or possible to test an interpretable machine learning method in the real  
42 context: for example, it would be unethical to test a prototype explanation system on patients each  
43 time one has a new way to convey information about a treatment recommendation. In such cases, we  
44 might want to run an experiment in which clinicians perform a task on made-up patients, or in some  
45 analogous non-medical context where the participant pool is bigger and more affordable. Similarly,  
46 one might create a relatively accessible image classification debugging context where one can control  
47 the incorrect labels, distribution shifts, etc. (e.g. [Ade+20b]) and see what and if explanations help  
users detect problems in this simpler setting. Using simpler setting could be a way to shed light on

1 what properties are important for debugging image classification more broadly. Synthetic contexts  
2 may also allow us to study more general properties of an interpretable machine learning method. For  
3 example, we may be interested in how different forms of explanation have different cognitive loads  
4 or how a particular property affects performance (e.g., [Lag+19]). The same principles we outlined  
5 above for user studies in real contexts continue to apply, but there are some important cautions.  
6

7 **Cautions regarding synthetic contexts:** While user studies with synthetic contexts can be  
8 valuable for identifying scientific principles, one must still be cautious when testing in situations that  
9 would be difficult. For example, experimental subjects in a synthetic high-stakes context may not  
10 treat the stakes of the problem as seriously, may be relatively unburdened with respect to distractions  
11 or other demands on their time and attention (e.g. a quiet study environment vs. a chaotic hospital  
12 floor), and ignore important factors of the task (e.g., a participant ignores some of the information  
13 to complete the task as quickly as possible). Moreover, small differences in task definition can have  
14 big effects: even the difference between asking users to simply perform a task with an explanation  
15 available vs. asking users to answer some questions about the explanation first, may create very  
16 different results as the latter forces the user to pay attention to the explanation and the former does  
17 not. Priming users by giving them specific scenario where they can put themselves into a mindset  
18 (e.g., imagine now you are an engineer at a company trying to sell a classifier. The deadline is  
19 approaching and your boss asked for A) could also help.

21

22

### 23 **35.5 Discussion: How to Think about Interpretable Machine Learning**

24

25 Interpretable machine learning is a young, interdisciplinary field of study. As a result, consensus  
26 on definitions, evaluation methods, and appropriate abstractions is still forming. The goal of this  
27 section is to lay out a core set of principles about interpretable machine learning. While specifics in  
28 the previous sections may change, the principles below will be more durable.

29

30 **There is no universal, mathematical definition of interpretability, and there never**  
31 **will be. Defining a downstream performance metric (and justifying it) for each context**  
32 **is a must.** The information that best communicates to the human what is needed to perform a  
33 task will necessarily vary: for example, what a clinical expert needs to convince themselves that a  
34 proposed treatment policy is worth trying on patients is very different than what a person whose  
35 loan was just denied needs to determine what they need to change to get an approval. Similarly,  
36 methods to communicate characteristics of models built on pixel data may not be appropriate for  
37 communicating characteristics of models built on tabular data. We may hope to identify desired  
38 properties in explanations to maximize downstream task performance for different classes of end  
39 tasks—that is the grand challenge of interpretable machine learning—but ultimately, there will never  
40 be one metric for all contexts.

41 While this lack of a universal metric may feel disappointing at first, this is a common in other areas  
42 of machine learning when the techniques are used in real applications. For example, there exist many  
43 metrics for fairness, and not only is it impossible to satisfy them all at the same time [KMR16], but  
44 also in a particular situation, none of them may exactly match the desires of an algorithm designer  
45 and stakeholders. Even in a standard classification setting, there are many metrics that correspond  
46 to make the predicted and true labels as close as possible. Does one care about overall accuracy?

47

Sensitivity? Specificity? It is unlikely that one objective captures everything that is needed in one situation, much less across different contexts. Evaluation can still be rigorous as long as assumptions and requirements are made precise.

What sets interpretable machine learning apart from other areas of machine learning, however, is that a large class of evaluations require human input. As a necessarily interdisciplinary area, rigorous work in interpretable machine learning requires not only knowledge how to approach computation and statistics rigorously but also how approach experimental design and user studies rigorously.

**Interpretability is only a part of the solution for fairness, calibrated trust, accountability, causality and other important problems.** Learning models that are fair, safe, causal, or engender calibrated trust are all goals, whereas interpretability is one means toward that goal.

In some cases, we don't need interpretability. For example, if the goal can be fully formalized in mathematical terms (e.g., a regulatory requirement may mandate quotas or thresholds on specific fairness metrics that a model must meet), we do not need any human input. If a model behaves safely across an exhaustive set of pre-defined inputs, then it may be less important to understand how it produced its outputs. Similarly, if a model performs well across a variety of regimes, that might (appropriately) increase one's trust in it; if it makes errors, that might (appropriately) decrease trust without an inspection of any of the system's internals.

In other cases, human input is needed to achieve the end-task. For example, while there is much excellent work in the identification of causal models, under many circumstances, it is not possible to learn a model that is *guaranteed* to be causal from a dataset alone. Here, interpretability could be means for the end-task of causality by allowing a human to inspect the model's causal mechanism.

As another example, one could measure the safety of a clinical decision support system by tracking how often following its recommendations causes harm to patients—and stop using the system if it causing too much harm. However, if we use this approach to safety, we will only discover that the system is unsafe *after* a significant number of patients have been harmed. Here, interpretability could support the end-task of safety by allowing clinical experts to understand if the model's decision process has any red flags *prior* to deployment to minimize the potential harm.

In general, complex contexts and end-tasks will require a constellation of methods (and people) to achieve them. For example, formalizing a complex notion such as accountability will require a broad collection of people—from policy makers and ethicists to corporations, engineers, and users—unifying vocabularies, exchanging domain knowledge, and identifying goals. Evaluating or monitoring it will involve various empirical measures of quality and insights from interpretability. While methods research in interpretable machine learning will often, by necessity, focus on a specific aspect of an interpretable machine learning approach, in many real settings, interpretability will be one part of responsible machine learning.

**Interpretability is distinct from full transparency into the model or knowing the model's code** Staring at the weights of every neuron in a large network is likely to be as effective as taking one's laptop apart to understand a bug in your code. There are many good reasons for open source projects and models, but open source code itself is not interpretable: A typical user will not be able to reason through 100K lines of parameters despite having all the pieces available. Interpretability is not just about the mechanism of the complex model itself, it's equally about the users who consume the explanations for their end-task.

1 **Interpretability is not about understanding everything about the model; it is about**  
2 **understanding enough to do the end-task.** The ultimate measure of an interpretable machine  
3 learning method is whether it helps the user perform their end-task well. Suppose the end-task is  
4 to fix an overheating laptop. If one knows the likely sources of the heat, the laptop is probably  
5 interpretable enough to be fixed, even if one does not know the chemical properties of its components.  
6 On the other hand, if the laptop keeps freezing up, knowing about the sources of heat may not be the  
7 right information. Importantly, both end-tasks are have clear downstream performance metrics: we  
8 can observe whether the information helped the user perform actions that make the laptop overheat  
9 or freeze up less.

10  
11 As another example, consider AlphaGo, Google DeepMind’s AI go player that beat the human  
12 world champion, Lee SeDol. The model is so complex that one cannot fully understand its decision  
13 process, including surprising moves like its famous move 37[Met16]. That said, partial probes (e.g.,  
14 does AlphaGo believe the same move would have made a different impact if it was made earlier but  
15 similar position in the game) might still help a Go expert gain insights on the rationale for the move  
16 in the context of what they already know about the game.

17  
18 **However, any partial view of a model is, necessarily, only a partial view; it does not**  
19 **tell the full story.** While we just argued that many end-tasks do not require knowing everything  
20 about a model, we also need to understand the flip-side of that statement for the sake of caution;  
21 if the human user does not understand everything about the model, then that implies there are  
22 (potentially important) parts of the model that the user is not aware of. A set of features to change  
23 in order to flip a loan decision is a partial view of a model, and it might be what a user needs to  
24 know prior to resubmitting an application. However, one partial view might not provide enough  
25 information to vet if the model is discriminatory. Specifically, any probe will only return what it is  
26 designed to compute (e.g., an approximation of a complex function with a simpler one). Different  
27 probes may be able to reveal different properties at different levels of quality. Incorrectly believing  
28 the partial view is the full story could result in incorrect insights.

29  
30 **Partial Views can Lack Stability and Enable Attacks.** Relatedly, any explanation that  
31 reveals only certain parts of a model can lack stability (e.g. see [AMJ18a]) and can be more  
32 easily attacked (e.g. see [Yeh+19a; GAZ19; Dom+19; Sla+20]). Especially when models are  
33 overparameterized such as neural networks, it is possible to learn models whose explanations say  
34 one thing (e.g. a feature is not important, according to some formalization of feature importance)  
35 while the model does another (e.g. uses the prohibited feature). Joint training only exacerbates the  
36 issue, as it allows the model to learn boundaries that pass some partial-view test while in reality  
37 violating the underlying constraint. Other adversarial approaches can work on the input, minimally  
38 perturbing it to change explanation while keeping the prediction constant or to change the prediction  
39 while keeping the explanation constant (again, for an explanation that is a specific partial view).

40 These concerns highlight an important open area: We need to improve ways to endow explanations  
41 with the property of translucence, that is, explanations that communicate what they can and cannot  
42 say about the model. Translucence is important because misinterpreted explanations that happen to  
43 favor a user’s views create false basis for trust.

44  
45 **Trade-offs between Inherently Interpretable Models and Performance often do not**  
46 **exist; Partial Views can help when they do.**

47

1 While some have claimed that there exists a trade-off between using an inherently-interpretable  
2 model and performance (defined as a model’s performance on some test data), this trade-off rarely  
3 exists in practice for several reasons[Rud19].  
4

5 First, in many cases, the data can be surprisingly well-fit by a fairly simple model (due to high  
6 noise, for example) or a model that can be decomposed into interpretable parts. One can often find a  
7 framing—an architecture, a regularizer, an optimizer—that produces inherently interpretable models  
8 with performance comparable to, or sometimes even better than, blackbox approaches [Wan+17a;  
9 LCG12; Car+15; Let+15b; UR16; FHDV20; KRS14]. In fact, interpretability and performance can  
10 be synergistic: inherently interpretable models often encode a preference for simpler models (consider  
11 an L1 regularizer that can enforce sparsity but initially developed to increase performance), which  
12 can result in models that are also often more robust [RDV18].  
13

14 Second, a narrow focus on the trade-off between using inherently interpretable models and a  
15 predefined metric of performance, as usually measured on a validation set, overlooks a broader issue:  
16 that predefined metric of performance may not tell the full story about the quality of the model. For  
17 example, using an inherently interpretable model may enable a person realize that a prediction is  
18 based on confounding, not causation—or other ways it might fail in deployment. In this way, one  
19 might get better performance with an inherently interpretable model in practice even if a blackbox  
20 appears to have better performance numbers in validation. An inherently interpretable model may  
21 also enable better human+model teaming by allowing the human user to step in and override the  
22 system appropriately.  
23

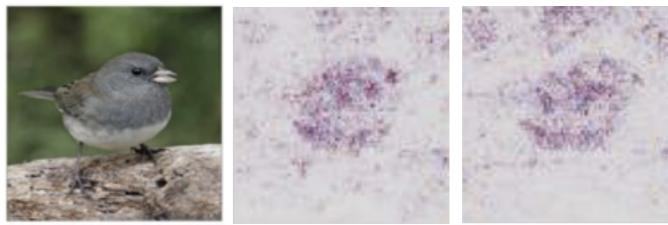
24 **Human factors are essential.** All machine learning systems ultimately connect to broader socio-  
25 technical contexts. However, in many cases, many aspects of model construction and optimization can  
26 be performed in a purely computational setting: there are techniques to check for appropriate model  
27 capacity, techniques for tuning a gradient descent or convex optimization. In contrast, interpretable  
28 machine learning must consider human factors **from the beginning**: there is no point optimizing  
29 an explanation to have various properties if it still fails to improve the user’s performance on the  
end-task—or even decreases their performance.

30 *Over-reliance.* Just because an explanation is present, does not mean that the user will analytically  
31 and reasonably incorporate the information provided into their ultimate decision-making task. Indeed,  
32 the presence of *any* explanation can increase a user’s trust in the model (it even has a justification  
33 for its recommendation!), exacerbating the general issue of over-trust in human+ML teams. Recent  
34 studies have found that even data scientists often misuse interpretability tools because they do not  
35 interpret the explanations in the way the tool intended, resulting inappropriately increased confidence.  
36 The authors note that the participants’ excitement about the tool led them to take it at face-value  
37 rather than dig deeper. [LM20] reports a similar finding, noting that evocative presentations can  
38 create a feeling of comprehension even if the information is not being presented accurately.  
39

40 Over-reliance can be combated with explicit measures to force the user to engage analytically and  
41 skeptically with the information in the explanation. For example, one could ask the user to submit  
42 their decision first and only then show the recommendation and accompanying explanation to pique  
43 their interest in why their choice and the recommendation might disagree (and prompting whether  
44 they want to change their choice). Another option might be to ask the user some basic question  
45 about the explanation prior to submitting their decision to force them to look at the explanation  
46 carefully. Yet another might be to provide only the relevant information (the explanation) without  
47 the recommendation, forcing the user to combine the additional information with their own. However,

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

## Original Image



11 *Figure 35.6: (Potential) perception issues: an explanation from a trained network (left) is visually indistin-*  
12 *guishable to humans from one from an untrained network (right)—even if they are not exactly identical.*

13  
14  
15

16 in all these cases, there is a delicate balance: users will often be amenable to expending additional  
17 cognitive effort if they can see it achieves better results, but if they feel the effort is too much, they  
18 may start ignoring the model entirely.

19

20 *Potential for Misuse.* A worse version of over-reliance is when explanations are used to manipulate  
21 a user—such as increase trust—rather than facilitating an end-task of the user’s interest—such as  
22 calibrating their trust. Furthermore, users may report that they *like* explanations are simple, require  
23 little cognitive effort, etc. even when those explanations do not help them perform their end-task.  
24 As creators of interpretable machine learning methods, one must be on alert to ensure that the  
25 explanations help the user achieve what they want to (ideally in a way that they also like).

26

27 *Misunderstandings from a lack of understanding of machine learning.* Even when correctly engaged,  
28 users in different contexts will have different levels of knowledge about machine learning. For example,  
29 not everyone may understand what it means for factors to be additive or the implications of feature  
30 importances selected based on Shapely values [Sha16]. Users may also attribute more understanding  
31 to a model than it actually has. For example, if they see a set of pixels highlighted around a beak, or  
32 a set of topic model terms about a disease, they may mistakenly believe that the machine learning  
33 model has some notion of concepts that matches theirs, when the truth might be quite different.

34

35 *Perception issues.* In some cases, the explanation may not be truly understandable to the human.  
36 For example, due to the inherent biases of our visual perception, humans may not perceive the  
37 difference between different explanatory images. In Figure 35.6, two explanations (in terms of  
38 important pixels in a bird image) seem to communicate similar message; for most people, both  
39 explanations seem to suggest that the belly and cheek of the bird are the important parts for this  
40 prediction. However, one of them is generated from a trained network (left), but the other one is  
41 from a network that returns random predictions (right). While the two saliency maps aren’t identical  
42 to machines, they look similar because humans don’t parse an image as pixel values, but as whole:  
43 they see a bird in both pictures.

44 Another common issue with pixel-based explanations is that explanation creators often multiply  
45 the original image with an importance “mask” (black and clear saliency mask, where black pixel  
46 represents no importance and a clear pixel represents maximum importance), introducing the arbi-

47

1 trary artifact that black objects never appear important. In addition, this binary mask is produced  
2 by clipping important pixels in certain percentile (e.g., only taking 99-th percentile), which can  
3 also introduce another artifact [Sun+19c]. The balancing act between artifacts introduced by visu-  
4 alization for the ease of understanding and faithfully representing the explanation remains a challenge.  
5

6 Together, all of these points on human factors emphasize what we said from the start: we cannot  
7 divorce the study and practice of interpretable machine learning from its intended socio-technical  
8 context.  
9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47



PART VI

## Decision making



# 36 Multi-step decision problems

## 36.1 Introduction

We introduced the basics of **Bayesian decision theory** in Section 3.8. In this chapter, we consider applications and extensions of this to scenarios in which the agent must make a sequence of decisions.

## 36.2 Decision (influence) diagrams

When dealing with structured multi-stage decision problems, it is useful to use a graphical notation called an **influence diagram** [HM81; KM08], also called a **decision diagram**. This extends directed probabilistic graphical models (Chapter 4) by adding **decision nodes** (also called **action nodes**), represented by rectangles, and **utility nodes** (also called **value nodes**), represented by diamonds. The original random variables are called **chance nodes**, and are represented by ovals, as usual.

### 36.2.1 Example: oil wildcatter

As an example (from [Rai68]), consider creating a model for the decision problem faced by an oil “**wildcatter**”, which is a person who drills wildcat wells, which are exploration wells drilled in areas not known to be oil fields.

Suppose you have to decide whether to drill an oil well or not at a given location. You have two possible actions:  $d = 1$  means drill,  $d = 0$  means don’t drill. You assume there are 3 states of nature:  $o = 0$  means the well is dry,  $o = 1$  means it is wet (has some oil), and  $o = 2$  means it is soaking (has a lot of oil). We can represent this as a decision diagram as shown in Figure 36.1(a).

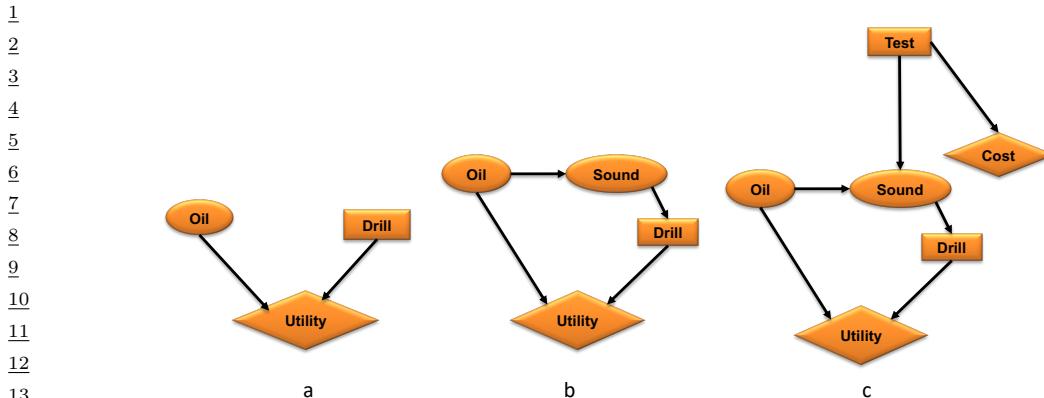
Suppose your prior beliefs are  $p(o) = [0.5, 0.3, 0.2]$ , and your utility function  $U(d, o)$  is specified by the following table:

	$o = 0$	$o = 1$	$o = 2$
$d = 0$	0	0	0
$d = 1$	-70	50	200

We see that if you don’t drill, you incur no costs, but also make no money. If you drill a dry well, you lose \$70; if you drill a wet well, you gain \$50; and if you drill a soaking well, you gain \$200.

What action should you take if you have no information beyond your prior knowledge? Your prior expected utility for taking action  $d$  is

$$\text{EU}(d) = \sum_{o=0}^2 p(o)U(d, o) \quad (36.1)$$



14  
15  
16  
17  
18  
19

Figure 36.1: Influence diagrams for the oil wild catter problem. Ovals are random variables (chance nodes), squares are decision (action) nodes, diamonds are utility (value) nodes. (a) Basic model. (b) An extension in which we have an information arc from the Sound chance node to the Drill decision node. (c) An extension in which we get to decide whether to perform a test or not, as well as whether to drill or not.

20 We find  $\text{EU}(d = 0) = 0$  and  $\text{EU}(d = 1) = 20$  and hence the maximum expected utility is

21

$$22 \quad \text{MEU} = \max\{\text{EU}(d = 0), \text{EU}(d = 1)\} = \max\{0, 20\} = 20 \quad (36.2)$$

23 Thus the optimal action is to drill,  $d^* = 1$ .

25

### 26 36.2.2 Information arcs

27 Now let us consider a slight extension to the model, in which you have access to a measurement  
28 (called a “sounding”), which is a noisy indicator about the state of the oil well. Hence we add an  
29  $O \rightarrow S$  arc to the model. In addition, we assume that the outcome of the sounding test will be  
30 available before we decide whether to drill or not; hence we add an **information arc** from  $S$  to  $D$ .  
31 This is illustrated in Figure 36.1(b). Note that the utility depends on the action and the true state  
32 of the world, but not the measurement.  
33

We assume the sounding variable can be in one of 3 states:  $s = 0$  is a diffuse reflection pattern,  
34 suggesting no oil;  $s = 1$  is an open reflection pattern, suggesting some oil; and  $s = 2$  is a closed  
35 reflection pattern, indicating lots of oil. Since  $S$  is caused by  $O$ , we add an  $O \rightarrow S$  arc to our model.  
36 Let us model the reliability of our sensor using the following conditional distribution for  $p(S|O)$ :  
37

	$s = 0$	$s = 1$	$s = 2$
$o = 0$	0.6	0.3	0.1
$o = 1$	0.3	0.4	0.3
$o = 2$	0.1	0.4	0.5

43 Suppose the sounding observation is  $s$ . The posterior expected utility of performing action  $d$  is

44

$$45 \quad \text{EU}(d|s) = \sum_{o=0}^2 p(o|s)U(o, d) \quad (36.3)$$

46

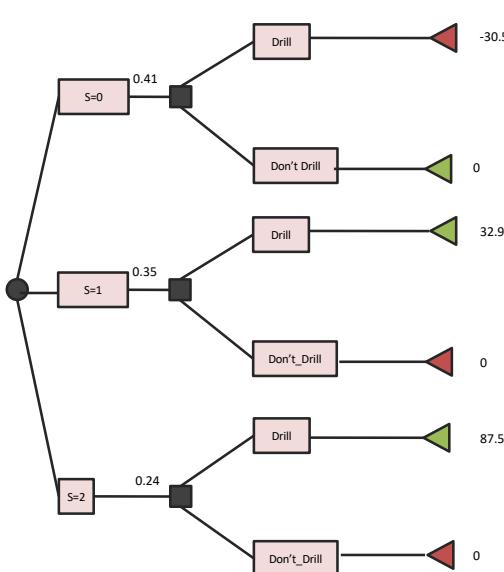


Figure 36.2: Decision tree for the oil wildcatter problem. Black circles are chance variables, black squares are decision nodes, diamonds are the resulting utilities. Green leaf nodes have higher utility than red leaf nodes.

We need to compute this for each possible observation,  $s \in \{0, 1, 2\}$ , and each possible action,  $d \in \{0, 1\}$ . If  $s = 0$ , we find the posterior over the oil state is  $p(o|s = 0) = [0.732, 0.219, 0.049]$ , and hence  $\text{EU}(d = 0|s = 0) = 0$  and  $\text{EU}(d = 1|s = 0) = -30.5$ . If  $s = 1$ , we similarly find  $\text{EU}(d = 0|s = 1) = 0$  and  $\text{EU}(d = 1|s = 1) = 32.9$ . If  $s = 2$ , we find  $\text{EU}(d = 0|s = 2) = 0$  and  $\text{EU}(d = 1|s = 2) = 87.5$ . Hence the optimal policy  $d^*(s)$  is as follows: if  $s = 0$ , choose  $d = 0$  and get \$0; if  $s = 1$ , choose  $d = 1$  and get \$32.9; and if  $s = 2$ , choose  $d = 1$  and get \$87.5.

The maximum expected utility of the wildcatter, before seeing the experimental sounding, can be computed using

$$\text{MEU} = \sum_s p(s) \text{EU}(d^*(s)|s) \quad (36.4)$$

where prior marginal on the outcome of the test is  $p(s) = \sum_o p(o)p(s|o) = [0.41, 0.35, 0.24]$ . Hence the MEU is

$$\text{MEU} = 0.41 \times 0 + 0.35 \times 32.9 + 0.24 \times 87.5 = 32.2 \quad (36.5)$$

These numbers can be summarized in the **decision tree** shown in Figure 36.2.

### 36.2.3 Value of information

Now suppose you can choose whether to do the test or not. This can be modelled as shown in Figure 36.1(c), where we add a new test node  $T$ . If  $T = 1$ , we do the test, and  $S$  can enter states

<sup>1</sup>  $\{0, 1, 2\}$ , determined by  $O$ , exactly as above. If  $T = 0$ , we don't do the test, and  $S$  enters a special  
<sup>2</sup> unknown state. There is also some cost associated with performing the test.

<sup>4</sup> Is it worth doing the test? This depends on how much our MEU changes if we know the outcome  
<sup>5</sup> of the test (namely the state of  $S$ ). If you don't do the test, we have  $MEU = 20$  from Equation (36.2).  
<sup>6</sup> If you do the test, you have  $MEU = 32.2$  from Equation (36.5). So the improvement in utility if  
<sup>7</sup> you do the test (and act optimally on its outcome) is \$12.2. This is called the **value of perfect**  
<sup>8</sup> **information** (VPI). So we should do the test as long as it costs less than \$12.2.

<sup>9</sup> In terms of graphical models, the VPI of a variable  $S$  can be determined by computing the MEU  
<sup>10</sup> for the base influence diagram,  $\mathcal{G}$ , in Figure 36.1(b), and then computing the MEU for the same  
<sup>11</sup> influence diagram where we add information arcs from  $S$  to the action node, and then computing the  
<sup>12</sup> difference. In other words,

$$\frac{13}{14} \quad VPI = MEU(\mathcal{G} + S \rightarrow D) - MEU(\mathcal{G}) \quad (36.6)$$

<sup>15</sup> where  $D$  is the decision node and  $S$  is the variable we are measuring. This will tell us whether it is  
<sup>16</sup> worth adding obtaining measurement  $S$ .  
<sup>17</sup>

#### <sup>18</sup> 36.2.4 Computing the optimal policy

<sup>20</sup> In general, given an influence diagram, we can compute the optimal policy automatically by modifying  
<sup>21</sup> the variable elimination algorithm (Section 9.4), as explained in [LN01; KM08]. The basic idea is to  
<sup>22</sup> work backwards from the final action, computing the optimal decision at each step, assuming all  
<sup>23</sup> following actions are chosen optimally. When the influence diagram has a simple chain structure,  
<sup>24</sup> as in a Markov decision process (Section 36.5), the result is equivalent to Bellman's equation  
<sup>25</sup> (Section 36.5.5).  
<sup>26</sup>

<sup>27</sup>

### <sup>28</sup> 36.3 A/B testing

<sup>29</sup>

<sup>30</sup> Suppose you are trying to decide which version of a product is likely to sell more, or which version of  
<sup>31</sup> a drug is likely to work better. Let us call the versions you are choosing between A and B; sometimes  
<sup>32</sup> version A is called the **control**, and version B is called the **treatment**. (Sometimes the different  
<sup>33</sup> actions are called "**arms**".)

<sup>34</sup> A very common approach to such problems is to use an **A/B test**, in which you try both actions  
<sup>35</sup> out for a while, by randomly assigning a different action to different subsets of the population, and  
<sup>36</sup> then you measure the resulting accumulated **reward** from each action, and you pick the winner.  
<sup>37</sup> (This is sometimes called a "**test and roll**" approach, since you test which method is best, and then  
<sup>38</sup> roll it out for the rest of the population.)

<sup>39</sup> A key problem in A/B testing is to come up with a decision rule, or policy, for deciding which  
<sup>40</sup> action is best, after obtaining potentially noisy results during the test phase. Another problem is  
<sup>41</sup> to choose how many people to assign to the treatment,  $n_1$ , and how many to the control,  $n_0$ . The  
<sup>42</sup> fundamental tradeoff is that using larger values of  $n_1$  and  $n_0$  will help you collect more data and  
<sup>43</sup> hence be more confident in picking the best action, but this incurs an **opportunity cost**, because  
<sup>44</sup> the testing phase involves performing actions that may not result in the highest reward. (This is  
<sup>45</sup> an example of the exploration-exploitation tradeoff, which we discuss more in Section 36.4.3.) In  
<sup>46</sup> this section, we give a simple Bayesian decision theoretic analysis of this problem, following the  
<sup>47</sup>

<sup>1</sup> presentation of [FB19].<sup>1</sup> More details on A/B testing can be found in [KTX20].

### <sup>4</sup> 36.3.1 A Bayesian approach

<sup>5</sup> We assume the  $i$ 'th reward for action  $j$  is given by  $Y_{ij} \sim \mathcal{N}(\mu_j, \sigma_j^2)$  for  $i = 1 : n_j$  and  $j = 0 : 1$ , where  
<sup>6</sup>  $j = 0$  corresponds to the control (action A),  $j = 1$  corresponds to the treatment (action B), and  $n_j$  is  
<sup>7</sup> the number of samples you collect from group  $j$ . The parameters  $\mu_j$  are the expected reward for  
<sup>8</sup> action  $j$ ; our goal is to estimate these parameters. (For simplicity, we assume the  $\sigma_j^2$  are known.)  
<sup>9</sup>

<sup>10</sup> We will adopt a Bayesian approach, which is well suited to sequential decision problems. For  
<sup>11</sup> simplicity, we will use Gaussian priors for the unknowns,  $\mu_j \sim \mathcal{N}(m_j, \tau_j^2)$ , where  $m_j$  is the prior  
<sup>12</sup> mean reward for action  $j$ , and  $\tau_j$  is our confidence in this prior. We assume the prior parameters are  
<sup>13</sup> known. (In practice we can use an empirical Bayes approach, as we discuss in Section 36.3.2.)

#### <sup>14</sup> 36.3.1.1 Optimal policy

<sup>16</sup> Initially we assume the sample size of the experiment (i.e. the values  $n_1$  for the treatment and  $n_0$  for  
<sup>17</sup> the control) are known. Our goal is to compute the optimal policy or decision rule  $\pi(\mathbf{y}_1, \mathbf{y}_0)$ , which  
<sup>18</sup> specifies which action to deploy, where  $\mathbf{y}_j = (y_{1j}, \dots, y_{n_j, j})$  is the data from action  $j$ .

<sup>19</sup> The optimal policy is simple: choose the action with the greater expected posterior expected  
<sup>20</sup> reward:

$$\pi^*(\mathbf{y}_1, \mathbf{y}_0) = \begin{cases} 1 & \text{if } \mathbb{E}[\mu_1 | \mathbf{y}_1] \geq \mathbb{E}[\mu_0 | \mathbf{y}_0] \\ 0 & \text{if } \mathbb{E}[\mu_1 | \mathbf{y}_1] < \mathbb{E}[\mu_0 | \mathbf{y}_0] \end{cases} \quad (36.7)$$

<sup>25</sup> All that remains is to compute the posterior over the unknown parameters,  $\mu_j$ . Applying Bayes'  
<sup>26</sup> rule for Gaussians (Equation (2.59)), we find that the corresponding posterior is given by  
<sup>27</sup>

$$p(\mu_j | \mathbf{y}_j, n_j) = \mathcal{N}(\mu_j | \hat{m}_j, \hat{\tau}_j^2) \quad (36.8)$$

$$1/\hat{\tau}_j = n_j/\sigma_j^2 + 1/\tau_j^2 \quad (36.9)$$

$$\hat{m}_j / \hat{\tau}_j = n_j \bar{y}_j / \sigma_j^2 + m_j / \tau_j^2 \quad (36.10)$$

<sup>32</sup> We see that the posterior precision (inverse variance) is a weighted sum of the prior precision plus  $n_j$   
<sup>33</sup> units of measurement precision. We also see that the posterior precision weighted mean is a sum of  
<sup>34</sup> the prior precision weighted mean and the measurement precision weighted mean.  
<sup>35</sup>

<sup>36</sup> Given the posterior, we can plug  $\hat{m}_j$  into Equation (36.7). In the fully symmetric case, where  
<sup>37</sup>  $n_1 = n_0$ ,  $m_1 = m_0 = m$ ,  $\tau_1 = \tau_0 = \tau$ , and  $\sigma_1 = \sigma_0 = \sigma$ , we find that the optimal policy is to simply  
<sup>38</sup> "pick the winner", which is the arm with higher empirical performance:

$$\pi^*(\mathbf{y}_1, \mathbf{y}_0) = \mathbb{I}\left(\frac{m}{\tau^2} + \frac{\bar{y}_1}{\sigma^2} > \frac{m}{\tau^2} + \frac{\bar{y}_0}{\sigma^2}\right) = \mathbb{I}(\bar{y}_1 > \bar{y}_0) \quad (36.11)$$

<sup>42</sup> However, when the problem is asymmetric, we need to take into account the different sample sizes  
<sup>43</sup> and/or different prior beliefs.

<sup>45</sup> 1. For a similar set of results in the time-discounted setting, see <https://chris-said.io/2020/01/10/optimizing-sample-sizes-in-ab-testing-part-I>.

1  
2 **36.3.1.2 Optimal sample size**

3 We now discuss how to compute the optimal sample size for each arm of the experiment, i.e., the  
4 values  $n_0$  and  $n_1$ . We assume the total population size is  $N$ , and we cannot reuse people from the  
5 testing phase,

6 The prior expected reward in the testing phase is given by

7

$$\mathbb{E}[R_{\text{test}}] = n_0 m_0 + n_1 m_1 \quad (36.12)$$

8 The expected reward in the roll phase depends on the decision rule  $\pi(\mathbf{y}_1, \mathbf{y}_0)$  that we use:

9

$$\mathbb{E}_{\pi}[R_{\text{roll}}] = \int_{\mu_1} \int_{\mu_0} \int_{\mathbf{y}_1} \int_{\mathbf{y}_0} (N - n_1 - n_0) (\pi(\mathbf{y}_1, \mathbf{y}_0) \mu_1 + (1 - \pi(\mathbf{y}_1, \mathbf{y}_0)) \mu_0) \quad (36.13)$$

10

$$\times p(\mathbf{y}_0 | \mu_0) p(\mathbf{y}_1 | \mu_1) p(\mu_0) p(\mu_1) d\mathbf{y}_0 d\mathbf{y}_1 d\mu_0 d\mu_1 \quad (36.14)$$

11 For  $\pi = \pi^*$  one can show that this equals

12

$$\mathbb{E}[R_{\text{roll}}] \triangleq \mathbb{E}_{\pi^*}[R_{\text{roll}}] = (N - n_1 - n_0) \left( m_1 + e \Phi\left(\frac{e}{v}\right) + v \phi\left(\frac{e}{v}\right) \right) \quad (36.15)$$

13 where  $\phi$  is the Gaussian pdf,  $\Phi$  is the Gaussian cdf,  $e = m_0 - m_1$  and

14

$$v = \sqrt{\frac{\tau_1^4}{\tau_1^2 + \sigma_1^2/n_1} + \frac{\tau_0^4}{\tau_0^2 + \sigma_0^2/n_0}} \quad (36.16)$$

15 In the fully symmetric case, Equation (36.15) simplifies to

16

$$\mathbb{E}[R_{\text{roll}}] = \underbrace{(N - 2n)m}_{R_a} + \underbrace{(N - 2n) \frac{\sqrt{2}\tau^2}{\sqrt{\pi} \sqrt{2\tau^2 + \frac{2}{n}\sigma^2}}}_{R_b} \quad (36.17)$$

17 This has an intuitive interpretation. The first term,  $R_a$ , is the prior reward we expect to get before  
18 we learn anything about the arms. The second term,  $R_b$ , is the reward we expect to see by virtue of  
19 picking the optimal action to deploy.

20 Let us we write  $R_b = (N - 2n)R_i$ , where  $R_i$  is the incremental gain. We see that the incremental  
21 gain increases with  $n$ , because we are more likely to pick the correct action with a larger sample  
22 size; however, this gain can only be accrued for a smaller number of people, as shown by the  $N - 2n$   
23 prefactor. (This is a consequence of the explore-exploit tradeoff.)

24 The total expected reward is given by adding Equation (36.12) and Equation (36.17):

25

$$\mathbb{E}[R] = \mathbb{E}[R_{\text{test}}] + \mathbb{E}[R_{\text{roll}}] = Nm + (N - 2n) \left( \frac{\sqrt{2}\tau^2}{\sqrt{\pi} \sqrt{2\tau^2 + \frac{2}{n}\sigma^2}} \right) \quad (36.18)$$

26 (The equation for the non-symmetric case is given in [FB19].)

27

We can maximize the expected reward in Equation (36.18) to find the optimal sample size for the testing phase, which (from symmetry) satisfies  $n_1^* = n_2^* = n^*$ , and from  $\frac{d}{dn^*} \mathbb{E}[R] = 0$  satisfies

$$n^* = \sqrt{\frac{N}{4}u^2 + \left(\frac{3}{4}u^2\right)^2} - \frac{3}{4}u^2 \leq \sqrt{N}\frac{\sigma}{2\tau} \quad (36.19)$$

where  $u^2 = \frac{\sigma^2}{\tau^2}$ . Thus we see that the optimal sample size  $n^*$  increases as the observation noise  $\sigma$  increases, since we need to collect more data to be confident of the right decision. However, the optimal sample size decreases with  $\tau$ , since a prior belief that the effect size  $\delta = \mu_1 - \mu_0$  will be large implies we expect to need less data to reach a confident conclusion.

### 36.3.1.3 Regret

Given a policy, it is natural to wonder how good it is. We define the **regret** of a policy to be the difference between the expected reward given **perfect information** (PI) about the true best action and the expected reward due to our policy. Minimizing regret is equivalent to making the expected reward of our policy equal to the best possible reward (which may be high or low, depending on the problem).

An oracle with perfect information about which  $\mu_j$  is bigger would pick the highest scoring action, and hence get an expected reward of  $N\mathbb{E}[\max(\mu_1, \mu_2)]$ . Since we assume  $\mu_j \sim \mathcal{N}(m, \tau^2)$ , we have

$$\mathbb{E}[R|PI] = N \left( m + \frac{\tau}{\sqrt{\pi}} \right) \quad (36.20)$$

Therefore the regret from the optimal policy is given by

$$\mathbb{E}[R|PI] - (\mathbb{E}[R_{\text{test}}|\pi^*] + \mathbb{E}[R_{\text{roll}}|\pi^*]) = N \frac{\tau}{\sqrt{\pi}} \left( 1 - \frac{\tau}{\sqrt{\tau^2 + \frac{\sigma^2}{n^*}}} \right) + \frac{2n^*\tau^2}{\sqrt{\pi}\sqrt{\tau^2 + \frac{\sigma^2}{n^*}}} \quad (36.21)$$

One can show that the regret is  $O(\sqrt{N})$ , which is optimal for this problem when using a time horizon (population size) of  $N$  [AG13].

### 36.3.1.4 Expected error rate

Sometimes the goal is posed as **best arm identification**, which means identifying whether  $\mu_1 > \mu_0$  or not. That is, if we define  $\delta = \mu_1 - \mu_0$ , we want to know if  $\delta > 0$  or  $\delta < 0$ . This is naturally phrased as a **hypothesis test**. However, this is arguably the wrong objective, since it is usually not worth spending money on collecting a large sample size to be confident that  $\delta > 0$  (say) if the magnitude of  $\delta$  is small. Instead, it makes more sense to optimize total expected reward, using the method in Section 36.3.1.1.

Nevertheless, we may want to know the probability that we have picked the wrong arm if we use the policy from Section 36.3.1.1. In the symmetric case, this is given by the following:

$$\Pr(\pi(\mathbf{y}_1, \mathbf{y}_0) = 1 | \mu_1 < \mu_0) = \Pr(Y_1 - Y_0 > 0 | \mu_1 < \mu_0) = 1 - \Phi \left( \frac{\mu_1 - \mu_0}{\sigma \sqrt{\frac{1}{n_1} + \frac{1}{n_0}}} \right) \quad (36.22)$$

1 The above expression assumed that  $\mu_j$  are known. Since they are not known, we can com-  
2 pute the expected error rate using  $\mathbb{E}[\Pr(\pi(\mathbf{y}_1, \mathbf{y}_0) = 1 | \mu_1 < \mu_0)]$ . By symmetry, the quantity  
3  $\mathbb{E}[\Pr(\pi(\mathbf{y}_1, \mathbf{y}_0) = 0 | \mu_1 > \mu_0)]$  is the same. One can show that both quantities are given by  
4

5

$$\text{Prob. error} = \frac{1}{4} - \frac{1}{2\pi} \arctan \left( \frac{\sqrt{2}\tau}{\sigma} \sqrt{\frac{n_1 n_0}{n_1 + n_0}} \right) \quad (36.23)$$

6

7 As expected, the error rate decreases with the sample size  $n_1$  and  $n_0$ , increases with observation noise  
8  $\sigma$ , and decreases with variance of the effect size  $\tau$ . Thus a policy that minimizes the classification  
9 error will also maximize expected reward, but it may pick an overly large sample size, since it does  
10 not take into account the magnitude of  $\delta$ .

11

12

### 13 36.3.2 Example

14 In this section, we give a simple example of the above framework. Suppose our goal is to do **website**  
15 **testing**, where have two different versions of a webpage that we want to compare in terms of their  
16 **click through rate**. The observed data is now binary,  $y_{ij} \sim \text{Ber}(\mu_j)$ , so it is natural to use a Beta  
17 prior,  $\mu_j \sim \text{Beta}(\alpha, \beta)$  (see Section 3.2.1). However, in this case the optimal sample size and decision  
18 rule is harder to compute (see [FB19; Sta+17] for details). As a simple approximation, we can assume  
19  $\bar{y}_{ij} \sim \mathcal{N}(\mu_j, \sigma^2)$ , where  $\mu_j \sim \mathcal{N}(m, \tau^2)$ ,  $m = \frac{\alpha}{\alpha+\beta}$ ,  $\tau^2 = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$ , and  $\sigma^2 = m(1-m)$ .

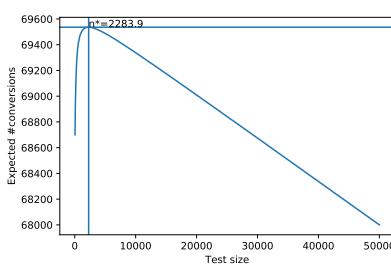
20 To set the Gaussian prior, [FB19] used empirical data from about 2000 prior A/B tests. For each  
21 test, they observed the number of times the page was served with each of the two variations, as well  
22 as the total number of times a user clicked on each version. Given this data, they used a hierarchical  
23 Bayesian model to infer  $\mu_j \sim \mathcal{N}(m=0.68, \tau=0.03)$ . This prior implies that the expected effect size  
24 is quite small,  $\mathbb{E}[|\mu_1 - \mu_0|] = 0.023$ . (This is consistent with the results in [Aze+20], who found that  
25 most changes made to the Microsoft Bing EXP platform had negligible effect, although there were  
26 occasionally some “big hits”.)

27 With this prior, and assuming a population of  $N = 100,000$ , Equation (36.19) says that the optimal  
28 number of trials to run is  $n_1^* = n_0^* = 2284$ . The expected reward (number of clicks or **conversions**)  
29 in the testing phase is  $\mathbb{E}[R_{\text{test}}] = 3106$ , and in the deployment phase  $\mathbb{E}[R_{\text{roll}}] = 66430$ , for a total  
30 reward of 69536. The expected error rate is 10%.

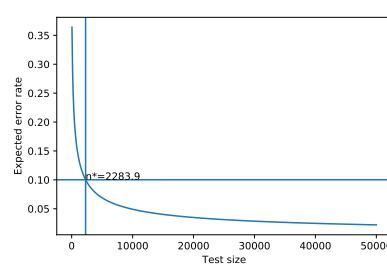
31 In Figure 36.3a, we plot the expected reward vs the size of the test phase  $n$ . We see that the  
32 reward increases sharply with  $n$  to the global maximum at  $n^* = 2284$ , and then drops off more slowly.  
33 This indicates that it is better to have a slightly larger test than one that is too small by the same  
34 amount. (However, when using a heavy tailed model, [Aze+20] finds that it is better to do lots of  
35 smaller tests.)

36 In Figure 36.3b, we plot the probability of picking the wrong action vs  $n$ . We see that tests that  
37 are larger than optimal only reduce this error rate marginally. Consequently, if you want to make  
38 the misclassification rate low, you may need a large sample size, particularly if  $\mu_1 - \mu_0$  is small, since  
39 then it will be hard to detect the true best action. However, it is also less important to identify  
40 the best action in this case, since both actions have very similar expected reward. This explains  
41 why classical methods for A/B testing based on frequentist statistics, which use hypothesis testing  
42 methods to determine if A is better than B, may often recommend sample sizes that are much larger  
43 than necessary. (See [FB19] and references therein for further discussion.)

44



(a)



(b)

Figure 36.3: Total expected profit (a) and error rate (b) as a function of the sample size used for website testing. Generated by `ab_test_demo.py`.

## 36.4 Contextual bandits

This section was co-authored with Lihong Li.

In Section 36.3, we discussed A/B testing, in which the decision maker tries two different actions,  $a_0$  and  $a_1$ , a fixed number of times,  $n_1$  and  $n_0$ , measures the resulting sequence of rewards,  $\mathbf{y}_1$  and  $\mathbf{y}_0$ , and then picks the best action to use for the rest of time (or the rest of the population) so as to maximize expected reward.

We can obviously generalize this beyond two actions. More importantly, we can generalize this beyond a one-stage decision problem. In particular, suppose we allow the decision maker to try an action  $a_t$ , observe the reward  $r_t$ , and then decide what to do at time step  $t + 1$ , rather than waiting until  $n_1 + n_0$  experiments are finished. This immediate feedback allows for **adaptive policies** that can result in much higher expected reward (lower regret). We have converted a one-stage decision problem into a **sequential decision problem**. There are many kinds of sequential decision problems, but in this section, we consider the simplest kind, known as a **bandit problem** (see e.g., [LS19; Sli19]).

### 36.4.1 Types of bandit

In a **multi-armed bandit** problem (MAB) there is an agent (decision maker) that can choose an action from some **policy**  $a_t \sim \pi_t$  at each step, after which it receives a **reward** sampled from the environment,  $r_t \sim p_R(a_t)$ , with expected value  $R(s, a) = \mathbb{E}[R|a]$ .<sup>2</sup>

We can think of this in terms of an agent at a casino who is faced with multiple slot machines, each of which pays out rewards at a different rate. A slot machine is sometimes called a **one-armed bandit**, so a set of  $K$  such machines is called a **multi-armed bandit**; each different action corresponds to pulling the arm of a different slot machine,  $a_t \in \{1, \dots, K\}$ . The goal is to quickly figure out which machine pays out the most money, and then to keep playing that one until you

<sup>2</sup> This is known as a **stochastic bandit**. It is also possible to allow the reward, and possibly the state, to be chosen in an adversarial manner, where nature tries to minimize the reward of the agent. This is known as an **adversarial bandit**.

1  
2 become as rich as possible.

3 We can extend this model by defining a **contextual bandit**, in which the input to the policy  
4 at each step is a randomly chosen state or context  $s_t \in \mathcal{S}$ . The states evolve over time according  
5 to some arbitrary process,  $s_t \sim p(s_t | s_{1:t-1})$ , independent of the actions of the agent. The policy  
6 now has the form  $a_t \sim \pi_t(a_t | s_t)$ , and the reward function now has the form  $r_t \sim p_R(r_t | s_t, a_t)$ , with  
7 expected value  $R(s, a) = \mathbb{E}[R | s, a]$ . At each step, the agent can use the observed data,  $\mathcal{D}_{1:t}$  where  
8  $\mathcal{D}_t = (s_t, a_t, r_t)$ , to update its policy, to maximize expected reward.

9 In the **finite horizon** formulation of (contextual) bandits, the goal is to maximize the expected  
10 **cumulative reward**:

$$\underline{11} \quad J \triangleq \sum_{t=1}^T \mathbb{E}_{p_R(r_t | s_t, a_t) \pi_t(a_t | s_t) p(s_t | s_{1:t-1})} [r_t] = \sum_{t=1}^T \mathbb{E}[r_t] \quad (36.24)$$

14  
15 (Note that the reward is accrued at each step, even while the agent updates its policy; this is  
16 sometimes called “**earning while learning**”.) In the **infinite horizon** formulation, where  $T = \infty$ ,  
17 the cumulative reward may be infinite. To prevent  $J$  from being unbounded, we introduce a **discount**  
18 **factor**  $0 < \gamma < 1$ , so that

$$\underline{19} \quad J \triangleq \sum_{t=1}^{\infty} \gamma^{t-1} \mathbb{E}[r_t] \quad (36.25)$$

22 The quantity  $\gamma$  can be interpreted as the probability that the agent is terminated at any moment in  
23 time (in which case it will cease to accumulate reward).

24 Another way to write this is as follows:

$$\underline{26} \quad J = \sum_{t=1}^{\infty} \gamma^{t-1} \mathbb{E}[r_t] = \sum_{t=1}^{\infty} \gamma^{t-1} \mathbb{E} \left[ \sum_{a=1}^K R_a(s_t, a_t) \right] \quad (36.26)$$

29 where we define

$$\underline{31} \quad R_a(s_t, a_t) = \begin{cases} R(s_t, a) & \text{if } a_t = a \\ 0 & \text{otherwise} \end{cases} \quad (36.27)$$

34 Thus we conceptually evaluate the reward for all arms, but only the one that was actually chosen  
35 (namely  $a_t$ ) gives a non-zero value to the agent, namely  $r_t$ .

36 There are many extensions of the basic bandit problem. A natural one is to allow the agent to  
37 perform **multiple plays**, choosing  $M \leq K$  distinct arms at once. Let  $\mathbf{a}_t$  be the corresponding action  
38 vector which specifies the identity of the chosen arms. Then we define the reward to be

$$\underline{40} \quad r_t = \sum_{a=1}^K R_a(s_t, \mathbf{a}_t) \quad (36.28)$$

43 where

$$\underline{44} \quad R_a(s_t, \mathbf{a}_t) = \begin{cases} R(s_t, a) & \text{if } a \in \mathbf{a}_t \\ 0 & \text{otherwise} \end{cases} \quad (36.29)$$

This is useful for modeling **resource allocation** problems.

Another variant is known as a **restless bandit** [Whi88]. This is the same as the multiple play formulation, except we additionally assume that each arm has its own state vector  $s_t^a$  associated with it, which evolves according to some stochastic process, regardless of whether arm  $a$  was chosen or not. We then define

$$r_t = \sum_{a=1}^K R_a(s_t^a, \mathbf{a}_t) \quad (36.30)$$

where  $s_t^a \sim p(s_t^a | s_{1:t-1}^a)$  is some arbitrary distribution, often assumed to be Markovian. (The fact that the states associated with each arm evolve even if the arm is not picked is what gives rise to the term “restless”.) This can be used to model serial dependence between the rewards given by each arm.

### 36.4.2 Applications

Contextual bandits have many applications. For example, consider an **online advertising system**. In this case, the state  $s_t$  represents features of the web page that the user is currently looking at, and the action  $a_t$  represents the identity of the ad which the system chooses to show. Since the relevance of the ad depends on the page, the reward function has the form  $R(s_t, a_t)$ , and hence the problem is contextual. The goal is to maximize the expected reward, which is equivalent to the expected number of times people click on ads; this is known as the **click through rate** or **CTR**. (See e.g., [Gra+10; Li+10; McM+13; Ago+14; Du+21; YZ22] for more information about this application.)

Another application of contextual bandits arises in **clinical trials** [VBW15]. In this case, the state  $s_t$  are features of the current patient we are treating, and the action  $a_t$  is the treatment the doctor chooses to give them (e.g., a new drug or a **placebo**). Our goal is to maximize expected reward, i.e., the expected number of people who get cured. (An alternative goal is to determine which treatment is best as quickly as possible, rather than maximizing expected reward; this variant is known as **best-arm identification** [ABM10].)

### 36.4.3 Exploration-exploitation tradeoff

The fundamental difficulty in solving bandit problems is known as the **exploration-exploitation tradeoff**. This refers to the fact that the agent needs to try multiple state/action combinations (this is known as exploration) in order to collect enough data so it can reliably learn the reward function  $R(s, a)$ ; it can then exploit its knowledge by picking the predicted best action for each state. If the agent starts exploiting an incorrect model too early, it will collect suboptimal data, and will get stuck in a negative **feedback loop**, as illustrated in Figure 36.4. This is different from supervised learning, where the data is drawn iid from a fixed distribution (see e.g.m [Jeu+19] for details).

We discuss some solutions to the exploration-exploitation problem below.

### 36.4.4 The optimal solution

In this section, we discuss the optimal solution to the exploration-exploitation tradeoff. Let us denote the posterior over the parameters of the reward function by  $\mathbf{b}_t = p(\theta | h_t)$ , where  $h_t = \{s_{1:t-1}, a_{1:t-1}, r_{1:t-1}\}$  is the history of observations; this is known as the **belief state** or

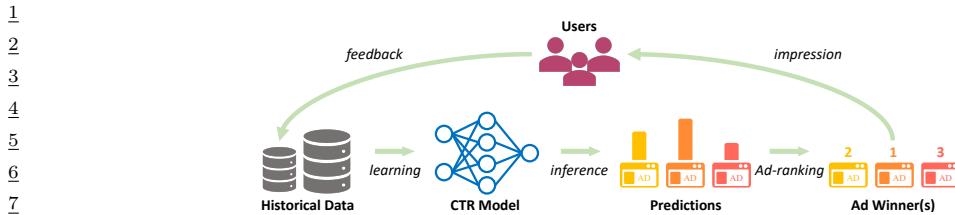


Figure 36.4: Illustration of the feedback problem in online advertising and recommendation systems. The click through rate (CTR) model is used to decide what ads to show, which affects what data is collected, which affects how the model learns. From Figure 1–2 of [Du+21]. Used with kind permission of Chao Du.

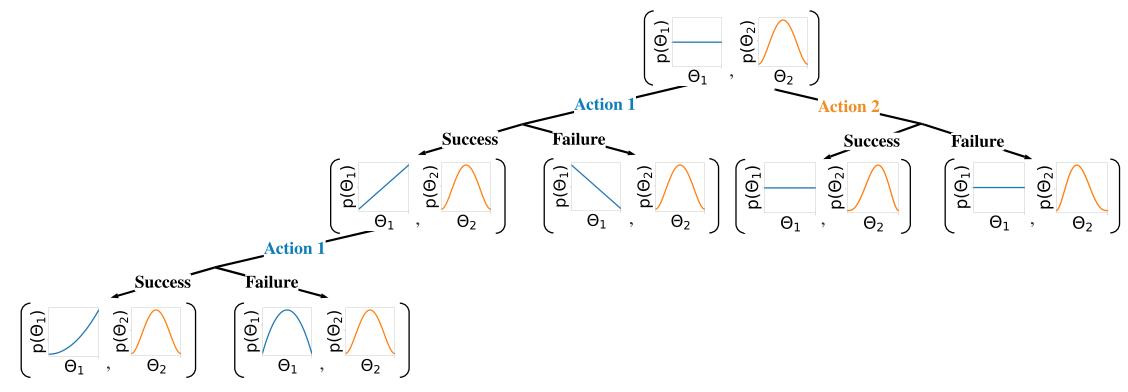


Figure 36.5: Illustration of sequential belief updating for a two-armed beta-Bernoulli bandit. The prior for the reward for action 1 is the (blue) uniform distribution Beta(1, 1); the prior for the reward for action 2 is the (orange) unimodal distribution Beta(2, 2). We update the parameters of the belief state based on the chosen action, and based on whether the observed reward is success (1) or failure (0). (Compare to Figure 3.5, where we display the predictive distribution for a single armed beta-Bernoulli model.)

30

31

**32 information state.** It is a finite sufficient statistic for the history  $\mathbf{h}_t$ . The belief state can be  
33 updated deterministically using Bayes rule:

$$34 \quad \mathbf{b}_t = \text{BayesRule}(\mathbf{b}_{t-1}, a_t, r_t) \quad (36.31)$$

36 For example, consider a context-free **Bernoulli bandit**, where  $p_R(r|a) = \text{Ber}(r|\mu_a)$ , and  $\mu_a =$   
37  $p_R(r=1|a) = R(a)$  is the expected reward for taking action  $a$ . Suppose we use a factored beta prior  
38

$$39 \quad p_0(\boldsymbol{\theta}) = \prod_a \text{Beta}(\mu_a | \alpha_0^a, \beta_0^a) \quad (36.32)$$

41

42 where  $\boldsymbol{\theta} = (\mu_1, \dots, \mu_K)$ . We can compute the posterior in closed form, as we discuss in Section 3.2.1.  
43 In particular, we find

$$44 \quad p(\boldsymbol{\theta} | \mathcal{D}_t) = \prod_a \text{Beta}(\mu_a | \underbrace{\alpha_t^a + N_t^0(a)}_{\alpha_t^a}, \underbrace{\beta_t^a + N_t^1(a)}_{\beta_t^a}) \quad (36.33)$$

47

1 where  
 2

$$N_t^r(a) = \sum_{s=1}^{t-1} \mathbb{I}(a_t = a, r_t = r) \quad (36.34)$$

3  
 4 This is illustrated in Figure 36.5 for a two-armed Bernoulli bandit.  
 5

6 We can use a similar method for a **Gaussian bandit**, where  $p_R(r|a) = \mathcal{N}(r|\mu_a, \sigma_a^2)$ , using results  
 7 from Section 3.2.3. In the case of contextual bandits, the problem becomes more complicated. If we  
 8 assume a **linear regression bandit**,  $p_R(r|s, a; \theta) = \mathcal{N}(r|\phi(s, a)^\top \theta, \sigma^2)$ , we can use Bayesian  
 9 linear regression to compute  $p(\theta|\mathcal{D}_t)$  in closed form, as we discuss in Section 15.2. If we assume  
 10 a **logistic regression bandit**,  $p_R(r|s, a; \theta) = \text{Ber}(r|\sigma(\phi(s, a)^\top \theta))$ , we can use Bayesian logistic  
 11 regression to compute  $p(\theta|\mathcal{D}_t)$ , as we discuss in Section 15.3.4. If we have a **neural bandit** of  
 12 the form  $p_R(r|s, a; \theta) = \text{GLM}(r|f(s, a; \theta))$  for some nonlinear function  $f$ , then posterior inference  
 13 becomes more challenging, as we discuss in Chapter 17. However, standard techniques, such as the  
 14 extended Kalman filter (Section 17.6.1) can be applied. (For a way to scale this approach to large  
 15 DNNs, see the “**subspace neural bandit**” approach of [DMKM22].)  
 16

17 Regardless of the algorithmic details, we can represent the belief state update as follows:  
 18

$$p(\mathbf{b}_t|\mathbf{b}_{t-1}, a_t, r_t) = \mathbb{I}(\mathbf{b}_t = \text{BayesRule}(\mathbf{b}_{t-1}, a_t, r_t)) \quad (36.35)$$

19 The observed reward at each step is then predicted to be  
 20

$$p(r_t|\mathbf{b}_t) = \int p_R(r_t|s_t, a_t; \theta)p(\theta|\mathbf{b}_t)d\theta \quad (36.36)$$

21 We see that this is a special form of a (controlled) Markov decision process (Section 36.5) known as a  
 22 **belief-state MDP**.  
 23

24 In the special case of context-free bandits with a finite number of arms, the optimal policy of this  
 25 belief state MDP can be computed using dynamic programming (c.f., Section 36.6); the result can  
 26 be represented as a table of action probabilities,  $\pi_t(a_1, \dots, a_K)$ , for each step; this is known as the  
 27 **Gittins index** [Git89]. However, computing the optimal policy for general contextual bandits is  
 28 intractable [PT87], so we have to resort to approximations, as we discuss below.  
 29

### 36.4.5 Upper confidence bounds (UCB)

30 The optimal solution to explore-exploit is intractable. However, an intuitively sensible approach is  
 31 based on the principle known as “**optimism in the face of uncertainty**”. The principle selects  
 32 actions greedily, but based on optimistic estimates of their rewards. The most important class of  
 33 strategies with this principle are collectively called **upper confidence bound** or **UCB** methods.  
 34

35 To use a UCB strategy, the agent maintains an optimistic reward function estimate  $\tilde{R}_t$ , so that  
 36  $\tilde{R}_t(s_t, a) \geq R(s_t, a)$  for all  $a$  with high probability, and then chooses the greedy action accordingly:  
 37

$$a_t = \underset{a}{\operatorname{argmax}} \tilde{R}_t(s_t, a) \quad (36.37)$$

38 UCB can be viewed as a form of **exploration bonus**, where the optimistic estimate encourages  
 39 exploration. Typically, the amount of optimism,  $\tilde{R}_t - R$ , decreases over time so that the agent  
 40

1 gradually reduces exploration. With properly constructed optimistic reward estimates, the UCB  
2 strategy has been shown to achieve near-optimal regret in many variants of bandits [LS19]. (We  
3 discuss regret in Section 36.4.7.)  
4

5 The optimistic function  $\tilde{R}$  can be obtained in different ways, sometimes in closed forms, as we  
6 discuss below.  
7

### 8 36.4.5.1 Frequentist approach

9 One approach is to use a **concentration inequality** [BLM16] to derive a high-probability upper  
10 bound of the estimation error:  $|\hat{R}_t(s, a) - R_t(s, a)| \leq \delta_t(s, a)$ , where  $\hat{R}_t$  is a usual estimate of  $R$   
11 (often the MLE), and  $\delta_t$  is a properly selected function. An optimistic reward is then obtained by  
12 setting  $\tilde{R}_t(s, a) = \hat{R}_t(s, a) + \delta_t(s, a)$ .  
13

14 As an example, consider again the context-free Bernoulli bandit,  $R(a) \sim \text{Ber}(\mu(a))$ . The MLE  
15  $\hat{R}_t(a) = \hat{\mu}_t(a)$  is given by the empirical average of observed rewards whenever action  $a$  was taken:  
16

$$\hat{\mu}_t(a) = \frac{N_t^1(a)}{N_t(a)} = \frac{N_t^1(a)}{N_t^0(a) + N_t^1(a)} \quad (36.38)$$

19 where  $N_t^r(a)$  is the number of times (up to step  $t - 1$ ) that action  $a$  has been tried and the observed  
20 reward was  $r$ , and  $N_t(a)$  is the total number of times action  $a$  has been tried:  
21

$$N_t(a) = \sum_{s=1}^{t-1} \mathbb{I}(a_s = a) \quad (36.39)$$

25 Then the **Chernoff-Hoeffding inequality** [BLM16] leads to  $\delta_t(a) = c/\sqrt{N_t(a)}$  for some proper  
26 constant  $c$ , so  
27

$$\tilde{R}_t(a) = \hat{\mu}_t(a) + \frac{c}{\sqrt{N_t(a)}} \quad (36.40)$$

### 32 36.4.5.2 Bayesian approach

33 We may also derive  $\tilde{R}$  from Bayesian inference. If we use a beta prior, we can compute the posterior  
34 in closed form, as shown in Equation (36.33). The posterior mean is  $\hat{\mu}_t(a) = \mathbb{E}[\mu(a)|\mathbf{h}_t] = \frac{\alpha_t^a}{\alpha_t^a + \beta_t^a}$ .  
35 From Equation (3.30), the posterior standard deviation is approximately  
36

$$\hat{\sigma}_t(a) = \sqrt{\mathbb{V}[\mu(a)|\mathbf{h}_t]} \approx \sqrt{\frac{\hat{\mu}_t(a)(1 - \hat{\mu}_t(a))}{N_t(a)}} \quad (36.41)$$

40 We can use similar techniques for a Gaussian bandit, where  $p_R(R|a, \boldsymbol{\theta}) = \mathcal{N}(R|\mu_a, \sigma_a^2)$ ,  $\mu_a$  is the  
41 expected reward, and  $\sigma_a^2$  the variance. If we use a conjugate prior, we can compute  $p(\mu_a, \sigma_a|\mathcal{D}_t)$   
42 in closed form (see Section 3.2.3). Using an uninformative version of the conjugate prior, we find  
43  $\mathbb{E}[\mu_a|\mathbf{h}_t] = \hat{\mu}_t(a)$ , which is just the empirical mean of rewards for action  $a$ . The uncertainty in  
44 this estimate is the standard error of the mean, given by Equation (3.105), i.e.,  $\sqrt{\mathbb{V}[\mu_a|\mathbf{h}_t]} =$   
45  $\hat{\sigma}_t(a)/\sqrt{N_t(a)}$ , where  $\hat{\sigma}_t(a)$  is the empirical standard deviation of the rewards for action  $a$ .  
46

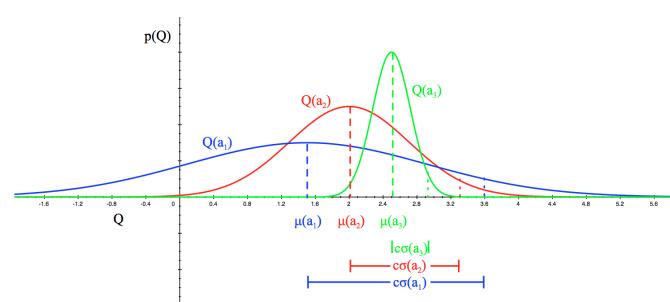


Figure 36.6: Illustration of the reward distribution  $Q(a)$  for 3 different actions, and the corresponding lower and upper confidence bounds. From [Sil18]. Used with kind permission of David Silver.

This approach can also be extended to contextual bandits, modulo the difficulty of computing the belief state.

Once we have computed the mean and posterior standard deviation, we define the optimistic reward estimate as

$$\tilde{R}_t(a) = \hat{\mu}_t(a) + c\hat{\sigma}_t(a) \quad (36.42)$$

for some constant  $c$  that controls how greedy the policy is. We see that this is similar to the frequentist method based on concentration inequalities, but is more general.

#### 36.4.5.3 Example

Figure 36.6 illustrates the UCB principle for a Gaussian bandit. We assume there are 3 actions, and we represent  $p(R(a)|\mathcal{D}_t)$  using a Gaussian. We show the posterior means  $Q(a) = \mu(a)$  with a vertical dotted line, and the scaled posterior standard deviations  $c\sigma(a)$  as a horizontal solid line.

#### 36.4.6 Thompson sampling

A common alternative to UCB is to use **Thompson sampling** [Tho33], also called **probability matching** [Sco10]. In this approach, we define the policy at step  $t$  to be  $\pi_t(a|s_t, \mathbf{h}_t) = p_a$ , where  $p_a$  is the probability that  $a$  is the optimal action. This can be computed using

$$p_a = \Pr(a = a_*|s_t, \mathbf{h}_t) = \int \mathbb{I}\left(a = \operatorname{argmax}_{a'} R(s_t, a'; \boldsymbol{\theta})\right) p(\boldsymbol{\theta}|\mathbf{h}_t) d\boldsymbol{\theta} \quad (36.43)$$

If the posterior is uncertain, the agent will sample many different actions, automatically resulting in exploration. As the uncertainty decreases, it will start to exploit its knowledge.

To see how we can implement this method, note that we can compute the expression in Equation (36.43) by using a single Monte Carlo sample  $\tilde{\boldsymbol{\theta}}_t \sim p(\boldsymbol{\theta}|\mathbf{h}_t)$ . We then plug in this parameter into our reward model, and greedily pick the best action:

$$a_t = \operatorname{argmax}_{a'} R(s_t, a'; \tilde{\boldsymbol{\theta}}_t) \quad (36.44)$$

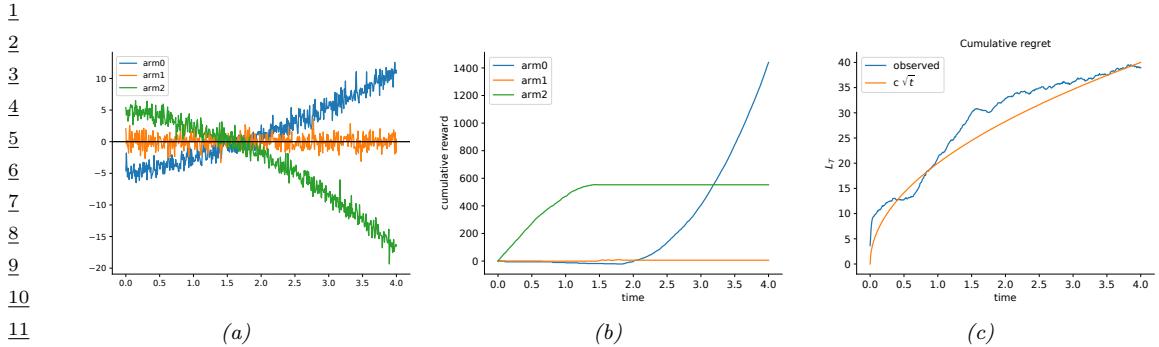


Figure 36.7: Illustration of Thompson sampling applied to a linear-Gaussian contextual bandit. The context has the form  $s_t = (1, t, t^2)$ . (a) True reward for each arm vs time. (b) Cumulative reward per arm vs time. (c) Cumulative regret vs time. Generated by `thompson_sampling_linear_gaussian.py`.

This sample-then-exploit approach will choose actions with exactly the desired probability, since

$$p_a = \int \mathbb{I} \left( a = \operatorname{argmax}_{a'} R(s_t, a'; \tilde{\theta}_t) \right) p(\tilde{\theta}_t | \mathbf{h}_t) = \Pr_{\tilde{\theta}_t \sim p(\theta | \mathbf{h}_t)} (a = \operatorname{argmax}_{a'} R(s_t, a'; \tilde{\theta}_t)) \quad (36.45)$$

Despite its simplicity, this approach can be shown to achieve optimal (logarithmic) regret (see e.g., [Rus+18] for a survey). In addition, it is very easy to implement, and hence is widely used in practice [Gra+10; Sco10; CL11].

In Figure 36.7, we give a simple example of Thompson sampling applied to a linear regression bandit. The context has the form  $s_t = (1, t, t^2)$ . The true reward function has the form  $R(s_t, a) = \mathbf{w}_a^\top s_t$ . The weights per arm are chosen as follows:  $\mathbf{w}_0 = (-5, 2, 0.5)$ ,  $\mathbf{w}_1 = (0, 0, 0)$ ,  $\mathbf{w}_2 = (5, -1.5, -1)$ . Thus we see that arm 0 is initially worse (large negative bias) but gets better over time (positive slope), arm 1 is useless, and arm 2 is initially better (large positive bias) but gets worse over time. The observation noise is the same for all arms,  $\sigma^2 = 1$ . See Figure 36.7(a) for a plot of the reward function.

We use a conjugate Gaussian-Gamma prior and perform exact Bayesian updating. Thompson sampling quickly discovers that arm 1 is useless. Initially it pulls arm 2 more, but it adapts to the non-stationary nature of the problem and switches over to arm 0, as shown in Figure 36.7(b).

34

### 36.4.7 Regret

We have discussed several methods for solving the exploration-exploitation tradeoff. It is useful to quantify the degree of suboptimality of these methods. A common approach is to compute the **regret**, which is defined as the difference between the expected reward under the agent's policy and the oracle policy  $\pi_*$ , which knows the true reward function. (Note that the oracle policy will in general be better than the Bayes optimal policy, which we discussed in Section 36.4.4.)

Specifically, let  $\pi_t$  be the agent's policy at time  $t$ . Then the **per-step regret** at  $t$  is defined as

$$l_t \triangleq \mathbb{E}_{p(s_t)} [R(s_t, \pi_*(s_t))] - \mathbb{E}_{\pi_t(a_t | s_t)p(s_t)} [R(s_t, a_t)] \quad (36.46)$$

If we only care about the final performance of the best discovered arm, as in most optimization problems, it is enough to look at the **simple regret** at the last step, namely  $l_T$ . Optimizing simple

regret results in a problem known as **pure exploration** [BMS11], since there is no need to exploit the information during the learning process. However, it is more common to focus on the the **cumulative regret**, also called the **total regret** or just the **regret**, which is defined as

$$L_T \triangleq \mathbb{E} \left[ \sum_{t=1}^T l_t \right] \quad (36.47)$$

Here the expectation is with respect to randomness in determining  $\pi_t$ , which depends on earlier states, actions and rewards, as well as other potential sources of randomness.

Under the typical assumption that rewards are bounded,  $L_T$  is at most linear in  $T$ . If the agent's policy converges to the optimal policy as  $T$  increases, then the regret is sublinear:  $L_T = o(T)$ . In general, the slower  $L_T$  grows, the more efficient the agent is in trading off exploration and exploitation.

To understand its growth rate, it is helpful to consider again a simple context-free bandit, where  $R_* = \operatorname{argmax}_a R(a)$  is the optimal reward. The total regret in the first  $T$  steps can be written as

$$L_T = \mathbb{E} \left[ \sum_{t=1}^T R_* - R(a_t) \right] = \sum_{a \in \mathcal{A}} \mathbb{E}[N_{T+1}(a)] (R_* - R(a)) = \sum_{a \in \mathcal{A}} \mathbb{E}[N_{T+1}(a)] \Delta_a \quad (36.48)$$

where  $N_{T+1}(a)$  is the total number of times the agent picks action  $a$  up to step  $T$ , and  $\Delta_a = R_* - R(a)$  is the reward **gap**. If the agent under-explores and converges to choosing a suboptimal action (say,  $\hat{a}$ ), then a linear regret is suffered with a per-step regret of  $\Delta_{\hat{a}}$ . On the other hand, if the agent over-explores, then  $N_t(a)$  will be too large for suboptimal actions, and the agent also suffers a linear regret.

Fortunately, it is possible to achieve sublinear regrets, using some of the methods discussed above, such as UCB and Thompson sampling. For example, one can show that Thompson sampling has  $O(\sqrt{KT \log T})$  regret [RR14]. This is shown empirically in Figure 36.7(c).

In fact, both UCB and Thompson sampling are optimal, in the sense that their regrets are essentially not improvable; that is, they match regret lower bounds. To establish such a lower bound, note that the agent needs to collect enough data to distinguish different reward distributions, before identifying the optimal action. Typically, the deviation of the reward estimate from the true reward decays at the rate of  $1/\sqrt{N}$ , where  $N$  is the sample size (see e.g., Equation (3.105)). Therefore, if two reward distributions are similar, distinguishing them becomes harder and requires more samples. (For example, consider the case of a bandit with Gaussian rewards with slightly different means and large variance, as shown in Figure 36.6.)

The following fundamental result is proved by [LR85] for the asymptotic regret (under certain mild assumptions not given here):

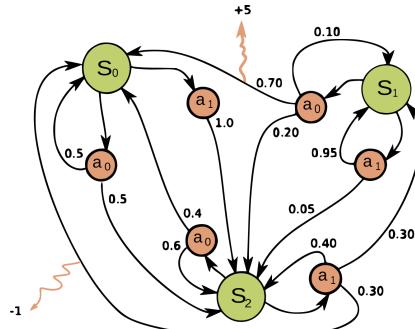
$$\liminf_{T \rightarrow \infty} L_T \geq \log T \sum_{a: \Delta_a > 0} \frac{\Delta_a}{D_{\text{KL}}(p_R(a) \| p_R(a_*))} \quad (36.49)$$

Thus, we see that the best we can achieve is logarithmic growth in the total regret. Similar lower bounds have also been obtained for various bandits variants.

## 36.5 Markov decision problems

In this section, we generalize the discussion of contextual bandits by allowing the state of nature to change depending on the actions chosen by the agent. The resulting model is called a **Markov**

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12



13 *Figure 36.8: Illustration of an MDP as a finite state machine (FSM). The MDP has three discrete states*  
 14 (*green circles*), two discrete actions (*orange circles*), and two non-zero rewards (*orange arrows*). The numbers  
 15 on the black edges represent state transition probabilities, e.g.,  $p(s' = s_0 | a = a_0, s = s_0) = 0.7$ ; most  
 16 state transitions are impossible (probability 0), so the graph is sparse. The numbers on the yellow wiggly  
 17 edges represent expected rewards, e.g.,  $R(s = s_1, a = a_0, s' = s_0) = +5$ ; state transitions with zero reward  
 18 are not annotated. From [https://en.wikipedia.org/wiki/Markov\\_decision\\_process](https://en.wikipedia.org/wiki/Markov_decision_process). Used with kind  
 19 permission of Wikipedia author waldoalvarez.

20

21

22 **decision process** or **MDP**, as we explain in detail below. This model forms the foundation of  
 23 reinforcement learning, which we discuss in Chapter 37.

24

### 25 36.5.1 Basics

26

27 A **Markov decision process** [Put94] can be used to model the interaction of an **agent** and an  
 28 **environment**. It is often described by a tuple  $\langle \mathcal{S}, \mathcal{A}, p_T, p_R, p_0 \rangle$ , where  $\mathcal{S}$  is a set of environment  
 29 states,  $\mathcal{A}$  a set of actions the agent can take,  $p_T$  a **transition model**,  $p_R$  a **reward model**, and  $p_0$   
 30 the initial state distribution. The interaction starts at time  $t = 0$ , where the initial state  $s_0 \sim p_0$ .  
 31 Then, at time  $t \geq 0$ , the agent observes the environment state  $s_t \in \mathcal{S}$ , and follows a **policy**  $\pi$  to  
 32 take an action  $a_t \in \mathcal{A}$ . In response, the environment emits a real-valued reward signal  $r_t \in \mathcal{R}$  and  
 33 enters a new state  $s_{t+1} \in \mathcal{S}$ . The policy is in general stochastic, with  $\pi(a|s)$  being the probability of  
 34 choosing action  $a$  in state  $s$ . We use  $\pi(s)$  to denote the conditional probability over  $\mathcal{A}$  if the policy  
 35 is stochastic, or the action it chooses if it is deterministic. The process at every step is called a  
 36 **transition**; at time  $t$ , it consists of the tuple  $(s_t, a_t, r_t, s_{t+1})$ , where  $a_t \sim \pi(s_t)$ ,  $s_{t+1} \sim p_T(s_t, a_t)$ ,  
 37 and  $r_t \sim p_R(s_t, a_t, s_{t+1})$ . Hence, under policy  $\pi$ , the probability of generating a trajectory  $\tau$  of  
 38 length  $T$  can be written explicitly as

$$39 \quad p(\tau) = p_0(s_0) \prod_{t=0}^{T-1} \pi(a_t | s_t) p_T(s_{t+1} | s_t, a_t) p_R(r_t | s_t, a_t, s_{t+1}) \quad (36.50)$$

40

41 It is useful to define the **reward function** from the reward model  $p_R$ , as the average immediate  
 42 reward of taking action  $a$  in state  $s$ , with the next state marginalized:

$$43 \quad R(s, a) \triangleq \mathbb{E}_{p_T(s'|s, a)} [\mathbb{E}_{p(r|s, a, s')} [r]] \quad (36.51)$$

44

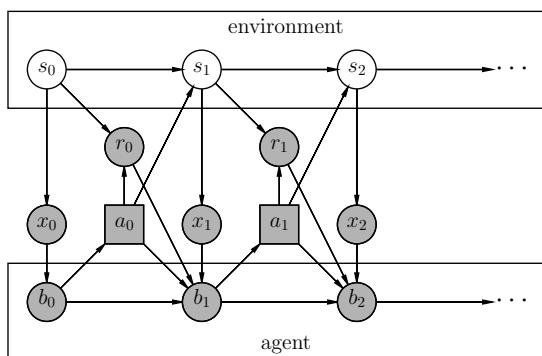


Figure 36.9: Illustration of a partially observable Markov decision process (POMDP) with hidden environment state  $s_t$  which generates the observation  $x_t$ , controlled by an agent with internal belief state  $b_t$  which generates the action  $a_t$ . The reward  $r_t$  depends on  $s_t$  and  $a_t$ . Nodes in this graph represent random variables (circles) and decision variables (squares).

Eliminating the dependence on next states does not lead to loss of generality in the following discussions, as our subject of interest is the total (additive) expected reward along the trajectory. For this reason, we often use the tuple  $\langle \mathcal{S}, \mathcal{A}, p_T, R, p_0 \rangle$  to describe an MDP.

In general, the state and action sets of an MDP can be discrete or continuous. When both sets are finite, we can represent these functions as lookup tables; this is known as a **tabular representation**. In this case, we can represent the MDP as a **finite state machine**, which is a graph where nodes correspond to states, and edges correspond to actions and the resulting rewards and next states. Figure 36.8 gives a simple example of an MDP with 3 states and 2 actions.

The field of **control theory**, which is very closely related to RL, uses slightly different terminology. In particular, the environment is called the **plant**, and the agent is called the **controller**. States are denoted by  $\mathbf{x}_t \in \mathcal{X} \subseteq \mathbb{R}^D$ , actions are denoted by  $\mathbf{u}_t \in \mathcal{U} \subseteq \mathbb{R}^K$ , and rewards are denoted by costs  $c_t \in \mathbb{R}$ . Apart from this notational difference, the fields of RL and control theory are very similar (see e.g., [Son98; Rec19]), although control theory tends to focus on provably optimal methods (by making strong modeling assumptions), whereas RL tends to tackle harder problems with heuristic methods, for which optimality guarantees are often hard to obtain.

### 36.5.2 Partially observed MDPs

An important generalization of the MDP framework relaxes the assumption that the agent sees the hidden world state  $s_t$  directly; instead we assume it only sees a potentially noisy observation generated from the hidden state,  $x_t \sim p(\cdot | s_t, a_t)$ . The resulting model is called a **partially observable Markov decision process** or **POMDP** (pronounced “pom-dee-pee”). Now the agent’s policy is a mapping from all the available data to actions,  $a_t \sim \pi(\mathcal{D}_{1:t-1}, x_t)$ ,  $\mathcal{D}_t = (x_t, a_t, r_t)$ . See Figure 36.9 for an illustration. MDPs are a special case where  $x_t = s_t$ .

In general, POMDPs are much harder to solve than MDPs. A common approximation is to use the last several observed inputs, say  $\mathbf{x}_{t-h:t}$  for history of size  $h$ , as a proxy for the hidden state, and

1  
2 then to treat this as a fully observed MDP.

3

### 4 36.5.3 Episodes and returns

5 The Markov decision process describes how a trajectory  $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots)$  is stochastically  
6 generated. If the agent can potentially interact with the environment forever, we call it a **continuing**  
7 **task**. Alternatively, the agent is in an **episodic task**, if its interaction terminates once the system  
8 enters a **terminal state or absorbing state**;  $s$  is absorbing if the next state from  $s$  is always  $s$   
9 with 0 reward. After entering a terminal state, we may start a new **episode** from a new initial state  
10  $s_0 \sim p_0$ . The episode length is in general random. For example, the amount of time a robot takes to  
11 reach its goal may be quite variable, depending on the decisions it makes, and the randomness in the  
12 environment. Note that we can convert an episodic MDP to a continuing MDP by redefining the  
13 transition model in absorbing states to be the initial-state distribution  $p_0$ . Finally, if the trajectory  
14 length  $T$  in an episodic task is fixed and known, it is called a **finite horizon problem**.

15 Let  $\tau$  be a trajectory of length  $T$ , where  $T$  may be  $\infty$  if the task is continuing. We define the  
16 **return** for the state at time  $t$  to be the sum of expected rewards obtained going forward, where each  
17 reward is multiplied by a **discount factor**  $\gamma \in [0, 1]$ :

$$\frac{19}{20} G_t \triangleq r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{T-t-1} r_{T-1} \quad (36.52)$$

$$\frac{21}{22} = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k} = \sum_{j=t}^{T-1} \gamma^{j-t} r_j \quad (36.53)$$

24  $G_t$  is sometimes called the **reward-to-go**. For episodic tasks that terminate at time  $T$ , we define  
25  $G_t = 0$  for  $t \geq T$ . Clearly, the return satisfies the following recursive relationship:

$$\frac{27}{28} G_t = r_t + \gamma(r_{t+1} + \gamma r_{t+2} + \cdots) = r_t + \gamma G_{t+1} \quad (36.54)$$

29 The discount factor  $\gamma$  plays two roles. First, it ensures the return is finite even if  $T = \infty$  (i.e.,  
30 infinite horizon), provided we use  $\gamma < 1$  and the rewards  $r_t$  are bounded. Second, it puts more weight  
31 on short-term rewards, which generally has the effect of encouraging the agent to achieve its goals  
32 more quickly (see Section 36.5.1 for an example). However, if  $\gamma$  is too small, the agent will become  
33 too greedy. In the extreme case where  $\gamma = 0$ , the agent is completely **myopic**, and only tries to  
34 maximize its immediate reward. In general, the discount factor reflects the assumption that there  
35 is a probability of  $1 - \gamma$  that the interaction will end at the next step. For finite horizon problems,  
36 where  $T$  is known, we can set  $\gamma = 1$ , since we know the life time of the agent a priori.<sup>3</sup>

37

### 38 36.5.4 Value functions

39 Let  $\pi$  be a given policy. We define the **state-value function**, or **value function** for short, as  
40 follows (with  $\mathbb{E}_\pi[\cdot]$  indicating that actions are selected by  $\pi$ ):

$$\frac{42}{43} V_\pi(s) \triangleq \mathbb{E}_\pi[G_0 | s_0 = s] = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s \right] \quad (36.55)$$

45  
46 3. We may also use  $\gamma = 1$  for continuing tasks, targeting the (undiscounted) average reward criterion [Put94].

47

This is the expected return obtained if we start in state  $s$  and follow  $\pi$  to choose actions in a continuing task (i.e.,  $T = \infty$ ).

Similarly, we define the **action-value function**, also known as the  **$Q$ -function**, as follows:

$$Q_\pi(s, a) \triangleq \mathbb{E}_\pi [G_0 | s_0 = s, a_0 = a] = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a \right] \quad (36.56)$$

This quantity represents the expected return obtained if we start by taking action  $a$  in state  $s$ , and then follow  $\pi$  to choose actions thereafter.

Finally, we define the **advantage function** as follows:

$$A_\pi(s, a) \triangleq Q_\pi(s, a) - V_\pi(s) \quad (36.57)$$

This tells us the benefit of picking action  $a$  in state  $s$  then switching to policy  $\pi$ , relative to the baseline return of always following  $\pi$ . Note that  $A_\pi(s, a)$  can be both positive and negative, and  $\mathbb{E}_{\pi(a|s)} [A_\pi(s, a)] = 0$  due to a useful equality:  $V_\pi(s) = \mathbb{E}_{\pi(a|s)} [Q_\pi(s, a)]$ .

### 36.5.5 Optimal value functions and policies

Suppose  $\pi_*$  is a policy such that  $V_{\pi_*} \geq V_\pi$  for all  $s \in \mathcal{S}$  and all policy  $\pi$ , then it is an **optimal policy**. There can be multiple optimal policies for the same MDP, but by definition their value functions must be the same, and are denoted by  $V_*$  and  $Q_*$ , respectively. We call  $V_*$  the **optimal state-value function**, and  $Q_*$  the **optimal action-value function**. Furthermore, any finite MDP must have at least one deterministic optimal policy [Put94].

A fundamental result about the optimal value function is **Bellman's optimality equations**:

$$V_*(s) = \max_a R(s, a) + \gamma \mathbb{E}_{p_T(s'|s, a)} [V_*(s')] \quad (36.58)$$

$$Q_*(s, a) = R(s, a) + \gamma \max_{a'} \mathbb{E}_{p_T(s'|s, a)} [Q_*(s', a')] \quad (36.59)$$

Conversely, the optimal value functions are the only solutions that satisfy the equations. In other words, although the value function is defined as the expectation of a sum of infinitely many rewards, it can be characterized by a recursive equation that involves only one-step transition and reward models of the MDP. Such a recursion play a central role in many RL algorithms we will see later in this chapter. Given a value function ( $V$  or  $Q$ ), the discrepancy between the right- and left-hand sides of Equations (36.58) and (36.59) are called **Bellman error** or **Bellman residual**.

Furthermore, given the optimal value function, we can derive an optimal policy using

$$\pi_*(s) = \operatorname{argmax}_a Q_*(s, a) \quad (36.60)$$

$$= \operatorname{argmax}_a [R(s, a) + \gamma \mathbb{E}_{p_T(s'|s, a)} [V_*(s')]] \quad (36.61)$$

Following such an optimal policy ensures the agent achieves maximum expected return starting from any state. The problem of solving for  $V_*$ ,  $Q_*$  or  $\pi_*$  is called **policy optimization**. In contrast, solving for  $V_\pi$  or  $Q_\pi$  for a given policy  $\pi$  is called **policy evaluation**, which constitutes an important subclass of RL problems as will be discussed in later sections. For policy evaluation, we have similar Bellman equations, which simply replace  $\max_a \{\cdot\}$  in Equations (36.58) and (36.59) with  $\mathbb{E}_{\pi(a|s)} [\cdot]$ .

<sup>1</sup> In Equations (36.60) and (36.61), as in the Bellman optimality equations, we must take a maximum  
<sup>2</sup> over all actions in  $\mathcal{A}$ , and the maximizing action is called the **greedy action** with respect to the  
<sup>3</sup> value functions,  $Q_*$  or  $V_*$ . Finding greedy actions is computationally easy if  $\mathcal{A}$  is a small finite set.  
<sup>4</sup> For high dimensional continuous spaces, we can treat  $a$  as a sequence of actions, and optimize one  
<sup>5</sup> dimension at a time [Met+17], or use gradient-free optimizers such as cross-entropy method (see  
<sup>6</sup> supplementary), as used in the **QT-Opt** method [Kal+18a]. Recently, **CAQL** (continuous action  
<sup>7</sup>  $Q$ -learning, [Ryu+20]) proposed to use mixed integer programming to solve the argmax problem,  
<sup>8</sup> leveraging the ReLU structure of the  $Q$ -network. We can also amortize the cost of this optimization  
<sup>9</sup> by training a policy  $a_* = \pi_*(s)$  after learning the optimal  $Q$ -function.  
<sup>10</sup>

<sup>11</sup>

### <sup>12</sup> 36.5.5.1 Example

<sup>13</sup> In this section, we show a simple example, to make concepts like value functions more concrete.  
<sup>14</sup> Consider the 1d **grid world** shown in Figure 36.10(a). There are 5 possible states, among them  $S_{T1}$   
<sup>15</sup> and  $S_{T2}$  are absorbing states, since the interaction ends once the agent enters them. There are 2  
<sup>16</sup> actions,  $\uparrow$  and  $\downarrow$ . The reward function is zero everywhere except at the goal state,  $S_{T2}$ , which gives a  
<sup>17</sup> reward of 1 upon entering. Thus the optimal action in every state is to move down.  
<sup>18</sup>

<sup>19</sup> Figure 36.10(b) shows the  $Q_*$  function for  $\gamma = 0$ . Note that we only show the function for  
<sup>20</sup> non-absorbing states, as the optimal  $Q$ -values are 0 in absorbing states by definition. We see that  
<sup>21</sup>  $Q_*(s_3, \downarrow) = 1.0$ , since the agent will get a reward of 1.0 on the next step if it moves down from  $s_3$ ;  
<sup>22</sup> however,  $Q_*(s, a) = 0$  for all other state-action pairs, since they do not provide nonzero immediate  
<sup>23</sup> reward. This optimal  $Q$ -function reflects the fact that using  $\gamma = 0$  is completely myopic, and ignores  
<sup>24</sup> the future.

<sup>25</sup> Figure 36.10(c) shows  $Q_*$  when  $\gamma = 1$ . In this case, we care about all future rewards equally. Thus  
<sup>26</sup>  $Q_*(s, a) = 1$  for all state-action pairs, since the agent can always reach the goal eventually. This is  
<sup>27</sup> infinitely far-sighted. However, it does not give the agent any short-term guidance on how to behave.  
<sup>28</sup> For example, in  $s_2$ , it is not clear if it should go up or down, since both actions will eventually  
<sup>29</sup> reach the goal with identical  $Q_*$ -values.

<sup>30</sup> Figure 36.10(d) shows  $Q_*$  when  $\gamma = 0.9$ . This reflects a preference for near-term rewards, while  
<sup>31</sup> also taking future reward into account. This encourages the agent to seek the shortest path to the  
<sup>32</sup> goal, which is usually what we desire. A proper choice of  $\gamma$  is up to the agent designer, just like the  
<sup>33</sup> design of the reward function, and has to reflect the desired behavior of the agent.

<sup>34</sup>

## <sup>35</sup> 36.6 Planning in an MDP

<sup>36</sup>

<sup>37</sup> In this section, we discuss how to compute an optimal policy when the MDP model is known. This  
<sup>38</sup> problem is called **planning**, in contrast to the learning problem where the models are unknown,  
<sup>39</sup> which is tackled using reinforcement learning Chapter 37. The planning algorithms we discuss are  
<sup>40</sup> based on **dynamic programming** (DP) and **linear programming** (LP).

<sup>41</sup> For simplicity, in this section, we assume discrete state and action sets with  $\gamma < 1$ . However, exact  
<sup>42</sup> calculation of optimal policies often depends polynomially on the sizes of  $\mathcal{S}$  and  $\mathcal{A}$ , and is intractable,  
<sup>43</sup> for example, when the state space is a Cartesian product of several finite sets. This challenge is known  
<sup>44</sup> as the **curse of dimensionality**. Therefore, approximations are typically needed, such as using  
<sup>45</sup> parametric or nonparametric representations of the value function or policy, both for computational  
<sup>46</sup> tractability and for extending the methods to handle MDPs with general state and action sets.  
<sup>47</sup>

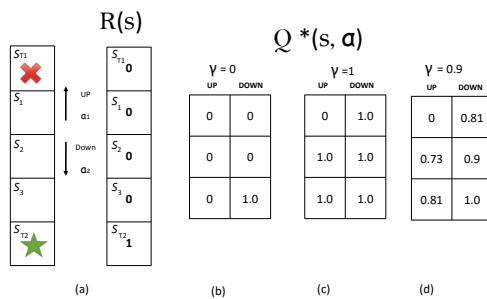


Figure 36.10: Left: illustration of a simple MDP corresponding to a 1d grid world of 3 non-absorbing states and 2 actions. Right: optimal Q-functions for different values of  $\gamma$ . Adapted from Figures 3.1, 3.2, 3.4 of [GK19].

In this case, we have **approximate dynamic programming** (ADP) and **approximate linear programming** (ALP) algorithms (see e.g., [Ber19]).

### 36.6.1 Value iteration

A popular and effective DP method for solving an MDP is **value iteration** (VI). Starting from an initial value function estimate  $V_0$ , the algorithm iteratively updates the estimate by

$$V_{k+1}(s) = \max_a \left[ R(s, a) + \gamma \sum_{s'} p(s'|s, a) V_k(s') \right] \quad (36.62)$$

Note that the update rule, sometimes called a **Bellman backup**, is exactly the right-hand side of the Bellman optimality equation Equation (36.58), with the unknown  $V_*$  replaced by the current estimate  $V_k$ . A fundamental property of Equation (36.62) is that the update is a **contraction**: it can be verified that

$$\max_s |V_{k+1}(s) - V_*(s)| \leq \gamma \max_s |V_k(s) - V_*(s)| \quad (36.63)$$

In other words, every iteration will reduce the maximum value function error by a constant factor. It follows immediately that  $V_k$  will converge to  $V_*$ , after which an optimal policy can be extracted using Equation (36.61). In practice, we can often terminate VI when  $V_k$  is close enough to  $V_*$ , since the resulting greedy policy wrt  $V_k$  will be near optimal. Value iteration can be adapted to learn the optimal action-value function  $Q_*$ .

In value iteration, we compute  $V_*(s)$  and  $\pi_*(s)$  for all possible states  $s$ , averaging over all possible next states  $s'$  at each iteration, as illustrated in Figure 36.11(right). However, for some problems, we may only be interested in the value (and policy) for certain special starting states. This is the case, for example, in **shortest path problems** on graphs, where we are trying to find the shortest route from the current state to a goal state. This can be modeled as an episodic MDP by defining a

1 transition matrix  $p_T(s'|s, a)$  where taking edge  $a$  from node  $s$  leads to the neighboring node  $s'$  with  
2 probability 1. The reward function is defined as  $R(s, a) = -1$  for all states  $s$  except the goal states,  
3 which are modeled as absorbing states.  
4

5 In problems such as this, we can use a method known as **real-time dynamic programming**  
6 (RTDP) [BBS95], to efficiently compute an **optimal partial policy**, which only specifies what to do  
7 for the reachable states. RTDP maintains a value function estimate  $V$ . At each step, it performs a  
8 Bellman backup for the current state  $s$  by  $V(s) \leftarrow \max_a \mathbb{E}_{p_T(s'|s, a)} [R(s, a) + \gamma V(s')]$ . It can picks an  
9 action  $a$  (often with some exploration), reaches a next state  $s'$ , and repeats the process. This can be  
10 seen as a form of the more general **asynchronous value iteration**, that focuses its computational  
11 effort on parts of the state space that are more likely to be reachable from the current state, rather  
12 than synchronously updating all states at each iteration.

13

#### 14 36.6.2 Policy iteration

15 Another effective DP method for computing  $\pi_*$  is **policy iteration**. It is an iterative algorithm that  
16 searches in the space of deterministic policies until converging to an optimal policy. Each iteration  
17 consists of two steps, **policy evaluation** and **policy improvement**.  
18

19 The policy evaluation step, as mentioned earlier, computes the value function for the current  
20 policy. Let  $\pi$  represent the current policy,  $\mathbf{v}(s) = V_\pi(s)$  represent the value function encoded as  
21 a vector indexed by states,  $\mathbf{r}(s) = \sum_a \pi(a|s) R(s, a)$  represent the reward vector, and  $\mathbf{T}(s'|s) =$   
22  $\sum_a \pi(a|s) p(s'|s, a)$  represent the state transition matrix. Bellman's equation for policy evaluation  
23 can be written in the matrix-vector form as

$$\underline{24} \quad \mathbf{v} = \mathbf{r} + \gamma \mathbf{T}\mathbf{v} \quad (36.64)$$

25

26 This is a linear system of equations in  $|\mathcal{S}|$  unknowns. We can solve it using matrix inversion:  
27  $\mathbf{v} = (\mathbf{I} - \gamma \mathbf{T})^{-1} \mathbf{r}$ . Alternatively, we can use value iteration by computing  $\mathbf{v}_{t+1} = \mathbf{r} + \gamma \mathbf{T}\mathbf{v}_t$  until near  
28 convergence, or some form of asynchronous variant that is computationally more efficient.

29 Once we have evaluated  $V_\pi$  for the current policy  $\pi$ , we can use it to derive a better policy  $\pi'$ , thus  
30 the name policy improvement. To do this, we simply compute a deterministic policy  $\pi'$  that acts  
31 greedily with respect to  $V_\pi$  in every state; that is,  $\pi'(s) = \operatorname{argmax}_a \{R(s, a) + \gamma \mathbb{E}[V_\pi(s')]\}$ . We can  
32 guarantee that  $V_{\pi'} \geq V_\pi$ . To see this, define  $\mathbf{r}'$ ,  $\mathbf{T}'$  and  $\mathbf{v}'$  as before, but for the new policy  $\pi'$ . The  
33 definition of  $\pi'$  implies  $\mathbf{r}' + \gamma \mathbf{T}'\mathbf{v}' \geq \mathbf{r} + \gamma \mathbf{T}\mathbf{v} = \mathbf{v}$ , where the equality is due to Bellman's equation.  
34 Repeating the same equality, we have

$$\underline{35} \quad \mathbf{v} \leq \mathbf{r}' + \gamma \mathbf{T}'\mathbf{v} \leq \mathbf{r}' + \gamma \mathbf{T}'(\mathbf{r}' + \gamma \mathbf{T}'\mathbf{v}) \leq \mathbf{r}' + \gamma \mathbf{T}'(\mathbf{r}' + \gamma \mathbf{T}'(\mathbf{r}' + \gamma \mathbf{T}'\mathbf{v})) \leq \dots \quad (36.65)$$

$$\underline{37} \quad = (\mathbf{I} + \gamma \mathbf{T}' + \gamma^2 \mathbf{T}'^2 + \dots) \mathbf{r} = (\mathbf{I} - \gamma \mathbf{T}')^{-1} \mathbf{r} = \mathbf{v}' \quad (36.66)$$

38

39 Starting from an initial policy  $\pi_0$ , policy iteration alternates between policy evaluation ( $E$ ) and  
40 improvement ( $I$ ) steps, as illustrated below:

$$\underline{41} \quad \pi_0 \xrightarrow{E} V_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V_{\pi_1} \dots \xrightarrow{I} \pi_* \xrightarrow{E} V_* \quad (36.67)$$

42

43 The algorithm stops at iteration  $k$ , if the policy  $\pi_k$  is greedy with respect to its own value function  
44  $V_{\pi_k}$ . In this case, the policy is optimal. Since there are at most  $|\mathcal{A}|^{|\mathcal{S}|}$  deterministic policies, and  
45 every iteration strictly improves the policy, the algorithm must converge after finite iterations.  
46

47

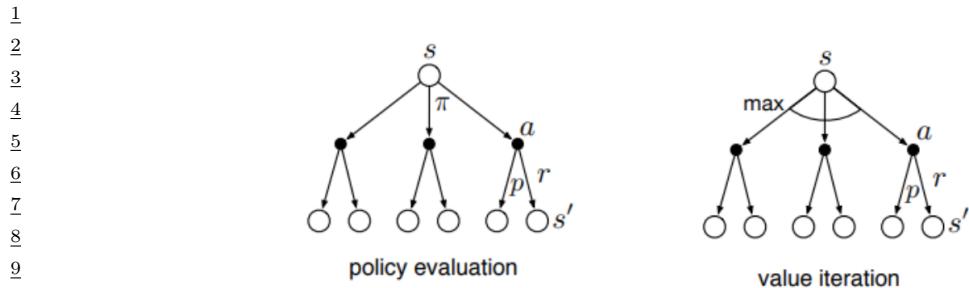


Figure 36.11: Policy iteration vs value iteration represented as backup diagrams. Empty circles represent states, solid (filled) circles represent actions. Adapted from Figure 8.6 of [SB18].

In PI, we alternate between policy evaluation (which involves multiple iterations, until convergence of  $V_\pi$ ), and policy improvement. In VI, we alternate between one iteration of policy evaluation followed by one iteration of policy improvement (the “max” operator in the update rule). In **generalized policy improvement**, we are free to intermix any number of these steps in any order. The process will converge once the policy is greedy wrt its own value function.

Note that policy evaluation computes  $V_\pi$  whereas value iteration computes  $V_*$ . This difference is illustrated in Figure 36.11, using a **backup diagram**. Here the root node represents any state  $s$ , nodes at the next level represent state-action combinations (solid circles), and nodes at the leaves representing the set of possible resulting next state  $s'$  for each possible action. In the former case, we average over all actions according to the policy, whereas in the latter, we take the maximum over all actions.

### 36.6.3 Linear programming

While dynamic programming is effective and popular, linear programming (LP) provides an alternative that finds important uses, such as in off-policy RL (Section 37.5). The primal form of LP is given by

$$\min_V \sum_s p_0(s)V(s) \quad \text{s.t.} \quad V(s) \geq R(s, a) + \gamma \sum_{s'} p_T(s'|s, a)V(s), \quad \forall (s, a) \in \mathcal{S} \times \mathcal{A} \quad (36.68)$$

where  $p_0(s) > 0$  for all  $s \in \mathcal{S}$ , and can be interpreted as the initial state distribution. It can be verified that any  $V$  satisfying the constraint in Equation (36.68) is optimistic [Put94], that is,  $V \geq V_*$ . When the objective is minimized, the solution  $V$  will be “pushed” to the smallest possible, which is  $V_*$ . Once  $V_*$  is found, any action  $a$  that makes the constraint tight in state  $s$  is optimal in that state.

The dual LP form is sometimes more intuitive:

$$\max_{d \geq 0} \sum_{s, a} d(s, a)R(s, a) \quad \text{s.t.} \quad \sum_a d(s, a) = (1 - \gamma)p_0(s) + \gamma \sum_{\bar{s}, \bar{a}} p_T(s|\bar{s}, \bar{a})d(\bar{s}, \bar{a}) \quad \forall s \in \mathcal{S} \quad (36.69)$$

Any nonnegative  $d$  satisfying the constraint above is the **normalized occupancy distribution** of

1 some corresponding policy  $\pi_d(a|s) \triangleq d(s, a) / \sum_{a'} d(s, a')$ : <sup>4</sup>

2

$$\frac{4}{5} \quad d(s, a) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t p(s_t = s, a_t = a | s_0 \sim p_0, a_t \sim \pi_d(s_t)) \quad (36.70)$$

3

4

5

6

7 The constant  $(1 - \gamma)$  normalizes  $d$  to be a valid distribution, so that it sums to unity. With this  
8 interpretation of  $d$ , the objective in Equation (36.69) is just the average per-step reward under the  
9 normalized occupancy distribution. Once an optimal solution  $d_*$  is found, an optimal policy can be  
10 immediately obtained by  $\pi_*(a|s) = d_*(s, a) / \sum_{a'} d_*(s, a')$ .

11 A challenge in solving the primal or dual LPs for MDPs is the large number of constraints and  
12 variables. Approximations are needed, where the variables are parameterized (either linearly or  
13 nonlinearly), and the the constraints are sampled or approximated (see e.g., [dV04; LBS17; CLW18]).

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46 4. If  $\sum_{a'} d(s, a') = 0$  for some state  $s$ , then  $\pi_d(s)$  may be defined arbitrarily, since  $s$  is not visited under the policy.

47

# 37 Reinforcement learning

*This chapter was co-authored with Lihong Li.*

## 37.1 Introduction

**Reinforcement learning** or **RL** is a paradigm of learning where an agent sequentially interacts with an initially unknown environment. The interaction typically results in a **trajectory**, or multiple trajectories. Let  $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, s_2, \dots, s_T)$  be a trajectory of length  $T$ , consisting of a sequence of states  $s_t$ , actions  $a_t$ , and rewards  $r_t$ .<sup>1</sup> The goal of the agent is to optimize her action-selection policy, so that the discounted cumulative reward,  $G_0 \triangleq \sum_{t=0}^{T-1} \gamma^t r_t$ , is maximized for some given **discount factor**  $\gamma \in [0, 1]$ .

In general,  $G_0$  is a random variable. We will focus on maximizing its expectation, inspired by the maximum expected utility principle (Section 3.8.1), but note other possibilities such as **conditional value at risk**<sup>2</sup> that can be more appropriate in risk-sensitive applications.

We will focus on the Markov decision process, where the generative model for the trajectory  $\tau$  can be factored into single-step models. When these model parameters are known, solving for an optimal policy is called **planning** (see Section 36.6); otherwise, RL algorithms may be used to obtain an optimal policy from trajectories, a process called **learning**.

In **model-free RL**, we try to learn the policy without explicitly representing and learning the models, but directly from the trajectories. In **model-based RL**, we first learn a model from the trajectories, and then use a planning algorithm on the learned model to solve for the policy. This chapter will introduce some of the key concepts and techniques, and will mostly follow the notation from [SB18]. More details can be found in textbooks such as [Sze10; SB18; Ber19; Aga+21a; Mey22], and reviews such as [WO12; Aru+17; FL+18; Li18].

### 37.1.1 Overview of methods

In this section, we give a brief overview of how to compute optimal policies when the MDP model is not known. Instead, the agent interacts with the environment and learns from the observed

---

1. Note that the time starts at 0 here, while it starts at 1 when we discuss bandits (Section 36.4). Our choices of notation is to be consistent with conventions in respective literature.

2. The conditional value at risk, or CVaR, is the expected reward conditioned on being in the worst 5% (say) of samples. See [Cho+15] for an example application in RL.

	Method	Functions learned	On/Off	Section
1	SARSA	$Q(s, a)$	On	Section 37.2.4
2	<i>Q</i> -learning	$Q(s, a)$	Off	Section 37.2.5
3	REINFORCE	$\pi(a s)$	On	Section 37.3.2
4	A2C	$\pi(a s), V(s)$	On	Section 37.3.3.1
5	TRPO / PPO	$\pi(a s), A(s, a)$	On	Section 37.3.4
6	DDPG	$a = \pi(s), Q(s, a)$	Off	Section 37.3.5
7	Soft actor-critic	$\pi(a s), Q(s, a)$	Off	Section 37.6.1
8	Model-based RL	$p(s' s, a)$	Off	Section 37.4
9				
10				
11				
12	<i>Table 37.1: Summary of some popular methods for RL. On/off refers to on-policy vs off-policy methods.</i>			
13				
14				
15	trajectories. This is the core focus of RL. We will go into more details into later sections, but first			
16	provide this roadmap.			
17	We may categorize RL methods by the quantity the agent represents and learns: value function,			
18	policy, and model; or by how actions are selected: on-policy (actions must be selected by the agent's			
19	current policy), and off-policy. Table 37.1 lists a few representative examples. More details are given			
20	in the subsequent sections. We will also discuss at greater depth two important topics of off-policy			
21	learning and inference-based control in Sections 37.5 and 37.6.			
22				
23	<b>37.1.2 Value based methods</b>			
24	In a value based method, we often try to learn the optimal $Q$ -function from experience, and then			
25	derive a policy from it using Equation (36.60). Typically, a function approximator (e.g., a neural			
26	network), $Q_w$ , is used to represent the $Q$ -function, which is trained iteratively. Given a transition			
27	$(s, a, r, s')$ , we define the <b>temporal difference</b> (also called the <b>TD error</b> ) as			
28				
29	$r + \gamma \max_a Q_w(s', a') - Q_w(s, a)$			
30				
31	Clearly, the expected TD error is the Bellman error evaluated at $(s, a)$ . Therefore, if $Q_w = Q_*$ , the			
32	TD error is 0 on average by Bellman's optimality equation. Otherwise, the error provides a signal for			
33	the agent to change $w$ to make $Q_w(s, a)$ closer to $R(s, a) + \gamma \max_a Q_w(s', a')$ . The update on $Q_w$			
34	is based on a target that is computed using $Q_w$ . This kind of update is known as <b>bootstrapping</b>			
35	in RL, and should not be confused with the statistical bootstrap (Section 13.2.3.1). Value based			
36	methods such as <b>Q-learning</b> and <b>SARSA</b> are discussed in Section 37.2.			
37				
38	<b>37.1.3 Policy search methods</b>			
39				
40	In <b>policy search</b> , we try to directly maximize $J(\pi_\theta)$ wrt the policy parameter $\theta$ . If $J(\pi_\theta)$ is			
41	differentiable wrt $\theta$ , we can use stochastic gradient ascent to optimize $\theta$ , which is known as <b>policy</b>			
42	<b>gradient</b> , as described in Section 37.3.1. The basic idea is to perform <b>Monte Carlo rollouts</b> , in			
43	which we sample trajectories by interacting with the environment, and then use the score function			
44	estimator (Section 6.6.3) to estimate $\nabla_\theta J(\pi_\theta)$ . Here, $J(\pi_\theta)$ is defined as an expectation whose			
45	distribution depends on $\theta$ , so it is invalid to swap $\nabla$ and $\mathbb{E}$ in computing the gradient, and the score			
46	function estimator can be used instead. An example of policy gradient is <b>REINFORCE</b> .			
47				

Policy gradient methods have the advantage that they provably converge to a local optimum for many common policy classes, whereas  $Q$ -learning may diverge when approximation is used (Section 37.5.3). In addition, policy gradient methods can easily be applied to continuous action spaces, since they do not need to compute  $\text{argmax}_a Q(s, a)$ . Unfortunately, the score function estimator for  $\nabla_{\theta} J(\pi_{\theta})$  can have a very high variance, so the resulting method can converge slowly.

One way to reduce the variance is to learn an approximate value function,  $V_w(s)$ , and to use it as a baseline in the score function estimator. We can learn  $V_w(s)$  using one of the value function methods similar to  $Q$ -learning. Alternatively, we can learn an advantage function,  $A_w(s, a)$ , and use it to estimate the gradient. These policy gradient variants are called **actor critic** methods, where the actor refers to the policy  $\pi_{\theta}$  and the critic refers to  $V_w$  or  $A_w$ . See Section 37.3.3 for details.

#### 37.1.4 Model-based RL

Value-based methods, such as  $Q$ -learning, and policy search methods, such as policy gradient, can be very **sample inefficient**, which means they may need to interact with the environment many times before finding a good policy. If an agent has prior knowledge of the MDP model, it can be more sample efficient to first learn the model, and then compute an optimal (or near-optimal) policy of the model without having to interact with the environment any more.

This approach is called **model-based RL**. The first step is to learn the MDP model including the  $p_T(s'|s, a)$  and  $R(s, a)$  functions, e.g., using DNNs. Given a collection of  $(s, a, r, s')$  tuples, such a model can be learned using standard supervised learning methods. The second step can be done by running an RL algorithm on synthetic experiences generated from the model, or by running a planning algorithm on the model directly (Section 36.6). In practice, we often interleave the model learning and planning phases, so we can use the partially learned policy to decide what data to collect. We discuss model-based RL in more detail in Section 37.4.

#### 37.1.5 Exploration-exploitation tradeoff

A fundamental problem in RL with unknown transition and reward models is to decide between choosing actions that the agent knows will yield high reward, or choosing actions whose reward is uncertain, but which may yield information that helps the agent get to parts of state-action space with even higher reward. This is called the **exploration-exploitation tradeoff**, which has been discussed in the simpler contextual bandit setting in Section 36.4. The literature on efficient exploration is huge. In this section, we briefly describe several representative techniques.

##### 37.1.5.1 $\epsilon$ -greedy

A common heuristic is to use an  **$\epsilon$ -greedy** policy  $\pi_{\epsilon}$ , parameterized by  $\epsilon \in [0, 1]$ . In this case, we pick the greedy action wrt the current model,  $a_t = \text{argmax}_a \hat{R}_t(s_t, a)$  with probability  $1 - \epsilon$ , and a random action with probability  $\epsilon$ . This rule ensures the agent's continual exploration of all state-action combinations. Unfortunately, this heuristic can be shown to be suboptimal, since it explores every action with at least a constant probability  $\epsilon / |\mathcal{A}|$ .

	$\hat{R}(s, a_1)$	$\hat{R}(s, a_2)$	$\pi_\epsilon(a s_1)$	$\pi_\epsilon(a s_2)$	$\pi_\tau(a s_1)$	$\pi_\tau(a s_2)$
1	1.00	9.00	0.05	0.95	0.00	1.00
2	4.00	6.00	0.05	0.95	0.12	0.88
3	4.90	5.10	0.05	0.95	0.45	0.55
4	5.05	4.95	0.95	0.05	0.53	0.48
5	7.00	3.00	0.95	0.05	0.98	0.02
6	8.00	2.00	0.95	0.05	1.00	0.00
7						
8						

9  
10 *Table 37.2: Comparison of  $\epsilon$ -greedy policy (with  $\epsilon = 0.1$ ) and Boltzmann policy (with  $\tau = 1$ ) for a simple  
11 MDP with 6 states and 2 actions. Adapted from Table 4.1 of [GK19].*

12

### 13 37.1.5.2 Boltzmann exploration 14

15 A source of inefficiency in the  $\epsilon$ -greedy rule is that exploration occurs uniformly over all actions.

16 The **Boltzmann policy** can be more efficient, by assigning higher probabilities to explore more  
17 promising actions:

18

$$19 \quad \pi_\tau(a|s) = \frac{\exp(\hat{R}_t(s_t, a)/\tau)}{\sum_{a'} \exp(\hat{R}_t(s_t, a')/\tau)} \quad (37.1)$$

20

21 where  $\tau > 0$  is a temperature parameter that controls how entropic the distribution is. As  $\tau$  gets  
22 close to 0,  $\pi_\tau$  becomes close to a greedy policy. On the other hand, higher values of  $\tau$  will make  
23  $\pi(a|s)$  more uniform, and encourage more exploration. Its action selection probabilities can be much  
24 “smoother” with respect to changes in the reward estimates than  $\epsilon$ -greedy, as illustrated in Table 37.2.  
25

26

### 27 37.1.5.3 Upper confidence bounds and Thompson sampling

28 The upper confidence bound (UCB) (Section 36.4.5) and Thompson sampling (Section 36.4.6)  
29 approaches may also be extended to MDPs. In contrast to the contextual bandit case, where the  
30 only uncertainty is in the reward function, here we must also take into account uncertainty in the  
31 transition probabilities.

32

33 As in the bandit case, the UCB approach requires to estimate an upper confidence bound for all  
34 actions'  $Q$ -values in the current state, and then take the action with the highest UCB score. One way  
35 to obtain UCBs of the  $Q$ -values is to use **count-based exploration**, where we learn the optimal  
36  $Q$ -function with an **exploration bonus** added to the reward in a transition  $(s, a, r, s')$ :

$$37 \quad \tilde{r} = r + \alpha / \sqrt{N_{s,a}} \quad (37.2)$$

38

39 where  $N_{s,a}$  is the number of times action  $a$  has been taken in state  $s$ , and  $\alpha \geq 0$  is a weighting term  
40 that controls the degree of exploration. This is the approach taken by the **MBIE-EB** method [SL08]  
41 for finite-state MDPs, and in the generalization to continuous-state MDPs through the use of  
42 hashing [Bel+16]. Other approaches also explicitly maintain uncertainty in state transition proba-  
43 bilities, and use that information to obtain UCBs. Examples are **MBIE** [SL08], **UCRL2** [JOA10],  
44 **UCBVI** [AOM17], among many others.

45 Thompson sampling can be similarly adapted, by maintaining the posterior distribution of the  
46 reward and transition model parameters. In finite-state MDPs, for example, the transition model is a  
47

categorical distribution conditioned on the state. We may use the conjugate prior of Dirichlet distributions (Section 3.2) for the transition model, so that the posterior distribution can be conveniently computed and sampled from. More details on this approach are found in [Rus+18].

Both UCB and Thompson sampling methods have been shown to yield efficient exploration with provably strong regret bounds (Section 36.4.7) [JOA10], or related PAC bounds [SLL09; DLB17], often under necessary assumptions such as finiteness of the MDPs. In practice, these methods may be combined with function approximation like neural networks and implemented approximately.

#### 37.1.5.4 Optimal solution using Bayes-adaptive MDPs

The Bayes optimal solution to the exploration-exploitation tradeoff can be computed by formulating the problem as a special kind of POMDP known as a **Bayes-adaptive MDP** or **BAMDP** [Duf02]. This extends the Gittins index approach in Section 36.4.4 to the MDP setting.

In particular, a BAMDP has a **belief state** space,  $\mathcal{B}$ , representing uncertainty about the reward model  $p_R(r|s, a, s')$  and transition model  $p_T(s'|s, a)$ . The transition model on this augmented MDP can be written as follows:

$$T^+(s_{t+1}, b_{t+1} | s_t, b_t, a_t, r_t) = T^+(s_{t+1} | s_t, a_t, b_t) T^+(b_{t+1} | s_t, a_t, r_t, s_{t+1}) \quad (37.3)$$

$$= \mathbb{E}_{b_t} [T(s_{t+1} | s_t, a_t)] \times \mathbb{I}(b_{t+1} = p(R, T | \mathbf{h}_{t+1})) \quad (37.4)$$

where  $\mathbb{E}_{b_t} [T(s_{t+1} | s_t, a_t)]$  is the posterior predictive distribution over next states, and  $p(R, T | \mathbf{h}_{t+1})$  is the new belief state given  $\mathbf{h}_{t+1} = (s_{1:t+1}, a_{1:t+1}, r_{1:t+1})$ , which can be computed using Bayes rule.

Similarly, the reward function for the augmented MDP is given by

$$R^+(r | s_t, b_t, a_t, s_{t+1}, b_{t+1}) = \mathbb{E}_{b_{t+1}} [R(s_t, a_t, s_{t+1})] \quad (37.5)$$

For small problems, we can solve the resulting augmented MDP optimally. However, in general this is computationally intractable. [Gha+15] surveys many methods to solve it more efficiently. For example, [KN09] develop an algorithm that behaves similarly to Bayes optimal policies, except in a provably small number of steps; [GSD13] propose an approximate method based on Monte Carlo rollouts. More recently, [Zin+20] propose an approximate method based on meta-learning (Section 20.6), in which they train a (model-free) policy for multiple related tasks. Each task is represented by a task embedding vector  $m$ , which is inferred from  $\mathbf{h}_t$  using a VAE (Section 22.2). The posterior  $p(m | \mathbf{h}_t)$  is used as a proxy for the belief state  $b_t$ , and the policy is trained to perform well given  $s_t$  and  $b_t$ . At test time, the policy is applied to the incrementally computed belief state; this allows the method to infer what kind of task this is, and then to use a pre-trained policy to quickly solve it.

## 37.2 Value-based RL

In this section, we assume the agent has access to samples from  $p_T$  and  $p_R$  by interacting with the environment. We will show how to use these samples to learn optimal  $Q$ -functions from which we can derive optimal policies.

1  
2 **37.2.1 Monte Carlo RL**

3 Recall that  $Q_\pi(s, a) = \mathbb{E}[G_t | s_t = s, a_t = a]$  for any  $t$ . A simple way to estimate this is to take action  
4  $a$ , and then sample the rest of the trajectory according to  $\pi$ , and then compute the average sum of  
5 discounted rewards. The trajectory ends when we reach a terminal state, if the task is episodic, or  
6 when the discount factor  $\gamma^t$  becomes negligibly small, whichever occurs first. This is the **Monte  
7 Carlo estimation** of the value function.

8 We can use this technique together with policy iteration (Section 36.6.2) to learn an optimal policy.  
9 Specifically, at iteration  $k$ , we compute a new, improved policy using  $\pi_{k+1}(s) = \operatorname{argmax}_a Q_k(s, a)$ ,  
10 where  $Q_k$  is approximated using MC estimation. This update can be applied to all the states visited  
11 on the sampled trajectory. This overall technique is called **Monte Carlo control**.

12 To ensure this method converges to the optimal policy, we need to collect data for every (state,  
13 action) pair, at least in the tabular case, since there is no generalization across different values of  
14  $Q(s, a)$ . One way to achieve this is to use an  $\epsilon$ -greedy policy. Since this is an on-policy algorithm,  
15 the resulting method will converge to the optimal  $\epsilon$ -soft policy, as opposed to the optimal policy. It  
16 is possible to use importance sampling to estimate the value function for the optimal policy, even if  
17 actions are chosen according to the  $\epsilon$ -greedy policy. However, it is simpler to just gradually reduce  $\epsilon$ .  
18

19  
20 **37.2.2 Temporal difference (TD) learning**

21 The Monte Carlo (MC) method in Section 37.2.1 results in an estimator for  $Q_\pi(s, a)$  with very high  
22 variance, since it has to unroll many trajectories, whose returns are a sum of many random rewards  
23 generated by stochastic state transitions. In addition, it is limited to episodic tasks (or finite horizon  
24 truncation of continuing tasks), since it must unroll to the end of the episode before each update  
25 step, to ensure it reliably estimates the long term return.

26 In this section, we discuss a more efficient technique called **temporal difference** or **TD** learning  
27 [Sut88]. The basic idea is to incrementally reduce the Bellman error for sampled states or state-actions,  
28 based on transitions instead of a long trajectory. More precisely, suppose we are to learn the value  
29 function  $V_\pi$  for a fixed policy  $\pi$ . Given a state transition  $(s, a, r, s')$  where  $a \sim \pi(s)$ , we change the  
30 estimate  $V(s)$  so that it moves toward the bootstrapping target (Section 37.1.2)

31  
32 
$$V(s_t) \leftarrow V(s_t) + \eta [r_t + \gamma V(s_{t+1}) - V(s_t)] \quad (37.6)$$

33 where  $\eta$  is the learning rate. The term multiplied by  $\eta$  above is known as the **TD error**. A more  
34 general form of TD update for parametric value function representations is  
35

36  
37 
$$\mathbf{w} \leftarrow \mathbf{w} + \eta [r_t + \gamma V_{\mathbf{w}}(s_{t+1}) - V_{\mathbf{w}}(s_t)] \nabla_{\mathbf{w}} V_{\mathbf{w}}(s_t) \quad (37.7)$$

38 of which Equation (37.6) is a special case. The TD update rule for learning  $Q_\pi$  is similar.

39 It can be shown that TD learning in the tabular case, Equation (37.6), converges to the correct  
40 value function, under proper conditions [Ber19]. However, it may diverge when approximation is  
41 used (Equation (37.7)), an issue we will discuss further in Section 37.5.3.

42 The potential divergence of TD is also consistent with the fact that Equation (37.7) is not SGD  
43 (Section 6.3) on any objective function, despite a very similar form. Instead, it is an example of  
44 **bootstrapping**, in which the estimate,  $V_{\mathbf{w}}(s_t)$ , is updated to approach a target,  $r_t + \gamma V_{\mathbf{w}}(s_{t+1})$ ,  
45 which is defined by the value function estimate itself. This idea is shared by DP methods like value  
46 iteration, although they rely on the complete MDP model to compute an exact Bellman backup. In  
47

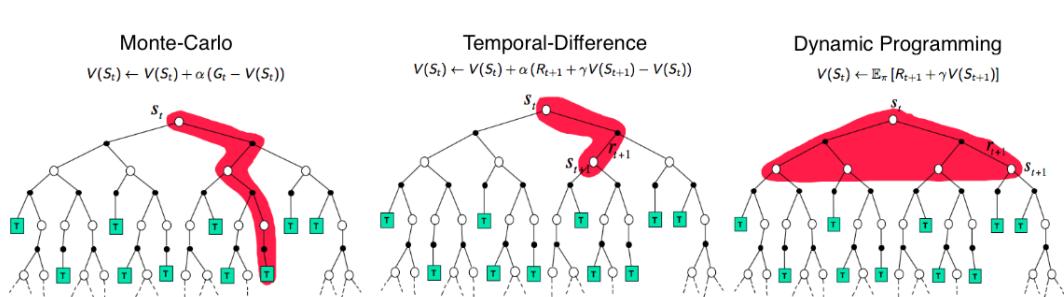


Figure 37.1: Backup diagrams of  $V(s_t)$  for Monte Carlo, temporal difference, and dynamic programming updates of the state-value function. Used with kind permission of Andy Barto.

contrast, TD learning can be viewed as using sampled transitions to approximate such backups. An example of non-bootstrapping approach is the Monte Carlo estimation in the previous section. It samples a complete trajectory, rather than individual transitions, to perform an update, and is often much less efficient. Figure 37.1 illustrates the difference between MC, TD, and DP.

### 37.2.3 TD learning with eligibility traces

A key difference between TD and MC is the way they estimate returns. Given a trajectory  $\tau = (s_0, a_0, r_0, s_1, \dots, s_T)$ , TD estimates the return from state  $s_t$  by one-step lookahead,  $G_{t:t+1} = r_t + \gamma V(s_{t+1})$ , where the return from time  $t+1$  is replaced by its value function estimate. In contrast, MC waits until the end of the episode or until  $T$  is large enough, then uses the estimate  $G_{t:T} = r_t + \gamma r_{t+1} + \dots + \gamma^{T-t-1} r_{T-1}$ . It is possible to interpolate between these by performing an  $n$ -step rollout, and then using the value function to approximate the return for the rest of the trajectory, similar to heuristic search (Section 37.4.1.1). That is, we can use the  $n$ -step estimate

$$G_{t:t+n} = r_t + \gamma r_{t+1} + \dots + \gamma^{n-1} r_{t+n-1} + \gamma^n V(s_{t+n}) \quad (37.8)$$

The corresponding  $n$ -step version of the TD update becomes

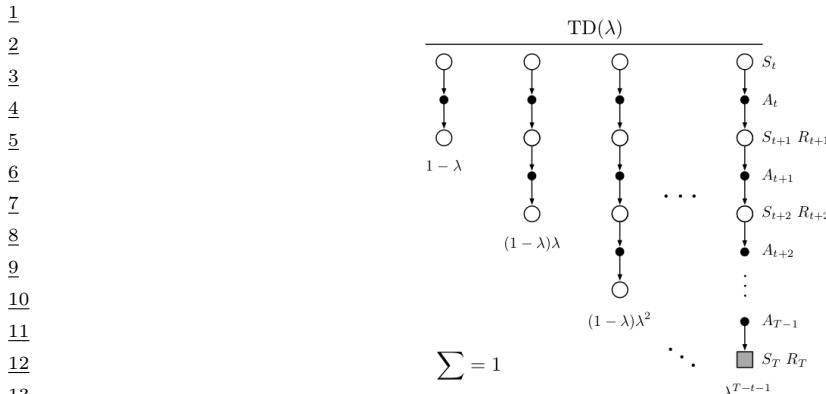
$$V(s_t) \leftarrow V(s_t) + \eta [G_{t:t+n} - V(s_t)] \quad (37.9)$$

Rather than picking a specific lookahead value,  $n$ , we can take a weighted average of all possible values, with a single parameter  $\lambda \in [0, 1]$ , by using

$$G_t^\lambda \triangleq (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n} \quad (37.10)$$

This is called the  **$\lambda$ -return**. The coefficient of  $1 - \lambda = (1 + \lambda + \lambda^2 + \dots)^{-1}$  in the front ensures this is a convex combination of  $n$ -step returns. See Figure 37.2 for an illustration.

An important benefit of using the geometric weighting in Equation (37.10) is that the corresponding TD learning update can be efficiently implemented, through the use of **eligibility traces**, even though  $G_t^\lambda$  is a sum of infinitely many terms. The method is called TD( $\lambda$ ), and can be combined with many algorithms to be studied in the rest of the chapter. See [SB18] for a detailed discussion.



*Figure 37.2: The backup diagram for TD( $\lambda$ ). Standard TD learning corresponds to  $\lambda = 0$ , and standard MC learning corresponds to  $\lambda = 1$ . From Figure 12.1 of [SB18]. Used with kind permission of Richard Sutton.*

### 37.2.4 SARSA: on-policy TD control

TD learning is for policy evaluation, as it estimates the value function for a fixed policy. In order to find an optimal policy, we may use the algorithm as a building block inside generalized policy iteration (Section 36.2). In this case, it is more convenient to work with the action-value function,  $Q$ , and a policy  $\pi$  that is greedy with respect to  $Q$ . The agent follows  $\pi$  in every step to choose actions, and upon a transition  $(s, a, r, s')$  the TD update rule is

$$Q(s, a) \leftarrow Q(s, a) + \eta [r + \gamma Q(s', a') - Q(s, a)] \quad (37.11)$$

where  $a' \sim \pi(s')$  is the action the agent will take in state  $s'$ . After  $Q$  is updated (for policy evaluation),  $\pi$  also changes accordingly as it is greedy with respect to  $Q$  (for policy improvement). This algorithm, first proposed by [RN94], was further studied and renamed to **SARSA** by [Sut96]; the name comes from its update rule that involves an augmented transition  $(s, a, r, s', a')$ .

In order for SARSA to converge to  $Q_*$ , every state-action pair must be visited infinitely often, at least in the tabular case, since the algorithm only updates  $Q(s, a)$  for  $(s, a)$  that it visits. One way to ensure this condition is to use a “greedy in the limit with infinite exploration” (**GLIE**) policy. An example is the  $\epsilon$ -greedy policy, with  $\epsilon$  vanishing to 0 gradually. It can be shown that SARSA with a GLIE policy will converge to  $Q_*$  and  $\pi_*$  [Sin+00].

39

### 37.2.5 Q-learning: off-policy TD control

40 SARSA is an **on-policy** algorithm, which means it learns the  $Q$ -function for the policy it is currently using, which is typically not the optimal policy (except in the limit for a GLIE policy). However, 41 with a simple modification, we can convert this to an **off-policy** algorithm that learns  $Q_*$ , even if a 42 suboptimal policy is used to choose actions.

43 The idea is to replace the sampled next action  $a' \sim \pi(s')$  in Equation (37.11) with a greedy action 44

in  $s'$ :  $a' = \operatorname{argmax}_b Q(s', b)$ . This results in the following update when a transition  $(s, a, r, s')$  happens

$$Q(s, a) \leftarrow Q(s, a) + \eta \left[ r + \gamma \max_b Q(s', b) - Q(s, a) \right] \quad (37.12)$$

This is the update rule of **Q-learning** for the tabular case [WD92]. The extension to work with function approximation can be done in a way similar to Equation (37.7). Since it is off-policy, the method can use  $(s, a, r, s')$  triples coming from any data source, such as older versions of the policy, or log data from an existing (non-RL) system. If every state-action pair is visited infinitely often, the algorithm provably converges to  $Q_*$  in the tabular case, with properly decayed learning rates [Ber19]. Algorithm 35 gives a vanilla implementation of Q-learning with  $\epsilon$ -greedy exploration.

---

**Algorithm 35:** Q-learning with  $\epsilon$ -greedy exploration

---

```

1 Initialize value function parameters  $w$ 
2 repeat
3   Sample starting state  $s$  of new episode
4   repeat
5     Sample action  $a = \begin{cases} \operatorname{argmax}_b Q_w(s, b), & \text{with probability } 1 - \epsilon \\ \text{random action}, & \text{with probability } \epsilon \end{cases}$ 
6     Observe state  $s'$ , reward  $r$ 
7     Compute the TD error:  $\delta = r + \gamma \max_{a'} Q_w(s', a') - Q_w(s, a)$ 
8      $w \leftarrow w + \eta \delta \nabla_w Q_w(s, a)$ 
9      $s \leftarrow s'$ 
10    until state  $s$  is terminal;
11 until converged;

```

---

### 37.2.5.1 Example

Figure 37.3 gives an example of Q-learning applied to the simple 1d grid world from Figure 36.10, using  $\gamma = 0.9$ . We show the  $Q$ -function at the start and end of each episode, after performing actions chosen by an  $\epsilon$ -greedy policy. We initialize  $Q(s, a) = 0$  for all entries, and use a step size of  $\eta = 1$ , so the update becomes  $Q_*(s, a) = r + \gamma Q_*(s', a_*)$ , where  $a_* = \downarrow$  for all states.

### 37.2.5.2 Double Q-learning

Standard Q-learning suffers from a problem known as the **optimizer's curse** [SW06], or the **maximization bias**. The problem refers to the simple statistical inequality,  $\mathbb{E}[\max_a X_a] \geq \max_a \mathbb{E}[X_a]$ , for a set of random variables  $\{X_a\}$ . Thus, if we pick actions greedily according to their random scores  $\{X_a\}$ , we might pick a wrong action just because random noise makes it appealing.

Figure 37.4 gives a simple example of how this can happen in an MDP. The start state is A. The right action gives a reward 0 and terminates the episode. The left action also gives a reward of 0, but then enters state B, from which there are many possible actions, with rewards drawn from  $\mathcal{N}(-0.1, 1.0)$ . Thus the expected return for any trajectory starting with the left action is

1	Q-function episode start	Episode	Time Step	Action	$(s, a, r, s')$	$r + \gamma Q^*(s', a)$	Q-function episode end
2	$Q_1$	UP	DOWN	1	$\downarrow (S_1, D, 0, S_2)$	$0 + 0.9 \times 0 = 0$	$S_1$
3		$S_1$	0	0	$\downarrow (S_2, U, 0, S_1)$	$0 + 0.9 \times 0 = 0$	$S_2$
4		$S_2$	0	0	$\downarrow (S_1, D, 0, S_2)$	$0 + 0.9 \times 0 = 0$	$S_1$
5		$S_1$	0	0	$\downarrow (S_2, U, 0, S_1)$	$0 + 0.9 \times 0 = 0$	$S_2$
6		$S_2$	0	0	$\downarrow (S_3, D, 1, S_{T2})$	1	$S_3$
7	$Q_2$	UP	DOWN	1	$\downarrow (S_1, D, 0, S_2)$	$0 + 0.9 \times 0 = 0$	$S_1$
8		$S_1$	0	0	$\downarrow (S_2, D, 0, S_3)$	$0 + 0.9 \times 1 = 0.9$	$S_2$
9		$S_2$	0	0	$\downarrow (S_3, D, 0, S_{T2})$	1	$S_3$
10		$S_1$	0	1			
11		UP	DOWN	2	$\downarrow (S_1, D, 0, S_2)$	$0 + 0.9 \times 0 = 0$	$S_1$
12	$Q_3$	$S_1$	0	0	$\downarrow (S_2, D, 0, S_3)$	$0 + 0.9 \times 0.81 = 0.81$	$S_2$
13		$S_2$	0	0.9	$\uparrow (S_3, D, 0, S_2)$	$0 + 0.9 \times 1 = 0.9$	$S_3$
14		$S_1$	0	0	$\downarrow (S_2, D, 0, S_3)$	$0 + 0.9 \times 0.81 = 0.81$	$S_2$
15		$S_2$	0	0.9	$\downarrow (S_3, D, 0, S_{T2})$	1	$S_3$
16		$S_1$	0	1			
17	$Q_4$	UP	DOWN	3	$\downarrow (S_1, D, 0, S_2)$	$0 + 0.9 \times 0.81 = 0.81$	$S_1$
18		$S_1$	0	0.81	$\downarrow (S_2, U, 0, S_3)$	$0 + 0.9 \times 0.81 = 0.73$	$S_2$
19		$S_2$	0	0.9	$\downarrow (S_3, D, 0, S_2)$	$0 + 0.9 \times 0.81 = 0.81$	$S_3$
20		$S_1$	0.81	1	$\uparrow (S_2, U, 0, S_3)$	$0 + 0.9 \times 0.81 = 0.73$	$S_2$
21		$S_1$	0	0.81	$\downarrow (S_3, D, 0, S_2)$	$0 + 0.9 \times 0.81 = 0.81$	$S_3$
22	$Q_5$	UP	DOWN	4	$\downarrow (S_1, D, 0, S_2)$	$0 + 0.9 \times 0.81 = 0.81$	$S_1$
23		$S_1$	0	0.81	$\uparrow (S_2, U, 0, S_3)$	$0 + 0.9 \times 0.81 = 0.73$	$S_2$
24		$S_2$	0.73	0.9	$\downarrow (S_3, D, 0, S_2)$	$0 + 0.9 \times 0.81 = 0.81$	$S_3$
25		$S_1$	0.81	1	$\uparrow (S_2, U, 0, S_3)$	$0 + 0.9 \times 0.81 = 0.73$	$S_2$
26		$S_1$	0	0.81	$\downarrow (S_3, D, 0, S_2)$	1	$S_3$
27							

Figure 37.3: Illustration of  $Q$  learning for the 1d grid world in Figure 36.10 using  $\epsilon$ -greedy exploration. At the end of episode 1, we make a transition from  $S_3$  to  $S_{T2}$  and get a reward of  $r = 1$ , so we estimate  $Q(S_3, \downarrow) = 1$ . In episode 2, we make a transition from  $S_2$  to  $S_3$ , so  $S_2$  gets incremented by  $\gamma Q(S_3, \downarrow) = 0.9$ . Adapted from Figure 3.3 of [GK19].

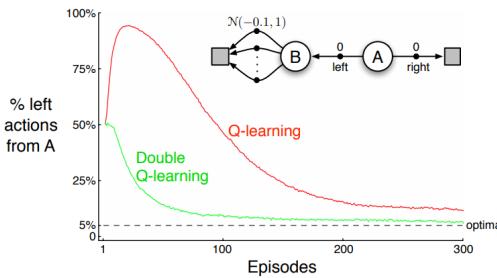


Figure 37.4: Comparison of  $Q$ -learning and double  $Q$ -learning on a simple episodic MDP using  $\epsilon$ -greedy action selection with  $\epsilon = 0.1$ . The initial state is  $A$ , and squares denote absorbing states. The data are averaged over 10,000 runs. From Figure 6.5 of [SB18]. Used with kind permission of Rich Sutton.

46  
47

1 –0.1, making it suboptimal. Nevertheless, the RL algorithm may pick the left action due to the  
2 maximization bias making B appear to have a positive value.  
3

4 One solution to avoid the maximization bias is to use two separate  $Q$ -functions,  $Q_1$  and  $Q_2$ , one  
5 for selecting the greedy action, and the other for estimating the corresponding  $Q$ -value. In particular,  
6 upon seeing a transition  $(s, a, r, s')$ , we perform the following update  
7

$$\underline{8} \quad Q_1(s, a) \leftarrow Q_1(s, a) + \eta \left[ r + \gamma Q_2\left(s', \operatorname{argmax}_{a'} Q_1(s', a')\right) - Q_1(s, a) \right] \quad (37.13)$$

9 and may repeat the same update but with the roles of  $Q_1$  and  $Q_2$  swapped. This technique is  
10 called **double Q-learning** [Has10]. Figure 37.4 shows the benefits of the algorithm over standard  
11 Q-learning in a toy problem.  
12

### 13 37.2.6 Deep Q-network (DQN)

14 When function approximation is used, Q-learning may be hard to use in practice due to instability  
15 problems. Here, we will describe two important heuristics, popularized by the **deep Q-network** or  
16 **DQN** work [Mni+15], which was able to train agents to outperform humans at playing Atari games,  
17 using CNN-structured  $Q$ -networks.  
18

19 The first technique, originally proposed in [Lin92], is to leverage an **experience replace** buffer,  
20 which stores the most recent  $(s, a, r, s')$  transition tuples. In contrast to standard Q-learning which  
21 updates the  $Q$ -function when a new transition occurs, the DQN agent also performs additional  
22 updates using transitions sampled from the buffer. This modification has two advantages. First, it  
23 improves data efficiency as every transition can be used multiple times. Second, it improves stability  
24 in training, by reducing the correlation of the data samples that the network is trained on.  
25

26 The second idea to improve stability is to regress the  $Q$ -network to a “frozen” **target network**  
27 computed at an earlier iteration, rather than trying to chase a constantly moving target. Specifically,  
28 we maintain an extra, frozen copy of the  $Q$ -network,  $Q_{\mathbf{w}^-}$ , of the same structure as  $Q_{\mathbf{w}}$ . This new  
29  $Q$ -network is to compute bootstrapping targets for training  $Q_{\mathbf{w}}$ , in which the loss function is  
30

$$\underline{31} \quad \mathcal{L}^{\text{DQN}}(\mathbf{w}) = \mathbb{E}_{(s, a, r, s') \sim U(\mathcal{D})} \left[ (r + \gamma \max_{a'} Q_{\mathbf{w}^-}(s', a') - Q_{\mathbf{w}}(s, a))^2 \right] \quad (37.14)$$

32 where  $U(\mathcal{D})$  is a uniform distribution over the replay buffer  $\mathcal{D}$ . We then periodically set  $\mathbf{w}^- \leftarrow \mathbf{w}$ ,  
33 usually after a few episodes. This approach is an instance of **fitted value iteration** [SB18].  
34

35 Various improvements to DQN have been proposed. One is **double DQN** [HGS16], which uses  
36 the double learning technique (Section 37.2.5.2) to remove the maximization bias. The second is to  
37 replace the uniform distribution in Equation (37.14) with one that favors more important transition  
38 tuples, resulting in the use of **prioritized experience replay** [Sch+16a]. For example, we can  
39 sample transitions from  $\mathcal{D}$  with probability  $p(s, a, r, s') \propto (|\delta| + \varepsilon)^\eta$ , where  $\delta$  is the corresponding  
40 TD error (under the current  $Q$ -function),  $\varepsilon > 0$  a hyperparameter to ensure every experience is  
41 chosen with nonzero probability, and  $\eta \geq 0$  controls the “inverse temperature” of the distribution  
42 (so  $\eta = 0$  corresponds to uniform sampling). The third is to learn a value function  $V_{\mathbf{w}}$  and an  
43 advantage function  $A_{\mathbf{w}}$ , with shared parameter  $\mathbf{w}$ , instead of learning  $Q_{\mathbf{w}}$ . The resulting **dueling**  
44 **DQN** [Wan+16] is shown to be more sample efficient, especially when there are many actions with  
45 similar  $Q$ -values.  
46

<sup>1</sup> The **rainbow** method [Hes+18] combines all three improvements, as well as others, including  
<sup>2</sup> multi-step returns (Section 37.2.3), distributional RL [BDM17] (which predicts the distribution  
<sup>3</sup> of returns, not just the expected return), and noisy nets [For+18b] (which adds random noise to  
<sup>4</sup> the network weights to encourage exploration). It produces state-of-the-art results on the Atari  
<sup>5</sup> benchmark.  
<sup>6</sup>

<sup>7</sup>

### <sup>8</sup> 37.3 Policy-based RL

<sup>9</sup>  
<sup>10</sup> In the previous section, we considered methods that estimate the action-value function,  $Q(s, a)$ , from  
<sup>11</sup> which we derive a policy, which may be greedy or softmax. However, these methods have three main  
<sup>12</sup> disadvantages: (1) they can be difficult to apply to continuous action spaces; (2) they may diverge if  
<sup>13</sup> function approximation is used; and (3) the training of  $Q$ , often based on TD-style updates, is not  
<sup>14</sup> directly related to the expected return garnered by the learned policy.

<sup>15</sup> In this section, we discuss **policy search** methods, which directly optimize the parameters of the  
<sup>16</sup> policy so as to maximize its expected return. However, we will see that these methods often benefit  
<sup>17</sup> from estimating a value or advantage function to reduce the variance in the policy search process.  
<sup>18</sup>

#### <sup>19</sup> 37.3.1 The policy gradient theorem

<sup>20</sup>  
<sup>21</sup> We start by defining the objective function for policy learning, and then derive its gradient. We  
<sup>22</sup> consider the episodic case. A similar result can be derived for the continuing case with the average  
<sup>23</sup> reward criterion [SB18, Sec 13.6].

<sup>24</sup> We define the objective to be the expected return of a policy, which we aim to maximize:

$$\begin{aligned} \text{25} \quad J(\pi) &\triangleq \mathbb{E}_{p_0, \pi}[G_0] = \mathbb{E}_{p_0(s_0)}[V_\pi(s_0)] = \mathbb{E}_{p_0(s_0)\pi(a_0|s_0)}[Q_\pi(s_0, a_0)] \end{aligned} \quad (37.15)$$

<sup>26</sup> We consider policies  $\pi_\theta$  parameterized by  $\theta$ , and compute the gradient of Equation (37.15) wrt  $\theta$ :

$$\begin{aligned} \text{27} \quad \nabla_\theta J(\pi_\theta) &= \mathbb{E}_{p_0(s_0)} \left[ \nabla_\theta \left( \sum_{a_0} \pi_\theta(a_0|s_0) Q_{\pi_\theta}(s_0, a_0) \right) \right] \\ \text{28} \quad &= \mathbb{E}_{p_0(s_0)} \left[ \sum_{a_0} \nabla_\theta \pi_\theta(a_0|s_0) Q_{\pi_\theta}(s_0, a_0) \right] + \mathbb{E}_{p_0(s_0)\pi_\theta(a_0|s_0)}[\nabla_\theta Q_{\pi_\theta}(s_0, a_0)] \end{aligned} \quad (37.16)$$

$$\begin{aligned} \text{29} \quad &= \mathbb{E}_{p_0(s_0)} \left[ \sum_{a_0} \nabla_\theta \pi_\theta(a_0|s_0) Q_{\pi_\theta}(s_0, a_0) \right] + \mathbb{E}_{p_0(s_0)\pi_\theta(a_0|s_0)}[\nabla_\theta Q_{\pi_\theta}(s_0, a_0)] \end{aligned} \quad (37.17)$$

<sup>30</sup> Now we calculate the term  $\nabla_\theta Q_{\pi_\theta}(s_0, a_0)$ :

$$\begin{aligned} \text{31} \quad \nabla_\theta Q_{\pi_\theta}(s_0, a_0) &= \nabla_\theta [R(s_0, a_0) + \gamma \mathbb{E}_{p_T(s_1|s_0, a_0)}[V_{\pi_\theta}(s_1)]] = \gamma \nabla_\theta \mathbb{E}_{p_T(s_1|s_0, a_0)}[V_{\pi_\theta}(s_1)] \end{aligned} \quad (37.18)$$

<sup>32</sup> The right-hand side above is in a form similar to  $\nabla_\theta J(\pi_\theta)$ . Repeating the same steps as before gives

$$\begin{aligned} \text{33} \quad \nabla_\theta J(\pi_\theta) &= \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{p_t(s)} \left[ \sum_a \nabla_\theta \pi_\theta(a|s) Q_{\pi_\theta}(s, a) \right] \\ \text{34} \quad &= \frac{1}{1-\gamma} \mathbb{E}_{p_{\pi_\theta}^\infty(s)} \left[ \sum_a \nabla_\theta \pi_\theta(a|s) Q_{\pi_\theta}(s, a) \right] \end{aligned} \quad (37.19)$$

$$\begin{aligned} \text{35} \quad &= \frac{1}{1-\gamma} \mathbb{E}_{p_{\pi_\theta}^\infty(s)\pi_\theta(a|s)}[\nabla_\theta \log \pi_\theta(a|s) Q_{\pi_\theta}(s, a)] \end{aligned} \quad (37.20)$$

$$\begin{aligned} \text{36} \quad &= \frac{1}{1-\gamma} \mathbb{E}_{p_{\pi_\theta}^\infty(s)\pi_\theta(a|s)}[\nabla_\theta \log \pi_\theta(a|s) Q_{\pi_\theta}(s, a)] \end{aligned} \quad (37.21)$$

<sup>37</sup>

where  $p_t(s)$  is the probability of visiting  $s$  in time  $t$  if we start with  $s_0 \sim p_0$  and follow  $\pi_\theta$ , and  $p_{\pi_\theta}^\infty(s) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t p_t(s)$  is the normalized discounted state visitation distribution. Equation (37.21) is known as the **policy gradient theorem** [Sut+99].

In practice, estimating the policy gradient using Equation (37.21) can have a high variance. A baseline  $b(s)$  can be used for variance reduction (Section 6.6.3.1):

$$\nabla_\theta J(\pi_\theta) = \frac{1}{1 - \gamma} \mathbb{E}_{p_{\pi_\theta}^\infty(s)\pi_\theta(a|s)} [\nabla_\theta \log \pi_\theta(a|s)(Q_{\pi_\theta}(s, a) - b(s))] \quad (37.22)$$

A common choice for the baseline is  $b(s) = V_{\pi_\theta}(s)$ . We will discuss how to estimate it below.

### 37.3.2 REINFORCE

One way to apply the policy gradient theorem to optimize a policy is to use stochastic gradient ascent. Suppose  $\tau = (s_0, a_0, r_0, s_1, \dots, s_T)$  is a trajectory with  $s_0 \sim p_0$  and  $\pi_\theta$ . Then,

$$\nabla_\theta J(\pi_\theta) = \frac{1}{1 - \gamma} \mathbb{E}_{p_{\pi_\theta}^\infty(s)\pi_\theta(a|s)} [\nabla_\theta \log \pi_\theta(a|s) Q_{\pi_\theta}(s, a)] \quad (37.23)$$

$$\approx \sum_{t=0}^{T-1} \gamma^t G_t \nabla_\theta \log \pi_\theta(a_t|s_t) \quad (37.24)$$

where the return  $G_t$  is defined in Equation (36.52), and the factor  $\gamma^t$  is due to the definition of  $p_{\pi_\theta}^\infty$  where the state at time  $t$  is discounted.

We can use a baseline in the gradient estimate to get the following update rule:

$$\theta \leftarrow \theta + \eta \sum_{t=0}^{T-1} \gamma^t (G_t - b(s_t)) \nabla_\theta \log \pi_\theta(a_t|s_t) \quad (37.25)$$

This is called the **REINFORCE** algorithm [Wil92].<sup>3</sup> The update equation can be interpreted as follows: we compute the sum of discounted future rewards induced by a trajectory, compared to a baseline, and if this is positive, we increase  $\theta$  so as to make this trajectory more likely, otherwise we decrease  $\theta$ . Thus, we reinforce good behaviors, and reduce the chances of generating bad ones.

We can use a constant (state-independent) baseline, or we can use a state-dependent baseline,  $b(s_t)$  to further lower the variance. A natural choice is to use an estimated value function,  $V_w(s)$ , which can be learned, e.g., with MC. Algorithm 36 gives the pseudo code where stochastic gradient updates are used with separate learning rates.

### 37.3.3 Actor-critic methods

An **actor-critic** method [BSA83] uses the policy gradient method, but where the expected return is estimated using temporal difference learning of a value function instead of MC rollouts. The term “actor” refers to the policy, and the term “critic” refers to the value function. The use of bootstrapping

<sup>3</sup> The term “REINFORCE” is an acronym for “REward Increment = nonnegative Factor x Offset Reinforcement x Characteristic Eligibility”. The phrase “Characteristic eligibility” refers to the  $\nabla \log \pi_\theta(a_t|s_t)$  term; the phrase “offset reinforcement” refers to the  $G_t - b(s_t)$  term; and the phrase “nonnegative factor” refers to the learning rate  $\eta$  of SGD.

---

1  
2   **Algorithm 36:** REINFORCE with value function baseline  
3   1 Initialize policy parameters  $\boldsymbol{\theta}$ , baseline parameters  $\mathbf{w}$   
4   2 **repeat**  
5   3    Sample an episode  $\tau = (s_0, a_0, r_0, s_1, \dots, s_T)$  using  $\pi_{\boldsymbol{\theta}}$   
6   4    Compute  $G_t$  for all  $t \in \{0, 1, \dots, T-1\}$  using Equation (36.52)  
7   5    **for**  $t = 0, 1, \dots, T-1$  **do**  
8   6       $\delta = G_t - V_{\mathbf{w}}(s_t)$  // scalar error  
9   7       $\mathbf{w} \leftarrow \mathbf{w} + \eta_{\mathbf{w}} \delta \nabla_{\mathbf{w}} V_{\mathbf{w}}(s_t)$   
10   8       $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_{\boldsymbol{\theta}} \gamma^t \delta \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t | s_t)$   
11  
12   9 **until** converged;  
13  
14

---

15 in TD updates allows more efficient learning of the value function compared to MC. In addition, it  
16 allows us to develop a fully online, incremental algorithm, that does not need to wait until the end of  
17 the trajectory before updating the parameters (as in Algorithm 36).

18   Concretely, consider the use of the one-step TD(0) method to estimate the return in the episodic  
19 case, i.e., we replace  $G_t$  with  $G_{t:t+1} = r_t + \gamma V_{\mathbf{w}}(s_{t+1})$ . If we use  $V_{\mathbf{w}}(s_t)$  as a baseline, the REINFORCE  
20 update in Equation (37.25) becomes

21  
22   
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \sum_{t=0}^{T-1} \gamma^t (G_{t:t+1} - V_{\mathbf{w}}(s_t)) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t | s_t) \quad (37.26)$$

23  
24  
25   
$$= \boldsymbol{\theta} + \eta \sum_{t=0}^{T-1} \gamma^t (r_t + \gamma V_{\mathbf{w}}(s_{t+1}) - V_{\mathbf{w}}(s_t)) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t | s_t) \quad (37.27)$$

26   **37.3.3.1 A2C and A3C**

27 Note that  $r_{t+1} + \gamma V_{\mathbf{w}}(s_{t+1}) - V_{\mathbf{w}}(s_t)$  is a single sample approximation to the advantage function  
28  $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$ . This method is therefore called **advantage actor critic** or **A2C**  
29 (Algorithm 37). If we run the actors in parallel and asynchronously update their shared parameters,  
30 the method is called **asynchronous advantage actor critic** or **A3C** [Mni+16].

31

32   **37.3.3.2 Eligibility traces**

33

34   In A2C, we use a single step rollout, and then use the value function in order to approximate the  
35 expected return for the trajectory. More generally, we can use the  $n$ -step estimate

36  
37   
$$G_{t:t+n} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots + \gamma^{n-1} r_{t+n-1} + \gamma^n V_{\mathbf{w}}(s_{t+n}) \quad (37.28)$$

38 and obtain an  $n$ -step advantage estimate as follows:

39  
40   
$$A_{\pi_{\boldsymbol{\theta}}}^{(n)}(s_t, a_t) = G_{t:t+n} - V_{\mathbf{w}}(s_t) \quad (37.29)$$

41   The  $n$  steps of actual rewards are an unbiased sample, but have high variance. By contrast,  
42  $V_{\mathbf{w}}(s_{t+n+1})$  has lower variance, but is biased. By changing  $n$ , we can control the bias-variance  
43

---

1  
2   **Algorithm 37:** Advantage actor critic (A2C) algorithm  
3   1 Initialize actor parameters  $\theta$ , critic parameters  $w$   
4   2 **repeat**  
5   3    Sample starting state  $s_0$  of a new episode  
6   4    **for**  $t = 0, 1, 2, \dots$  **do**  
7   5    | Sample action  $a_t \sim \pi_\theta(\cdot | s_t)$   
8   6    | Observe next state  $s_{t+1}$  and reward  $r_t$   
9   7    |  $\delta = r_t + \gamma V_w(s_{t+1}) - V_w(s_t)$   
10   8    |  $w \leftarrow w + \eta_w \delta \nabla_w V_w(s_t)$   
11   9    |  $\theta \leftarrow \theta + \eta_\theta \gamma^t \delta \nabla_\theta \log \pi_\theta(a_t | s_t)$   
12  
13   10 **until** converged;

---

14  
15  
16  
17 tradeoff. Instead of using a single value of  $n$ , we can take an weighted average, with weight  
18 proportional to  $\lambda^n$  for  $A_{\pi_\theta}^{(n)}(s_t, a_t)$ , as in TD( $\lambda$ ). The average can be shown to be equivalent to  
19

20  
21   
$$A_{\pi_\theta}^{(\lambda)}(s_t, a_t) \triangleq \sum_{\ell=0}^{\infty} (\gamma \lambda)^\ell \delta_{t+\ell} \quad (37.30)$$
  
22

23 where  $\delta_t = r_t + \gamma V_w(s_{t+1}) - V_w(s_t)$  is the TD error at time  $t$ . Here,  $\lambda \in [0, 1]$  is a parameter  
24 that controls the bias-variance tradeoff: larger values decrease the bias but increase the variance,  
25 as in TD( $\lambda$ ). We can implement Equation (37.30) efficiently using eligibility traces, as shown in  
26 Algorithm 38, as an example of **generalized advantage estimation (GAE)** [Sch+16b]. See [SB18,  
27 Ch.12] for further details.  
28

---

29  
30   **Algorithm 38:** Actor critic with eligibility traces  
31

---

32   1 Initialize actor parameters  $\theta$ , critic parameters  $w$   
33   2 **repeat**  
34   3    Initialize eligibility trace vectors:  $z_\theta \leftarrow \mathbf{0}$ ,  $z_w \leftarrow \mathbf{0}$   
35   4    Sample starting state  $s_0$  of a new episode  
36   5    **for**  $t = 0, 1, 2, \dots$  **do**  
37   6    | Sample action  $a_t \sim \pi_\theta(\cdot | s_t)$   
38   7    | Observe state  $s_{t+1}$  and reward  $r_t$   
39   8    | Compute the TD error:  $\delta = r_t + \gamma V_w(s_{t+1}) - V_w(s_t)$   
40   9    |  $z_w \leftarrow \gamma \lambda_w z_w + \nabla_w V_w(s)$   
41   10   |  $z_\theta \leftarrow \gamma \lambda_\theta z_\theta + \gamma^t \nabla_\theta \log \pi_\theta(a_t | s_t)$   
42   11   |  $w \leftarrow w + \eta_w \delta z_w$   
43   12   |  $\theta \leftarrow \theta + \eta_\theta \delta z_\theta$   
44   13 **until** converged;

---

### <sup>1</sup> <sup>2</sup> 37.3.4 Bound optimization methods

<sup>3</sup> In policy gradient methods, the objective  $J(\theta)$  does not necessarily increase monotonically, but  
<sup>4</sup> rather can collapse especially if the learning rate is not small enough. We now describe methods that  
<sup>5</sup> guarantee monotonic improvement, similar to bound optimization algorithms (Section 6.7).

<sup>6</sup> We start with a useful fact that relate the policy values of two arbitrary policies [KL02]:  
<sup>7</sup>

$$\underline{\text{8}} \quad J(\pi') - J(\pi) = \frac{1}{1-\gamma} \mathbb{E}_{p_{\pi}^{\infty}(s)} [\mathbb{E}_{\pi'(a|s)} [A_{\pi}(s, a)]] \quad (\underline{\text{37.31}})$$

<sup>9</sup>

<sup>10</sup> where  $\pi$  can be interpreted as the current policy during policy optimization, and  $\pi'$  a candidate new  
<sup>11</sup> policy (such as the greedy policy wrt  $Q_{\pi}$ ). As in the policy improvement theorem (Section 36.6.2),  
<sup>12</sup> if  $\mathbb{E}_{\pi'(a|s)} [A_{\pi}(s, a)] \geq 0$  for all  $s$ , then  $J(\pi') \geq J(\pi)$ . However, we cannot ensure this condition to  
<sup>13</sup> hold when function approximation is used, as such a uniformly improving policy  $\pi'$  may not be  
<sup>14</sup> representable by our parametric family,  $\{\pi_{\theta}\}_{\theta \in \Theta}$ . Therefore, nonnegativity of Equation (37.31) is  
<sup>15</sup> not easy to ensure, when we do not have a direct way to sample states from  $p_{\pi'}^{\infty}$ .

<sup>16</sup> One way to ensure monotonic improvement of  $J$  is to improve the policy conservatively. Define  
<sup>17</sup>  $\pi_{\theta} = \theta\pi' + (1-\theta)\pi$  for  $\theta \in [0, 1]$ . It follows from the policy gradient theorem (Equation (37.21), with  
<sup>18</sup>  $\theta = [\theta]$ ) that  $J(\pi_{\theta}) - J(\pi) = \theta L(\pi') + O(\theta^2)$ , where

$$\underline{\text{19}} \quad L(\pi') \triangleq \frac{1}{1-\gamma} \mathbb{E}_{p_{\pi}^{\infty}(s)} [\mathbb{E}_{\pi'(a|s)} [A_{\pi}(s, a)]] = \frac{1}{1-\gamma} \mathbb{E}_{p_{\pi}^{\infty}(s) \pi(a|s)} \left[ \frac{\pi'(a|s)}{\pi(a|s)} A_{\pi}(s, a) \right] \quad (\underline{\text{37.32}})$$

<sup>20</sup>

<sup>21</sup> In the above, we have switched the state distribution from  $p_{\pi'}^{\infty}$  in Equation (37.31) to  $p_{\pi}^{\infty}$ , while at  
<sup>22</sup> the same time introducing a higher order residual term of  $O(\theta^2)$ . The linear term,  $\theta L(\pi')$ , can be  
<sup>23</sup> estimated and optimized based on episodes sampled by  $\pi$ . The higher order term can be bounded in  
<sup>24</sup> various ways, resulting in different lower bounds of  $J(\pi_{\theta}) - J(\pi)$ . We can then optimize  $\theta$  to make  
<sup>25</sup> sure this lower bound is positive, which would imply  $J(\pi_{\theta}) - J(\pi) > 0$ . In **conservative policy**  
<sup>26</sup> **iteration** [KL02], the following (slightly simplified) lower bound is used  
<sup>27</sup>

$$\underline{\text{28}} \quad J^{\text{CPI}}(\pi_{\theta}) \triangleq J(\pi) + \theta L(\pi') - \frac{2\varepsilon\gamma}{(1-\gamma)^2} \theta^2 \quad (\underline{\text{37.33}})$$

<sup>29</sup>

<sup>30</sup> where  $\varepsilon = \max_s |\mathbb{E}_{\pi'(a|s)} [A_{\pi}(s, a)]|$ .

<sup>31</sup> This idea can be generalized to policies beyond those in the form of  $\pi_{\theta}$ , where the condition  
<sup>32</sup> of a small enough  $\theta$  is replaced by a small enough divergence between  $\pi'$  and  $\pi$ . In **safe policy**  
<sup>33</sup> **iteration** [Pir+13], the divergence is the maximum total variation, while in **trust region policy**  
<sup>34</sup> **optimization (TRPO)** [Sch+15b], the divergence is the maximum KL-divergence. In the latter  
<sup>35</sup> case,  $\pi'$  may be found by optimizing the following lower bound  
<sup>36</sup>

$$\underline{\text{37}} \quad J^{\text{TRPO}}(\pi') \triangleq J(\pi) + L(\pi') - \frac{\varepsilon\gamma}{(1-\gamma)^2} \max_s D_{\text{KL}}(\pi(s) \| \pi'(s)) \quad (\underline{\text{37.34}})$$

<sup>38</sup>

<sup>39</sup> where  $\varepsilon = \max_{s,a} |A_{\pi}(s, a)|$ .

<sup>40</sup> In practice, the above update rule can be overly conservative, and approximations are used.  
<sup>41</sup> [Sch+15b] propose a version that implements two ideas: one is to replace the point-wise maximum  
<sup>42</sup> KL-divergence by some average KL-divergence (usually averaged over  $p_{\pi_{\theta}}^{\infty}$ ); the second is to maximize  
<sup>43</sup> the first two terms in Equation (37.34), with  $\pi'$  lying in a KL-ball centered at  $\pi$ . That is, we solve  
<sup>44</sup>

$$\underline{\text{45}} \quad \underset{\pi'}{\text{argmax}} L(\pi') \quad \text{s.t. } \mathbb{E}_{p_{\pi}^{\infty}(s)} [D_{\text{KL}}(\pi(s) \| \pi'(s))] \leq \delta \quad (\underline{\text{37.35}})$$

<sup>46</sup>

<sup>47</sup>

for some threshold  $\delta > 0$ .

In Section 6.4.2.1, we show that the trust region method, using a KL penalty at each step, is equivalent to natural gradient descent (see e.g., [Kak02; PS08b]). This is important, because a step of size  $\eta$  in parameter space does not always correspond to a step of size  $\eta$  in the policy space:

$$d_{\theta}(\theta_1, \theta_2) = d_{\theta}(\theta_2, \theta_3) \not\Rightarrow d_{\pi}(\pi_{\theta_1}, \pi_{\theta_2}) = d_{\pi}(\pi_{\theta_2}, \pi_{\theta_3}) \quad (37.36)$$

where  $d_{\theta}(\theta_1, \theta_2) = \|\theta_1 - \theta_2\|$  is the Euclidean distance, and  $d_{\pi}(\pi_1, \pi_2) = D_{\text{KL}}(\pi_1 \| \pi_2)$  the KL distance. In other words, the effect on the policy of any given change to the parameters depends on where we are in parameter space. This is taken into account by the natural gradient method, resulting in faster and more robust optimization. The natural policy gradient can be approximated using the KFAC method (Section 6.4.4), as done in [Wu+17].

Other than TRPO, another approach inspired by Equation (37.34) is to use the KL-divergence as a penalty term, replacing the factor  $2\varepsilon\gamma/(1-\gamma)^2$  by a tuning parameter. However, it often works better, and is simpler, by using the following clipped objective, which results in the **proximal policy optimization** or **PPO** method [Sch+17]:

$$J^{\text{PPO}}(\pi') \triangleq \frac{1}{1-\gamma} \mathbb{E}_{p_{\pi}^{\infty}(s)\pi(a|s)} \left[ \kappa_{\epsilon} \left( \frac{\pi'(a|s)}{\pi(a|s)} \right) A_{\pi}(s, a) \right] \quad (37.37)$$

where  $\kappa_{\epsilon}(x) \triangleq \text{clip}(x, 1-\epsilon, 1+\epsilon)$  ensures  $|\kappa(x) - 1| \leq \epsilon$ . This method can be modified to ensure monotonic improvement as discussed in [WHT19], making it a true bound optimization method.

### 37.3.5 Deterministic policy gradient methods

In this section, we consider the case of a deterministic policy, that predicts a unique action for each state, so  $a_t = \mu_{\theta}(s_t)$ , rather than  $a_t \sim \pi_{\theta}(s_t)$ . We assume the states and actions are continuous, and define the objective as

$$J(\mu_{\theta}) \triangleq \frac{1}{1-\gamma} \mathbb{E}_{p_{\mu_{\theta}}^{\infty}(s)} [R(s, \mu_{\theta}(s))] \quad (37.38)$$

The **deterministic policy gradient theorem** [Sil+14] provides a way to compute the gradient:

$$\nabla_{\theta} J(\mu_{\theta}) = \frac{1}{1-\gamma} \mathbb{E}_{p_{\mu_{\theta}}^{\infty}(s)} [\nabla_{\theta} Q_{\mu_{\theta}}(s, \mu_{\theta}(s))] \quad (37.39)$$

$$= \frac{1}{1-\gamma} \mathbb{E}_{p_{\mu_{\theta}}^{\infty}(s)} [\nabla_{\theta} \mu_{\theta}(s) \nabla_a Q_{\mu_{\theta}}(s, a)|_{a=\mu_{\theta}(s)}] \quad (37.40)$$

where  $\nabla_{\theta} \mu_{\theta}(s)$  is the  $M \times N$  Jacobian matrix, and  $M$  and  $N$  are the dimensions of  $\mathcal{A}$  and  $\theta$ , respectively. For stochastic policies of the form  $\pi_{\theta}(a|s) = \mu_{\theta}(s) + \text{noise}$ , the standard policy gradient theorem reduces to the above form as the noise level goes to zero.

Note that the gradient estimate in Equation (37.40) integrates over the states but not over the actions, which helps reduce the variance in gradient estimation from sampled trajectories. However, since the deterministic policy does not do any exploration, we need to use an off-policy method, that collects data from a stochastic behavior policy  $\beta$ , whose stationary state distribution is  $p_{\beta}^{\infty}$ . The original objective,  $J(\mu_{\theta})$ , is approximated by the following:

$$J_b(\mu_{\theta}) \triangleq \mathbb{E}_{p_{\beta}^{\infty}(s)} [V_{\mu_{\theta}}(s)] = \mathbb{E}_{p_{\beta}^{\infty}(s)} [Q_{\mu_{\theta}}(s, \mu_{\theta}(s))] \quad (37.41)$$

1 with the off-policy deterministic policy gradient approximated by (see also Section 37.5.1.2)  
2

$$\nabla_{\theta} J_b(\mu_{\theta}) \approx \mathbb{E}_{p_{\beta}^{\infty}(s)} [\nabla_{\theta} [Q_{\mu_{\theta}}(s, \mu_{\theta}(s))]] = \mathbb{E}_{p_{\beta}^{\infty}(s)} [\nabla_{\theta} \mu_{\theta}(s) \nabla_a Q_{\mu_{\theta}}(s, a)|_{a=\mu_{\theta}(s)} ds] \quad (37.42)$$

3 where we have dropped a term that depends on  $\nabla_{\theta} Q_{\mu_{\theta}}(s, a)$  and is hard to estimate [Sil+14].  
4

5 To apply Equation (37.42), we may learn  $Q_w \approx Q_{\mu_{\theta}}$  with TD, giving rise to the following updates:  
6

$$\delta = r_t + \gamma Q_w(s_{t+1}, \mu_{\theta}(s_{t+1})) - Q_w(s_t, a_t) \quad (37.43)$$

$$w_{t+1} \leftarrow w_t + \eta_w \delta \nabla_w Q_w(s_t, a_t) \quad (37.44)$$

$$\theta_{t+1} \leftarrow \theta_t + \eta_{\theta} \nabla_{\theta} \mu_{\theta}(s_t) \nabla_a Q_w(s_t, a)|_{a=\mu_{\theta}(s_t)} \quad (37.45)$$

12 This avoids importance sampling in the actor update because of the deterministic policy gradient,  
13 and avoids importance sampling in the critic update because of the use of Q-learning.  
14

15 If  $Q_w$  is linear in  $w$ , and uses features of the form  $\phi(s, a) = a^T \nabla_{\theta} \mu_{\theta}(s)$ , where  $a$  is the vector  
16 representation of  $a$ , then we say the function approximator for the critic is **compatible** with the  
17 actor; in this case, one can show that the above approximation does not bias the overall gradient.  
18 The method has been extended in various ways. The **DDPG** algorithm of [Lil+16], which stands  
19 for “deep deterministic policy gradient”, uses the DQN method (Section 37.2.6) to update  $Q$  that  
20 is represented by deep neural networks. The **TD3** algorithm [FHM18], which stands for “twin  
21 delayed DDPG”, extends DDPG by using double DQN (Section 37.2.5.2) and other heuristics to  
22 further improve performance. Finally, the **D4PG** algorithm [BM+18], which stands for “distributed  
23 distributional DDPG”, extends DDPG to handle distributed training, and to handle **distributional**  
24 **RL** (i.e., working with distributions of rewards instead of expected rewards [BDM17]).  
25

### 37.3.6 Gradient-free methods

26 The policy gradient estimator computes a “zeroth order” gradient, which essentially evaluates the  
27 function with randomly sampled trajectories. Sometimes it can be more efficient to use a derivative-  
28 free optimizer (Section 6.12), that does not even attempt to estimate the gradient. For example,  
29 [MGR18] obtain good results by training linear policies with random search, and [Sal+17b] show  
30 how to use evolutionary strategies to optimize the policy of a robotic controller.  
31

32

## 37.4 Model-based RL

34

35 Model-free approaches to RL typically need a lot of interactions with the environment to achieve  
36 good performance. For example, state of the art methods for the Atari benchmark, such as rainbow  
37 (Section 37.2.6), use millions of frames, equivalent to many days of playing at the standard frame rate.  
38 By contrast, humans can achieve the same performance in minutes [Tsi+17]. Similarly, OpenAI’s  
39 robot hand controller [And+20] learns to manipulate a cube using 100 years of simulated data.  
40

41 One promising approach to greater sample efficiency is **model-based RL (MBRL)**. In this  
42 approach, we first learn the transition model and reward function,  $p_T(s'|s, a)$  and  $R(s, a)$ , then  
43 use them to compute a near-optimal policy. This approach can significantly reduce the amount of  
44 real-world data that the agent needs to collect, since it can “try things out” in its imagination (i.e.,  
45 the models), rather than having to try them out empirically.

46 There are several ways we can use a model, and many different kinds of model we can create. Some  
47 of the algorithms mentioned earlier, such as MBIE and UCLR2 for provably efficient exploration

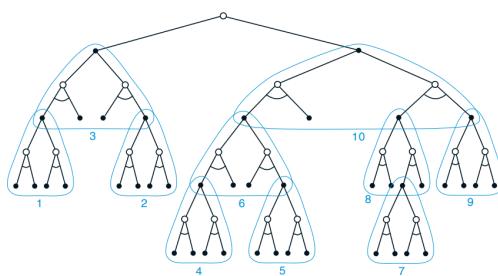


Figure 37.5: Illustration of heuristic search. In this figure, the subtrees are ordered according to a depth-first search procedure. From Figure 8.9 of [SB18]. Used with kind permission of Richard Sutton.

(Section 37.1.5.3), are examples of model-based methods. MBRL also provides a natural connection between RL and planning (Section 36.6) [Sut90]. We discuss some examples in the sections below, and refer to [MBJ20; PKP21; MH20] for more detailed reviews.

### 37.4.1 Model predictive control (MPC)

So far in this chapter, we have focused on trying to learn an optimal policy  $\pi_*(s)$ , which can then be used at run time to quickly pick the best action for any given state  $s$ . However, we can also avoid performing all this work in advance, and wait until we know what state we are in, call it  $s_t$ , and then use a model to predict future states and rewards that might follow for each possible sequence of future actions we might pursue. We then take the action that looks most promising, and repeat the process at the next step. More precisely, we compute

$$a_{t:t+H-1}^* = \underset{\mathbf{a}_{t:t+H-1}}{\operatorname{argmax}} \mathbb{E} \left[ \sum_{h=0}^{H-1} R(s_{t+h}, a_{t+h}) + \hat{V}(s_{t+H}) \right] \quad (37.46)$$

where the expectation is over state sequences that might result from executing  $a_{t:t+H-1}$  from state  $s_t$ . Here,  $H$  is called the **planning horizon**, and  $\hat{V}(s_{t+H})$  is an estimate of the reward-to-go at the end of this  $H$ -step look-ahead process. This is known as **receding horizon control** or **model predictive control (MPC)** [MM90; CA13]. We discuss some special cases of this below.

#### 37.4.1.1 Heuristic search

If the state and action spaces are finite, we can solve Equation (37.46) exactly, although the time complexity will typically be exponential in  $H$ . However, in many situations, we can prune off unpromising trajectories, thus making the approach feasible in large scale problems.

In particular, consider a discrete, deterministic MDP where reward maximization corresponds to finding a shortest path to a goal state. We can expand the successors of the current state according to all possible actions, trying to find the goal state. Since the search tree grows exponentially with depth, we can use a **heuristic function** to prioritize which nodes to expand; this is called **best-first search**, as illustrated in Figure 37.5.

<sup>1</sup> If the heuristic function is an optimistic lower bound on the true distance to the goal, it is called  
<sup>2</sup> **admissible**; If we aim to maximize total rewards, admissibility means the heuristic function is an  
<sup>3</sup> upper bound of the true value function. Admissibility ensures we will never incorrectly prune off  
<sup>4</sup> parts of the search space. In this case, the resulting algorithm is known as  **$A^*$  search**, and is optimal.  
<sup>5</sup> For more details on classical AI **heuristic search** methods, see [Pea84; RN19].  
<sup>6</sup>

<sup>7</sup>

### <sup>8</sup> 37.4.1.2 Monte-Carlo tree search (MCTS)

<sup>9</sup> Monte-Carlo tree search or MCTS is similar to heuristic search, but learns a value function for  
<sup>10</sup> each encountered state, rather than relying on a manually designed heuristic. It is inspired by UCB  
<sup>11</sup> for bandits (Section 36.4.5), but applies to general sequential decision making problems including  
<sup>12</sup> MDPs [KS06] where the UCB is used for efficient exploration of the state space.

<sup>13</sup> The MCTS method forms the basis of the famous **AlphaGo** and **AlphaZero** programs [Sil+16;  
<sup>14</sup> Sil+18], which can play expert-level Go, chess and shogi (Japanese chess). The action-value functions  
<sup>15</sup> for the intermediate nodes in the search tree are represented by deep neural networks, and updated  
<sup>16</sup> by RL methods that we discuss in Section 37.2. MCTS can also be applied to many other kinds of  
<sup>17</sup> sequential decision problems, such as experiment design for sequentially creating molecules [SPW18].  
<sup>18</sup> For more details on MCTS, see e.g., [Mun14].  
<sup>19</sup>

### <sup>20</sup> 37.4.1.3 Trajectory optimization for continuous actions

<sup>22</sup> For continuous actions, we cannot enumerate all possible branches in the search tree. Instead,  
<sup>23</sup> Equation (37.46) can be viewed as a nonlinear program, where  $a_{t:t+H-1}$  are the real-valued variables  
<sup>24</sup> to be optimized. If the system dynamics are linear and the reward function corresponds to negative  
<sup>25</sup> quadratic cost, the optimal action sequence can be solved mathematically, as in the **linear-quadratic-**  
<sup>26</sup> **Gaussian (LQG)** controller (see e.g., [AM89]). However, this problem is hard in general and often  
<sup>27</sup> solved by numerical methods such as **shooting** and **collocation** [Die+07; Rao10; Kal+11]. Many  
<sup>28</sup> of them work in an iterative fashion, starting with an initial action sequence followed by a step to  
<sup>29</sup> improve it. This process repeats until convergence of the cost.

<sup>30</sup> An example is **differential dynamic programming (DDP)** [JM70; TL05]. In each iteration,  
<sup>31</sup> DDP starts with a reference trajectory, and linearizes the system dynamics around states on the  
<sup>32</sup> trajectory to form a locally quadratic approximation of the reward function. This system can be  
<sup>33</sup> solved using LQG, whose optimal solution results in a new trajectory. The algorithm then moves to  
<sup>34</sup> the next iteration, with the new trajectory as the reference trajectory.

<sup>35</sup> Other alternatives are possible, including black-box (gradient-free) optimization methods like the  
<sup>36</sup> cross entropy method. (See the supplementary material for details.)

<sup>37</sup>

### <sup>38</sup> 37.4.2 Combining model-based and model-free

<sup>39</sup>

<sup>40</sup> In Section 37.4.1, we discussed MPC, which uses the model to decide which action to take at each  
<sup>41</sup> step. However, this can be slow, and can suffer from problems when the model is inaccurate. An  
<sup>42</sup> alternative is to use the learned model to help reduce the sample complexity of policy learning.

<sup>43</sup> There are many ways to do this. One approach is to generate rollouts from the model, and then  
<sup>44</sup> train a policy or  $Q$ -function on the “hallucinated” data. This is the basis of the famous **dyna** method  
<sup>45</sup> [Sut90]. In [Jan+19], they propose a similar method, but generate short rollouts from previously  
<sup>46</sup> visited real states; this ensures the model only has to extrapolate locally.

<sup>47</sup>

In [Web+17], they train a model to predict future states and rewards, but then use the hidden states of this model as additional context for a policy-based learning method. This can help overcome partial observability. They call their method **imagination-augmented agents**. A related method appears in [Jad+17], who propose to train a model to jointly predict future rewards and other auxiliary signals, such as future states. This can help in situations when rewards are sparse or absent.

### 37.4.3 MBRL using Gaussian processes

This section gives some examples of dynamics models that have been learned for low-dimensional continuous control problems. Such problems frequently arise in robotics. Since the dynamics are often nonlinear, it is useful to use a flexible and sample-efficient model family, such as Gaussian processes (Section 18.1). We will use notation like  $s$  and  $a$  for states and actions to emphasize they are vectors.

#### 37.4.3.1 PILCO

We first describe the **PILCO** method [DR11; DFR15], which stands for “probabilistic inference for learning control”. It is extremely data efficient for continuous control problems, enabling learning from scratch on real physical robots in a matter of minutes.

PILCO assumes the world model has the form  $s_{t+1} = f(s_t, a_t) + \epsilon_t$ , where  $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \Sigma)$ , and  $f$  is an unknown, continuous function.<sup>4</sup> The basic idea is to learn a Gaussian process (Section 18.1) approximation of  $f$  based on some initial random trajectories, and then to use this model to generate “fantasy” rollout trajectories of length  $T$ , that can be used to evaluate the expected cost of the current policy,  $J(\pi_\theta) = \sum_{t=1}^T \mathbb{E}_{a_t \sim \pi_\theta} [c(s_t)]$ , where  $s_0 \sim p_0$ . This function and its gradients wrt  $\theta$  can be computed deterministically, if a Gaussian assumption about the state distribution at each step is made, because the Gaussian belief state can be propagated deterministically through the GP model. Therefore, we can use deterministic batch optimization methods, such as Levenberg-Marquardt, to optimize the policy parameters  $\theta$ , instead of applying SGD to sampled trajectories.

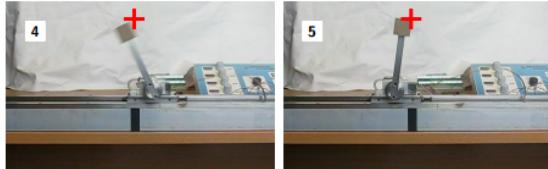
Due to its data efficiency, it is possible to apply PILCO to real robots. Figure 37.6a shows the results of applying it to solve a **cart-pole swing-up** task, where the goal is to make the inverted pendulum swing up by applying a horizontal force to move the cart back and forth. The state of the system  $s \in \mathbb{R}^4$  consists of the position  $x$  of the cart (with  $x = 0$  being the center of the track), the velocity  $\dot{x}$ , the angle  $\theta$  of the pendulum (measured from hanging downward), and the angular velocity  $\dot{\theta}$ . The control signal  $a \in \mathbb{R}$  is the force applied to the cart. The target state is  $s_* = (0, \star, \pi, \star)$ , corresponding to the cart being in the middle and the pendulum being vertical, with velocities unspecified. The authors used an RBF controller with 50 basis functions, amounting to a total of 305 policy parameters. The controller was successfully trained using just 7 real world trials.<sup>5</sup>

Figure 37.6b shows the results of applying PILCO to solve a **block stacking** task using a low-quality robot arm with 6 degrees of freedom. A separate controller was trained for each block. The state space  $s \in \mathbb{R}^3$  is the 3d location of the center of the block in the arm’s gripper (derived from an RGBD sensor), and the control  $a \in \mathbb{R}^4$  corresponds to the pulse widths of four servo motors. A linear policy was successfully trained using as few as 10 real world trials.

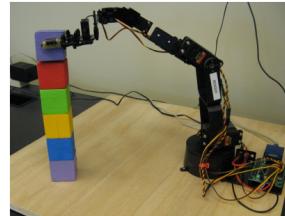
<sup>4</sup> An alternative, which often works better, is to use  $f$  to model the residual, so that  $s_{t+1} = s_t + f(s_t, a_t) + \epsilon_t$ .

<sup>5</sup> 2 random initial trials, each 5 seconds, and then 5 policy-generated trials, each 2.5 seconds, totalling 17.5 seconds.

1  
2  
3  
4  
5  
6  
7  
8  
9  
10



(a)



(b)

11 *Figure 37.6: (a) A cart-pole system being controlled by a policy learned by PILCO using just 17.5 seconds*  
12 *of real-world interaction. The goal state is marked by the red cross. The initial state is where the cart is*  
13 *stationary on the right edge of the workspace, and the pendulum is horizontal. For a video of the system*  
14 *learning, see <https://bit.ly/35fpLmR>. (b) A low-quality robot arm being controlled by a block-stacking*  
15 *policy learned by PILCO using just 230 seconds of real-world interaction. From Figures 11, 12 from [DFR15].*  
16 *Used with kind permission of Marc Deisenroth.*

17  
18  
19

### 20 37.4.3.2 GP-MPC

21 [KD18a] have proposed an extension to PILCO that they call **GP-MPC**, since it combines a GP  
22 dynamics model with model predictive control (Section 37.4.1). In particular, they use an open-loop  
23 control policy to propose a sequence of actions,  $\mathbf{a}_{t:t+H-1}$ , as opposed to sampling them from a policy.  
24 They compute a Gaussian approximation to the future state trajectory,  $p(\mathbf{s}_{t+1:t+H}|\mathbf{a}_{t:t+H-1}, \mathbf{s}_t)$ , by  
25 moment matching, and use this to deterministically compute the expected reward and its gradient  
26 wrt  $\mathbf{a}_{t:t+H-1}$  (as opposed to the policy parameters  $\theta$ ). Using this, they can solve Equation (37.46)  
27 to find  $\mathbf{a}_{t:t+H-1}^*$ ; finally, they execute the first step of this plan,  $a_t^*$ , and repeat the whole process.  
28 The advantage of GP-MPC over policy-based PILCO is that it can handle constraints more easily,  
29 and it can be more data efficient, since it continually updates the GP model after every step (instead  
30 of at the end of an trajectory).

31  
32

### 33 37.4.4 MBRL using DNNs

34

35 Gaussian processes do not scale well to large sample sizes and high dimensional data. Deep neural  
36 networks (DNNs) work much better in this regime. However, they do not naturally model uncertainty,  
37 which can cause MPC methods to fail. We discuss various methods for representing uncertainty with  
38 DNNs in Section 17.1. Here, we mention a few approaches that have been used for MBRL.

39 The **deep PILCO** method uses DNNs together with Monte Carlo dropout (Section 17.4.5) to  
40 represent uncertainty [GMAR16]. [Chu+18] proposed **probabilistic ensembles with trajectory**  
41 **sampling** or **PETS**, which represents uncertainty using an ensemble of DNNs (Section 17.4.8).  
42 Many other approaches are possible, depending on the details of the problem being tackled.

43 Since these are all stochastic methods (as opposed to the GP methods above), they can suffer from  
44 a high variance in the predicted returns, which can make it difficult for the MPC controller to pick  
45 the best action. We can reduce variance with the **common random number** trick [KSN99], where  
46 all rollouts share the same random seed, so differences in  $J(\pi_\theta)$  can be attributed to changes in  $\theta$

47

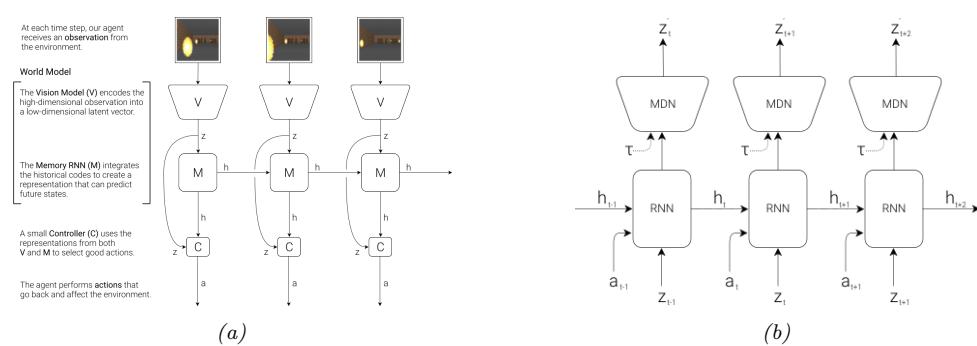


Figure 37.7: (a) Illustration of an agent interacting with the VizDoom environment. (The yellow blobs represent fireballs being thrown towards the agent by various enemies.) The agent has a world model, composed of a vision system  $V$  and a memory RNN  $M$ , and has a controller  $C$ . (b) Detailed representation of the memory model. Here  $h_t$  is the deterministic hidden state of the RNN at time  $t$ , which is used to predict the next latent of the VAE,  $z_{t+1}$ , using a mixture density network (MDN). Here  $\tau$  is a temperature parameter used to increase the variance of the predictions, to prevent the controller from exploiting model inaccuracies. From Figures 4, 6 of [HS18]. Used with kind permission of David Ha.

but not other factors. This technique was used in **PEGASUS** [NJ00]<sup>6</sup> and in [HMD18].

### 37.4.5 MBRL using latent-variable models

In this section, we describe some methods that learn latent variable models, rather than trying to predict dynamics directly in the observed space, which is hard to do when the states are images.

#### 37.4.5.1 World models

The ‘‘world models’’ paper [HS18] showed how to learn a generative model of two simple video games (CarRacing and a VizDoom-like environment), such that the model can be used to train a policy entirely in simulation. The basic idea is shown in Figure 37.7. First, we collect some random experience, and use this to fit a VAE model (Section 22.2) to reduce the dimensionality of the images,  $\mathbf{x}_t \in \mathbb{R}^{64 \times 64 \times 3}$ , to a latent  $\mathbf{z}_t \in \mathbb{R}^{64}$ . Next, we train an RNN to predict  $p(\mathbf{z}_{t+1} | \mathbf{z}_t, \mathbf{a}_t, \mathbf{h}_t)$ , where  $\mathbf{h}_t$  is the deterministic RNN state, and  $\mathbf{a}_t$  is the continuous action vector (3-dimensional in both cases). The emission model for the RNN is a mixture density network, in order to model multi-modal futures. Finally, we train the controller using  $\mathbf{z}_t$  and  $\mathbf{h}_t$  as inputs; here  $\mathbf{z}_t$  is a compact representation of the current frame, and  $\mathbf{h}_t$  is a compact representation of the predicted distribution over  $\mathbf{z}_{t+1}$ .

The authors of [HS18] trained the controller using a derivative free optimizer called **CMA-ES** (covariance matrix adaptation evolutionary strategy, see supplementary). It can work better than policy gradient methods, as discussed in Section 37.3.6. However, it does not scale to high dimensions. To tackle this, the authors use a linear controller, which has only 867 parameters.<sup>7</sup> By contrast,

6. PEGASUS stands for ‘‘Policy Evaluation-of-Goodness And Search Using Scenarios’’, where the term ‘‘scenario’’ refers to one of the shared random samples.

7. The input is a 32-dimensional  $\mathbf{z}_t$  plus a 256-dimensional  $\mathbf{h}_t$ , and there are 3 outputs. So the number of parameters

<sup>1</sup> VAE has 4.3M parameters and MDN-RNN 422k. Fortunately, these two models can be trained in an  
<sup>2</sup> unsupervised way from random rollouts, so sample efficiency is less critical than when training the  
<sup>3</sup> policy.  
<sup>4</sup>

<sup>5</sup> So far, we have described how to use the representation learned by the generative model as  
<sup>6</sup> informative features for the controller, but the controller is still learned by interacting with the  
<sup>7</sup> real world. Surprisingly, we can also train the controller entirely in “dream mode”, in which the  
<sup>8</sup> generated images from the VAE decoder at time  $t$  are fed as input to the VAE encoder at time  $t + 1$ ,  
<sup>9</sup> and the MDN-RNN is trained to predict the next reward  $r_{t+1}$  as well as  $\mathbf{z}_{t+1}$ . Unfortunately, this  
<sup>10</sup> method does not always work, since the model (which is trained in an unsupervised way) may fail to  
<sup>11</sup> capture task-relevant features (due to underfitting) and may memorize task-irrelevant features (due  
<sup>12</sup> to overfitting). The controller can learn to exploit weaknesses in the model (similar to an adversarial  
<sup>13</sup> attack) and achieve high simulated reward, but such a controller may not work well when transferred  
<sup>14</sup> to the real world.

<sup>15</sup> One approach to combat this is to artificially increase the variance of the MDN model (by using  
<sup>16</sup> a temperature parameter  $\tau$ ), in order to make the generated samples more stochastic. This forces  
<sup>17</sup> the controller to be robust to large variations; the controller will then treat the real world as just  
<sup>18</sup> another kind of noise. This is similar to the technique of domain randomization, which is sometimes  
<sup>19</sup> used for sim-to-real applications; see e.g., [MAZA18].  
<sup>20</sup>

### <sup>21</sup> 37.4.5.2 PlaNet and Dreamer

<sup>22</sup> In [HS18], they first learn the world model on random rollouts, and then train a controller. On harder  
<sup>23</sup> problems, it is necessary to iterate these two steps, so the model can be trained on data collected by  
<sup>24</sup> the controller, in an iterative fashion.  
<sup>25</sup>

<sup>26</sup> In this section, we describe one method of this kind, known as **PlaNet** [Haf+19]. PlaNet  
<sup>27</sup> uses a POMDP model, where  $\mathbf{z}_t$  are the latent states,  $\mathbf{s}_t$  are the observations,  $\mathbf{a}_t$  are the ac-  
<sup>28</sup> tions, and  $r_t$  are the rewards. It fits a recurrent state space model (Section 31.5.2) of the form  
<sup>29</sup>  $p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})p(\mathbf{s}_t|\mathbf{z}_t)p(r_t|\mathbf{z}_t)$  using variational inference, where the posterior is approximated by  
<sup>30</sup>  $q(\mathbf{z}_t|\mathbf{s}_{1:t}, \mathbf{a}_{1:t-1})$ . After fitting the model to some random trajectories, the system uses the inference  
<sup>31</sup> model to compute the current belief state, and then uses the cross entropy method to find an  
<sup>32</sup> action sequence for the next  $H$  steps to maximize expected reward, by optimizing in latent space.  
<sup>33</sup> The system then executes  $\mathbf{a}_t^*$ , updates the model, and repeats the whole process. To encourage  
<sup>34</sup> the dynamics model to capture long term trajectories, they use the “latent overshooting” training  
<sup>35</sup> method described in Section 31.5.3. The PlaNet method outperforms model-free methods, such as  
<sup>36</sup> A3C (Section 37.3.3.1) and D4PG (Section 37.3.5), on various image-based continuous control tasks,  
<sup>37</sup> illustrated in Figure 37.8.

<sup>38</sup> Although PlaNet is sample efficient, it is not computationally efficient. For example, they use  
<sup>39</sup> CEM with 1000 samples and 10 iterations to optimize trajectories with a horizon of length 12, which  
<sup>40</sup> requires 120,000 evaluations of the transition dynamics to choose a single action. [AY19] improve  
<sup>41</sup> this by replacing CEM with differentiable CEM (see supplementary), and then optimize in a latent  
<sup>42</sup> space of action sequences. This is much faster, but the results are not quite as good. However, since  
<sup>43</sup> the whole policy is now differentiable, it can be fine-tuned using PPO (Section 37.3.4), which closes  
<sup>44</sup> the performance gap at negligible cost.  
<sup>45</sup>

<sup>46</sup>  $(32 + 256) \times 3 + 3 = 867$ , to account for the weights and biases.  
<sup>47</sup>

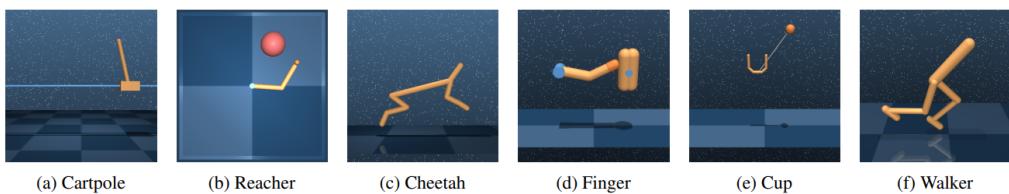


Figure 37.8: Illustration of some image-based control problems used in the PlaNet paper. Inputs are  $64 \times 64 \times 3$ . (a) The cartpole swingup task has a fixed camera so the cart can move out of sight, making this a partially observable problem. (b) The reacher task has a sparse reward. (c) The cheetah running task includes both contacts and a larger number of joints. (d) The finger spinning task includes contacts between the finger and the object. (e) The cup task has a sparse reward that is only given once the ball is caught. (f) The walker task requires balance and predicting difficult interactions with the ground when the robot is lying down. From Figure 1 of [Haf+19]. Used with kind permission of Danijar Hafner.

A recent extension of the PlaNet paper, known as **Dreamer**, was proposed in [Haf+20]. In this paper, the online MPC planner is replaced by a policy network,  $\pi(a_t|z_t)$ , which is learned using gradient-based actor-critic in latent space. The inference and generative models are trained by maximizing the ELBO, as in PlaNet. The policy is trained by SGD to maximize expected total reward as predicted by the value function, and the value function is trained by SGD to minimize MSE between predicted future reward and the TD- $\lambda$  estimate (Section 37.2.2). They show that Dreamer gives better results than PlaNet, presumably because they learn a policy to optimize the long term reward (as estimated by the value function), rather than relying on MPC based on short-term rollouts.

### 37.4.6 Robustness to model errors

The main challenge with MBRL is that errors in the model can result in poor performance of the resulting policy, due to the distribution shift problem (Section 20.2). That is, the model is trained to predict states and rewards that it has seen using some behavior policy (e.g., the current policy), and then is used to compute an optimal policy under the learned model. When the latter policy is followed, the agent will experience a different distribution of states, under which the learned model may not be a good approximation of the real environment.

We require the model to generalize in a robust way to new states and actions. (This is related to the off-policy learning problem that we discuss in Section 37.5.) Failing that, the model should at least be able to quantify its uncertainty (Section 20.4). These topics are the focus of much recent research (see e.g., [Luo+19; Kur+19; Jan+19; Isl+19; Man+19; WB20; Eys+21]).

## 37.5 Off-policy learning

We have seen examples of off-policy methods such as Q-learning. They do not require that training data be generated by the policy it tries to evaluate or improve. Therefore, they tend to have greater data efficiency than their on-policy counterparts, by taking advantage of data generated by other policies. They are also easier to be applied in practice, especially in domains where costs and risks of

1 following a new policy must be considered. This section covers this important topic.  
2

3 A key challenge in off-policy learning is that the data distribution is typically different from the  
4 desired one, and this mismatch must be dealt with. For example, the probability of visiting a state  $s$   
5 at time  $t$  in a trajectory depends not only on the MDP's transition model, but also on the policy  
6 that is being followed. If we are to estimate  $J(\pi)$ , as defined in Equation (37.15), but the trajectories  
7 are generated by a different policy  $\pi'$ , simply averaging rewards in the data gives us  $J(\pi')$ , not  $J(\pi)$ .  
8 We have to somehow correct for the gap, or "bias". Another challenge is that off-policy data can also  
9 make an algorithm unstable and divergent, which we will discuss in Section 37.5.3.

10 Removing distribution mismatches is not unique in off-policy learning, and is also needed in  
11 supervised learning to handle covariate shift (Section 20.2.3), and in causal effect estimation (Chap-  
12 ter 38), among others. Off-policy learning is also closely related to **offline reinforcement learning**  
13 (also called **batch reinforcement learning**): the former emphasizes the distributional mismatch  
14 between data and the agent's policy, and the latter emphasizes that the data is static and no further  
15 online interaction with the environment is allowed [LP03; EGW05; Lev+20]. Clearly, in the offline  
16 scenario with fixed data, off-policy learning is typically a critical technical component. Recently,  
17 several datasets have been prepared to facilitate empirical comparisons of offline RL methods (see  
18 e.g., [Gul+20; Fu+20]).

19 Finally, while this section focuses on MDPs, most methods can be simplified and adapted to the  
20 special case of contextual bandits (Section 36.4). In fact, off-policy methods have been successfully  
21 used in numerous industrial bandit applications (see e.g., [Li+10; Bot+13; SJ15; HLR16]).

22

### 23 37.5.1 Basic techniques

24

25 We start with four basic techniques, and will consider more sophisticated ones in subsequent sections.  
26 The off-policy data is assumed to be a collection of trajectories:  $\mathcal{D} = \{\tau^{(i)}\}_{1 \leq i \leq n}$ , where each  
27 trajectory is a sequence as before:  $\tau^{(i)} = (s_0^{(i)}, a_0^{(i)}, r_0^{(i)}, s_1^{(i)}, \dots)$ . Here, the reward and next states  
28 are sampled according to the reward and transition models; the actions are chosen by a **behavior**  
29 **policy**, denoted  $\pi_b$ , which is different from the **target policy**,  $\pi_e$ , that the agent is evaluating or  
30 improving. When  $\pi_b$  is unknown, we are in a **behavior-agnostic off-policy** setting.

31

#### 32 37.5.1.1 Direct method

33

34 A natural approach to off-policy learning starts with estimating the unknown reward and transition  
35 models of the MDP from off-policy data. This can be done using regression and density estimation  
36 methods on the reward and transition models, respectively, to obtain  $\hat{R}$  and  $\hat{P}$ ; see Section 37.4 for  
37 further discussions. These estimated models then give us an inexpensive way to (approximately)  
38 simulate the original MDP, and we can apply on-policy methods on the simulated data. This method  
39 directly models the outcome of taking an action in a state, thus the name **direct method**, and is  
40 sometimes known as **regression estimator** and **plug-in estimator**.

41 While the direct method is natural and sometimes effective, it has a few limitations. First, a small  
42 estimation error in the simulator has a compounding effect in long-horizon problems (or equivalently,  
43 when the discount factor  $\gamma$  is close to 1). Therefore, an agent that is optimized against an MDP  
44 simulator may overfit the estimation errors. Unfortunately, learning the MDP model, especially the  
45 transition model, is generally difficult, making the method limited in domains where  $\hat{R}$  and  $\hat{P}$  can be  
46 learned to high fidelity. See Section 37.4.6 for a related discussion.

47

1    **37.5.1.2 Importance sampling**

2    The second approach relies on importance sampling (IS) (Section 11.5) to correct for distributional  
3    mismatches in the off-policy data. To demonstrate the idea, consider the problem of estimating the  
4    target policy value  $J(\pi_e)$  with a fixed horizon  $T$ . Correspondingly, the trajectories in  $\mathcal{D}$  are also of  
5    length  $T$ . Then, the IS off-policy estimator, first adopted by [PSS00], is given by  
6

7

$$\hat{J}_{\text{IS}}(\pi_e) \triangleq \frac{1}{n} \sum_{i=1}^n \frac{p(\boldsymbol{\tau}^{(i)} | \pi_e)}{p(\boldsymbol{\tau}^{(i)} | \pi_b)} \sum_{t=0}^{T-1} \gamma^t r_t^{(i)} \quad (37.47)$$

8

9    It can be verified that  $\mathbb{E}_{\pi_b} [\hat{J}_{\text{IS}}(\pi_e)] = J(\pi_e)$ , that is,  $\hat{J}_{\text{IS}}(\pi_e)$  is **unbiased**, provided that  $p(\boldsymbol{\tau} | \pi_b) > 0$   
10 whenever  $p(\boldsymbol{\tau} | \pi_e) > 0$ . The **importance ratio**,  $\frac{p(\boldsymbol{\tau}^{(i)} | \pi_e)}{p(\boldsymbol{\tau}^{(i)} | \pi_b)}$ , is used to compensate for the fact that the  
11 data is sampled from  $\pi_b$  and not  $\pi_e$ . Furthermore, this ratio does *not* depend on the MDP models,  
12 because for any trajectory  $\boldsymbol{\tau} = (s_0, a_0, r_0, s_1, \dots, s_T)$ , we have from Equation (36.50) that  
13

14

$$\frac{p(\boldsymbol{\tau} | \pi_e)}{p(\boldsymbol{\tau} | \pi_b)} = \frac{p(s_0) \prod_{t=0}^{T-1} \pi_e(a_t | s_t) p_T(s_{t+1} | s_t, a_t) p_R(r_t | s_t, a_t, s_{t+1})}{p(s_0) \prod_{t=0}^{T-1} \pi_b(a_t | s_t) p_T(s_{t+1} | s_t, a_t) p_R(r_t | s_t, a_t, s_{t+1})} = \prod_{t=0}^{T-1} \frac{\pi_e(a_t | s_t)}{\pi_b(a_t | s_t)} \quad (37.48)$$

15

16    This simplification makes it easy to apply IS, as long as the target and behavior policies are known. If  
17 the behavior policy is unknown, we can estimate it from  $\mathcal{D}$  (using e.g., logistic regression or DNNs), and  
18 replace  $\pi_b$  by its estimate  $\hat{\pi}_b$  in Equation (37.48). For convenience, define the **per-step importance**  
19 **ratio** at time  $t$  by  $\rho_t(\boldsymbol{\tau}) \triangleq \pi_e(a_t | s_t) / \pi_b(a_t | s_t)$ , and similarly,  $\hat{\rho}_t(\boldsymbol{\tau}) \triangleq \pi_e(a_t | s_t) / \hat{\pi}_b(a_t | s_t)$ .  
20

21    Although IS can in principle eliminate distributional mismatches, in practice its usability is often  
22 limited by its potentially high variance. Indeed, the importance ratio in Equation (37.47) can be  
23 arbitrarily large if  $p(\boldsymbol{\tau}^{(i)} | \pi_e) \gg p(\boldsymbol{\tau}^{(i)} | \pi_b)$ . There are many improvements to the basic IS estimator.  
24 One improvement is based on the observation that the reward  $r_t$  is independent of the trajectory  
25 beyond time  $t$ . This leads to a **per-decision importance sampling** variant that often yields lower  
26 variance (see Section 11.6.1 for a statistical motivation, and [LBB20] for a further discussion):  
27

28

$$\hat{J}_{\text{PDIS}}(\pi_e) \triangleq \frac{1}{n} \sum_{i=1}^n \sum_{t=0}^{T-1} \prod_{t' \leq t} \rho_{t'}(\boldsymbol{\tau}^{(i)}) \gamma^t r_t^{(i)} \quad (37.49)$$

29

30    There are many other variants such as self-normalized IS and truncated IS, both of which aim to  
31 reduce variance possibly at the cost of a small bias; precise expressions of these alternatives are found,  
32 e.g., in [Liu+18b]. In the next subsection, we will discuss another systematic way to improve IS.  
33

34    IS may also be applied to improve a policy against the policy value given in Equation (37.15).  
35 However, directly applying the calculation of Equation (37.48) runs into a fundamental issue with IS,  
36 which we will discuss in Section 37.5.2. For now, we may consider the following approximation of  
37 policy value, averaging over the state distribution of the behavior policy:  
38

39

$$J_b(\pi_\theta) \triangleq \mathbb{E}_{p_\beta^\infty(s)} [V_\pi(s)] = \mathbb{E}_{p_\beta^\infty(s)} \left[ \sum_a \pi_\theta(a | s) Q_\pi(s, a) \right] \quad (37.50)$$

40

<sup>1</sup> Differentiating this and ignoring the term  $\nabla_{\theta}Q_{\pi}(s, a)$ , as suggested by [DWS12], gives a way to  
<sup>2</sup> (approximately) estimate the **off-policy policy-gradient** using a one-step IS correction ratio:  
<sup>3</sup>

$$\begin{aligned}\nabla_{\theta}J_b(\pi_{\theta}) &\approx \mathbb{E}_{p_{\beta}^{\infty}(s)} \left[ \sum_a \nabla_{\theta}\pi_{\theta}(a|s)Q_{\pi}(s, a) \right] \\ &= \mathbb{E}_{p_{\beta}^{\infty}(s)\beta(a|s)} \left[ \frac{\pi_{\theta}(a|s)}{\beta(a|s)} \nabla_{\theta}\log\pi_{\theta}(a|s)Q_{\pi}(s, a) \right]\end{aligned}$$

<sup>10</sup> Finally, we note that in the tabular MDP case, there exists a policy  $\pi_*$  that is optimal in all states  
<sup>11</sup>(Section 36.5.5). This policy maximizes  $J$  and  $J_b$  simultaneously, so Equation (37.50) can be a good  
<sup>12</sup>proxy for Equation (37.15) as long as all states are “covered” by the behavior policy  $\pi_b$ . The situation  
<sup>13</sup>is similar when the set of value functions or policies under consideration is sufficiently expressive: an  
<sup>14</sup>example is a Q-learning like algorithm called Retrace [Mun+16; ASN20]. Unfortunately, in general  
<sup>15</sup>when we work with parametric families of value functions or policies, such a uniform optimality is  
<sup>16</sup>lost, and the distribution of states has a direct impact on the solution found by the algorithm. We  
<sup>17</sup>will revisit this problem in Section 37.5.2.

<sup>18</sup>

### <sup>19</sup>37.5.1.3 Doubly robust

<sup>20</sup>

<sup>21</sup>It is possible to combine the direct and importance sampling methods discussed previously. To  
<sup>22</sup>develop intuition, consider the problem of estimating  $J(\pi_e)$  in a contextual bandit (Section 36.4),  
<sup>23</sup>that is, when  $T = 1$  in  $\mathcal{D}$ . The **doubly robust** (DR) estimator is given by

$$\hat{J}_{\text{DR}}(\pi_e) \triangleq \frac{1}{n} \sum_{i=1}^n \left( \frac{\pi_e(a_0^{(i)}|s_0^{(i)})}{\hat{\pi}_b(a_0^{(i)}|s_0^{(i)})} \left( r_0^{(i)} - \hat{Q}(s_0^{(i)}, a_0^{(i)}) \right) + \hat{V}(s_0^{(i)}) \right) \quad (37.51)$$

<sup>28</sup>where  $\hat{Q}$  is an estimate of  $Q_{\pi_e}$ , which can be obtained using methods discussed in Section 37.2,  
<sup>29</sup>and  $\hat{V}(s) = \mathbb{E}_{\pi_e(a|s)} [\hat{Q}(s, a)]$ . If  $\hat{\pi}_b = \pi_b$ , the term  $\hat{Q}$  is canceled by  $\hat{V}$  on average, and we get the  
<sup>30</sup>IS estimate that is unbiased; if  $\hat{Q} = Q_{\pi_e}$ , the term  $\hat{Q}$  is canceled by the reward on average, and  
<sup>31</sup>we get the estimator as in the direct method that is also unbiased. In other words, the estimator  
<sup>32</sup>Equation (37.51) is unbiased, as long as one of the estimates,  $\hat{\pi}_b$  and  $\hat{Q}$ , is right. This observation  
<sup>33</sup>justifies the name doubly robust, which has its origin in causal inference (see e.g., [BR05]).

<sup>35</sup> The above DR estimator may be extended to MDPs recursively, starting from the last step. Given  
<sup>36</sup>a length- $T$  trajectory  $\tau$ , define  $\hat{J}_{\text{DR}}[T] \triangleq 0$ , and for  $t < T$ ,

$$\hat{J}_{\text{DR}}[t] \triangleq \hat{V}(s_t) + \hat{\rho}_t(\tau) \left( r_t + \gamma \hat{J}_{\text{DR}}[t+1] - \hat{Q}(s_t, a_t) \right) \quad (37.52)$$

<sup>39</sup>

<sup>40</sup>where  $\hat{Q}(s_t, a_t)$  is the estimated cumulative reward for the remaining  $T - t$  steps. The DR estimator  
<sup>41</sup>of  $J(\pi_e)$ , denoted  $\hat{J}_{\text{DR}}(\pi_e)$ , is the average of  $\hat{J}_{\text{DR}}[0]$  over all  $n$  trajectories in  $\mathcal{D}$  [JL16]. It can be  
<sup>42</sup>verified (as an exercise) that the recursive definition is equivalent to  
<sup>43</sup>

$$\hat{J}_{\text{DR}}[0] = \hat{V}(s_0) + \sum_{t'=0}^{T-1} \left( \prod_{t'=0}^t \hat{\rho}_{t'}(\tau) \right) \gamma^t \left( r_t + \gamma \hat{V}(s_{t+1}) - \hat{Q}(s_t, a_t) \right) \quad (37.53)$$

<sup>46</sup>

This form can be easily generalized to the infinite-horizon setting by letting  $T \rightarrow \infty$  [TB16]. Other than double robustness, the estimator is also shown to result in minimum variance under certain conditions [JL16]. Finally, the DR estimator can be incorporated into policy gradient for policy optimization, to reduce gradient estimation variance [HJ20].

#### 37.5.1.4 Behavior regularized method

The three methods discussed previously do not impose any constraint on the target policy  $\pi_e$ . Typically, the more different  $\pi_e$  is from  $\pi_b$ , the less accurate our off-policy estimation can be. Therefore, when we optimize a policy in offline RL, a natural strategy is to favor target policies that are “close” to the behavior policy. Similar ideas are discussed in the context of conservative policy gradient (Section 37.3.4).

One approach is to impose a hard constraint on the proximity between the two policies. For example, we may modify the loss function of DQN (Equation (37.14)) as follows

$$\mathcal{L}_1^{\text{DQN}}(\mathbf{w}) \triangleq \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} \left[ (r + \gamma \max_{\pi: D(\pi, \pi_b) \leq \varepsilon} \mathbb{E}_{\pi(a'|s')} [Q_{\mathbf{w}^-}(s', a')] - Q_{\mathbf{w}}(s, a))^2 \right] \quad (37.54)$$

In the above, we replace the  $\max_{a'}$  operation by an expectation over a policy that stays close enough to the behavior policy, measured by some distance function  $D$ . For various instantiations and further details, see e.g., [FMP19; Kum+19a].

We may also impose a soft constraint on the proximity, by penalizing target policies that are too different. The DQN loss function can be adapted accordingly:

$$\mathcal{L}_2^{\text{DQN}}(\mathbf{w}) \triangleq \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} \left[ (r + \gamma \max_{\pi} \mathbb{E}_{\pi(a'|s')} [Q_{\mathbf{w}^-}(s', a')] - \alpha \gamma D(\pi(s'), \pi_b(s')) - Q_{\mathbf{w}}(s, a))^2 \right] \quad (37.55)$$

This idea has been used in contextual bandits [SJ15] and empirically studied in MDPs by [WTN19].

There are many choices for the function  $D$ , such as the KL-divergence, for both hard and soft constraints. More detailed discussions and examples can be found in [Lev+20].

Finally, behavior regularization and previous methods like IS can be combined, where the former ensures lower variance and greater generalization of the latter (e.g., [SJ15]). Furthermore, most proposed behavior regularized methods consider one-step difference in  $D$ , comparing  $\pi(s)$  and  $\pi_b(s)$  conditioned on  $s$ . In many cases, it is desired to consider the difference between the long-term distributions,  $p_{\beta}^{\infty}$  and  $p^{\infty}$ , which we will discuss next.

#### 37.5.2 The curse of horizon

The IS and DR approaches presented in the previous section all rely on an importance ratio to correct distributional mismatches. The ratio depends on the entire trajectory, and its variance grows exponentially in the trajectory length  $T$ . Correspondingly, the off-policy estimate of either the policy value or policy gradient can suffer an exponentially large variance (and thus very low accuracy), a challenge called the **curse of horizon** [Liu+18b]. Policies found by approximate algorithms like Q-learning and off-policy actor-critic often have hard-to-control error due to distribution mismatches.

This section discusses an approach to tackling this challenge, by considering corrections in the state-action distribution, rather than in the trajectory distribution. This change is critical: [Liu+18b]

1 describes an example, where the state-action distributions under the behavior and target policies  
2 are identical, but the importance ratio of a trajectory grows exponentially large. It is now more  
3 convenient to assume the off-policy data consists of a set of transitions:  $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{1 \leq i \leq m}$ ,  
4 where  $(s_i, a_i) \sim p_{\mathcal{D}}$  (some fixed but unknown sampling distribution, such as  $p_{\beta}^{\infty}$ ), and  $r_i$  and  
5  $s'_i$  are sampled from the MDP's reward and transition models. Given a policy  $\pi$ , we aim to  
6 estimate the correction ratio  $\zeta_*(s, a) = p_{\pi}^{\infty}(s, a)/p_{\mathcal{D}}(s, a)$ , as it allows us to rewrite the policy value  
7 (Equation (37.15)) as  
8

$$\frac{9}{10} J(\pi) = \frac{1}{1-\gamma} \mathbb{E}_{p_{\pi}^{\infty}(s,a)} [R(s,a)] = \frac{1}{1-\gamma} \mathbb{E}_{p_{\beta}^{\infty}(s,a)} [\zeta_*(s,a) R(s,a)] \quad (37.56)$$

12 For simplicity, we assume the initial state distribution  $p_0$  is known, or can be easily sampled from.  
13 This assumption is often easy to satisfy in practice.

14 The starting point is the following linear program formulation for any given  $\pi$ :

$$\frac{16}{17} \max_{d \geq 0} -D_f(d \| p_{\mathcal{D}}) \quad \text{s.t.} \quad d(s,a) = (1-\gamma)\mu_0(s)\pi(a|s) + \gamma \sum_{\bar{s}, \bar{a}} p(s|\bar{s}, \bar{a})d(\bar{s}, \bar{a})\pi(a|s) \quad \forall (s,a)$$

18 (37.57)

19 where  $D_f$  is the  $f$ -divergence (Section 2.9.1). The constraint is a variant of Equation (36.69), giving  
20 similar flow conditions in the space of  $\mathcal{S} \times \mathcal{A}$  under policy  $\pi$ . Under mild conditions,  $p_{\pi}^{\infty}$  is only  
21 solution that satisfies the flow constraints, so the objective does not affect the solution, but will  
22 facilitate the derivation below. We can now obtain the Lagrangian, with multipliers  $\{\nu(s,a)\}$ , and  
23 use the change-of-variables  $\zeta(s,a) = d(s,a)/p_{\mathcal{D}}(s,a)$  to obtain the following optimization problem:

$$\frac{25}{26} \max_{\zeta \geq 0} \min_{\nu} \mathcal{L}(\zeta, \nu) = \mathbb{E}_{p_{\mathcal{D}}(s,a)} [-f(\zeta(s,a))] + (1-\gamma)\mathbb{E}_{p_0(s)\pi(a|s)} [\nu(s,a)] \quad (37.58)$$

$$\frac{27}{28} + \mathbb{E}_{\pi(a'|s')p(s'|s,a)p_{\mathcal{D}}(s,a)} [\zeta(s,a) (\gamma\nu(s',a') - \nu(s,a))]$$

29 It can be shown that the saddle point to Equation (37.58) must coincide with the desired correction  
30 ratio  $\zeta_*$ . In practice, we may parameterize  $\zeta$  and  $\nu$ , and apply two-timescales stochastic gradient  
31 descent/ascent on the off-policy data  $\mathcal{D}$  to solve for an approximate saddle-point. This is the  
32 **DualDICE** method [Nac+19a], which is extended to **GenDICE** [Zha+20c].

33 Compared to the IS or DR approaches, Equation (37.58) does not compute the importance ratio of  
34 a trajectory, thus generally has a lower variance. Furthermore, it is behavior-agnostic, without having  
35 to estimate the behavior policy, or even to assume data consists of a collection of trajectories. Finally,  
36 this approach can be extended to be doubly robust (e.g., [UHJ20]), and to optimize a policy [Nac+19b]  
37 against the true policy value  $J(\pi)$  (as opposed to approximations like Equation (37.50)). For more  
38 examples along this line of approach, see [ND20] and the references therein.

39

### 40 37.5.3 The deadly triad

41

42 Other than introducing bias, off-policy data may also make a value-based RL method unstable and  
43 even divergent. Consider the simple MDP depicted in Figure 37.9a, due to [Bai95]. It has 7 states  
44 and 2 actions. Taking the dashed action takes the environment to the 6 upper states uniformly at  
45 random, while the solid action takes it to the bottom state. The reward is 0 in all transitions, and  
46  $\gamma = 0.99$ . The value function  $V_w$  uses a linear parameterization indicated by the expressions shown  
47

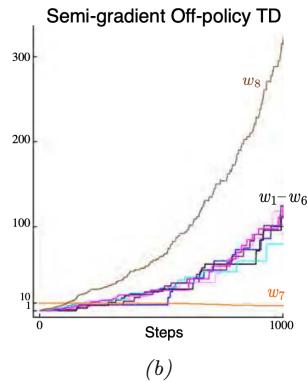
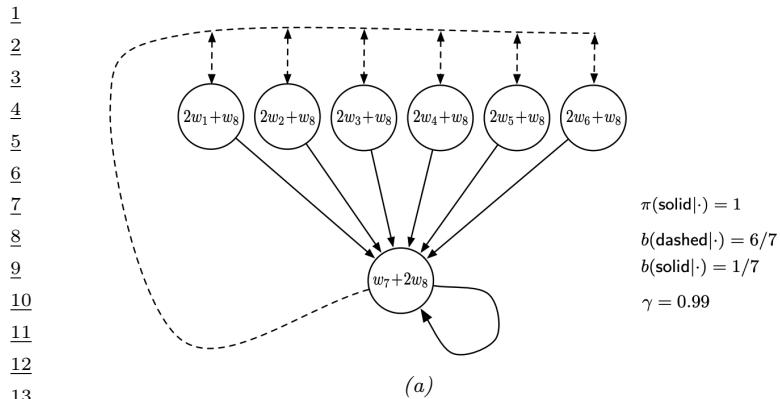


Figure 37.9: (a) ... (b) ... From Figures 11.1 & 11.2 of [SB18]. Used with kind permission of Richard Sutton.

inside the states, with  $\mathbf{w} \in \mathbb{R}^8$ . The target policies  $\pi$  always chooses the solid action in every state. Clearly, the true value function,  $V_\pi(s) = 0$ , can be exactly represented by setting  $\mathbf{w} = \mathbf{0}$ .

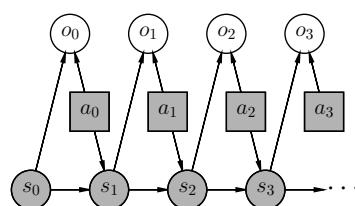
Suppose we use a behavior policy  $b$  to generate a trajectory, which chooses the dashed and solid actions with probabilities  $6/7$  and  $1/7$ , respectively, in every state. If we apply  $\text{TD}(0)$  on this trajectory, the parameters diverge to  $\infty$  (Figure 37.9b), even though the problem appears simple! In contrast, with on-policy data (that is, when  $b$  is the same as  $\pi$ ),  $\text{TD}(0)$  with linear approximation can be guaranteed to converge to a good value function approximate [TR97].

The divergence behavior is demonstrated in many value-based bootstrapping methods, including  $\text{TD}$ ,  $\text{Q}$ -learning, and related approximate dynamic programming algorithms, where the value function is represented either linearly (like the example above) or nonlinearly [Gor95; Ber19]. The root cause of these divergence phenomena is that the contraction property in the tabular case (Equation (36.63)) may no longer hold when  $V$  is approximated by  $V_{\mathbf{w}}$ . An RL algorithm can become unstable when it has these three components: off-policy learning, bootstrapping (for faster learning, compared to MC), function approximation (for generalization in large scale MDPs). This combination is known as **the deadly triad** [SB18]. It highlights another important challenge introduced by off-policy learning, and is a subject of ongoing research (e.g., [van+18; Kum+19a]).

A general way to ensure convergence in off-policy learning is to construct an objective function function, the minimization of which leads to a good value function approximation; see [SB18, Ch. 11] for more background. A natural candidate is the discrepancy between the left and right hand sides of the Bellman optimality equation Equation (36.58), whose unique solution is  $V_*$ . However, the “max” operator is not friendly to optimization. Instead, we may introduce an entropy term to smooth the greedy policy, resulting in a differential square loss in **path consistency learning (PCL)** [Nac+17]:

$$\min_{V, \pi} \mathcal{L}^{\text{PCL}}(V, \pi) \triangleq \mathbb{E} \left[ \frac{1}{2} (r + \gamma V(s') - \lambda \log \pi(a|s) - V(s))^2 \right] \quad (37.59)$$

where the expectation is over  $(s, a, r, s')$  tuples drawn from some off-policy distribution (e.g., uniform over  $\mathcal{D}$ ). Minimizing this loss, however, does not result in the optimal value function and policy in general, due to an issue known as “double sampling” [SB18, Sec. 11.5].



*Figure 37.10: A graphical model for optimal control. States and actions are observed, while optimality variables are not. Adapted from Figure 1b of [Lev18].*

This problem can be mitigated by introducing a dual function in the optimization [Dai+18]

$$\min_{V, \pi} \max_{\nu} \mathcal{L}^{\text{SBEED}}(V, \pi; \nu) \triangleq \mathbb{E} \left[ \nu(s, a) (r + \gamma V(s') - \lambda \log \pi(a|s) - V(s))^2 - \nu(s, a)^2 / 2 \right] \quad (37.60)$$

where  $\nu$  belongs to some function class (e.g., a DNN [Dai+18] or RKHS [FLL19]). It can be shown that optimizing Equation (37.60) forces  $\nu$  to model the Bellman error. So this approach is called **smoothed Bellman error embedding**, or **SBEED**. In both PCL and SBEED, the objective can be optimized by gradient-based methods on parameterized value functions and policies.

## 37.6 Control as inference

In this section, we will discuss another approach to policy optimization, by reducing it to probabilistic inference. This approach allows one to incorporate domain knowledge in modeling, and apply powerful tools from approximate inference (see e.g., Chapter 7), in a consistent and flexible framework, with applications to, for example, robotics [PS07; Tou09; Neu11; Haa+18c] epidemiological decision making [Woo+20], and driver route preference modeling [Zie+08].

### 37.6.1 Maximum entropy reinforcement learning

We now describe a graphical model that exemplifies such a reduction, which results in RL algorithms that are closely related to some discussed previously. The model allows a trade-off between reward and entropy maximization, and recovers the standard RL setting when the entropy part vanishes in the trade-off. Our discussion mostly follows the approach of [Lev18].

Figure 37.10 gives a probabilistic model, which not only captures state transitions as before, but also introduces a new variable,  $o_t$ . This variable is binary, indicating whether the action at time  $t$  is optimal or not, and has the following probability distribution:

$$p(o_t = 1 | s_t, a_t) = \exp(\lambda^{-1} R(s_t, a_t)) \quad (37.61)$$

for some temperature parameter  $\lambda > 0$  whose role will be clear soon. In the above, we have assumed without much loss of generality that  $R(s, a) < 0$ , so that Equation (37.61) gives a valid probability. Furthermore, we can assume a non-informative, uniform action prior,  $p(a_t | s_t)$ , to simplify

the exposition, for we can always push  $p(a_t|s_t)$  into Equation (37.61). Under these assumptions, the likelihood of observing a length- $T$  trajectory  $\tau$ , when optimality achieved in every step, is:

$$\begin{aligned} p(\tau|\mathbf{o}_{0:T-1} = \mathbf{1}) &\propto p(\tau, \mathbf{o}_{0:T-1} = \mathbf{1}) \propto p(s_0) \prod_{t=0}^{T-1} p(o_t = 1|s_t, a_t) p_T(s_{t+1}|s_t, a_t) \\ &= p(s_0) \prod_{t=0}^{T-1} p_T(s_{t+1}|s_t, a_t) \exp\left(\frac{1}{\lambda} \sum_{t=0}^{T-1} R(s_t, a_t)\right) \end{aligned} \quad (37.62)$$

The intuition of Equation (37.62) is clearest when the state transitions are deterministic. In this case,  $p_T(s_{t+1}|s_t, a_t)$  is either 1 or 0, depending on whether the transition is dynamically feasible or not. Hence,  $p(\tau|\mathbf{o}_{0:T-1} = \mathbf{1})$  is either proportional to  $\exp(\lambda^{-1} \sum_{t=0}^{T-1} R(s_t, a_t))$  if  $\tau$  is feasible, or 0 otherwise. Maximizing reward is equivalent to inferring a trajectory with maximum  $p(\tau|\mathbf{o}_{0:T-1} = \mathbf{1})$ .

The optimal policy in this probabilistic model is given by

$$\begin{aligned} p(a_t|s_t, \mathbf{o}_{t:T-1} = \mathbf{1}) &= \frac{p(s_t, a_t|\mathbf{o}_{t:T-1} = \mathbf{1})}{p(s_t|\mathbf{o}_{t:T-1} = \mathbf{1})} = \frac{p(\mathbf{o}_{t:T-1} = \mathbf{1}|s_t, a_t)p(a_t|s_t)p(s_t)}{p(\mathbf{o}_{t:T-1} = \mathbf{1}|s_t)p(s_t)} \\ &\propto \frac{p(\mathbf{o}_{t:T-1} = \mathbf{1}|s_t, a_t)}{p(\mathbf{o}_{t:T-1} = \mathbf{1}|s_t)} \end{aligned} \quad (37.63)$$

The two probabilities in Equation (37.63) can be computed as follows, starting with  $p(o_{T-1} = 1|s_{T-1}, a_{T-1}) = \exp(\lambda^{-1} R(s_{T-1}, a_{T-1}))$ ,

$$p(\mathbf{o}_{t:T-1} = \mathbf{1}|s_t, a_t) = \int_S p(\mathbf{o}_{t+1:T-1} = \mathbf{1}|s_{t+1}) p_T(s_{t+1}|s_t, a_t) \exp(\lambda^{-1} R(s_t, a_t)) ds_{t+1} \quad (37.64)$$

$$p(\mathbf{o}_{t:T-1} = \mathbf{1}|s_t) = \int_A p(\mathbf{o}_{t:T-1} = \mathbf{1}|s_t, a_t) p(a_t|s_t) da_t \quad (37.65)$$

The calculation above is expensive. In practice, we can approximate the optimal policy using a parametric form,  $\pi_\theta(a_t|s_t)$ . The resulted probability of trajectory  $\tau$  now becomes

$$p_\theta(\tau) = p(s_1) \prod_{t=0}^{T-1} p_T(s_{t+1}|s_t, a_t) \pi_\theta(a_t|s_t) \quad (37.66)$$

If we optimize  $\theta$  so that  $D_{\text{KL}}(p_\theta(\tau) \| p(\tau|\mathbf{o}_{0:T-1} = \mathbf{1}))$  is minimized, which can be simplified to

$$D_{\text{KL}}(p_\theta(\tau) \| p(\tau|\mathbf{o}_{0:T-1} = \mathbf{1})) = -\mathbb{E}_{p_\theta} \left[ \sum_{t=0}^{T-1} \lambda^{-1} R(s_t, a_t) + \mathbb{H}(\pi_\theta(s_t)) \right] + \text{const} \quad (37.67)$$

where the constant term only depends on the uniform action prior  $p(a_t|s_t)$ , but not  $\theta$ . In other words, the objective is to maximize total reward, with an entropy regularization favoring more uniform policies. Thus this approach is called **maximum entropy RL**, or **MERL**. If  $\pi_\theta$  can represent all stochastic policies, a softmax version of the Bellman equation can be obtained for Equation (37.67):

$$Q_*(s_t, a_t) = \lambda^{-1} R(s_t, a_t) + \mathbb{E}_{p_T(s_{t+1}|s_t, a_t)} \left[ \log \int_A \exp(Q_*(s_{t+1}, a_{t+1})) da \right] \quad (37.68)$$

1 with the convention that  $Q_*(s_T, a) = 0$  for all  $a$ , and the optimal policy has a softmax form:  
2  $\pi_*(a_t|s_t) \propto \exp(Q_*(s_t, a_t))$ . Note that the  $Q_*$  above is different from the usual optimal  $Q$ -function  
3 (Equation (36.59)), due to the introduction of the entropy term. However, as  $\lambda \rightarrow 0$ , their difference  
4 vanishes, and the softmax policy becomes greedy, recovering the standard RL setting.

5 The **soft actor-critic (SAC)** algorithm [Haa+18b; Haa+18c] is an off-policy actor-critic method  
6 whose objective function is equivalent to Equation (37.67) (by taking  $T$  to  $\infty$ ):  
7

$$\frac{8}{9} J^{\text{SAC}}(\boldsymbol{\theta}) \triangleq \mathbb{E}_{p_{\pi_{\boldsymbol{\theta}}}^{\infty}(s)\pi_{\boldsymbol{\theta}}(a|s)} [R(s, a) + \lambda \mathbb{H}(\pi_{\boldsymbol{\theta}}(s))] \quad (37.69)$$

10 Note that the entropy term has also the added benefit of encouraging exploration.

11 To compute the optimal policy, similar to other actor-critic algorithms, we will work with the “soft”  
12 state- and action-function approximations, parameterized by  $\mathbf{w}$  and  $\mathbf{u}$ , respectively:  
13

$$\frac{14}{15} Q_{\mathbf{w}}(s, a) = R(s, a) + \gamma \mathbb{E}_{p_T(s'|s, a)} [V_{\mathbf{u}}(s', a') - \lambda \log \pi_{\boldsymbol{\theta}}(a'|s')] \quad (37.70)$$

$$\frac{16}{17} V_{\mathbf{u}}(s, a) = \lambda \log \sum_a \exp(\lambda^{-1} Q_{\mathbf{w}}(s, a)) \quad (37.71)$$

18 This induces an improved policy (with entropy regularization):  $\pi_{\mathbf{w}}(a|s) = \exp(\lambda^{-1} Q_{\mathbf{w}}(s, a))/Z_{\mathbf{w}}(s)$ ,  
19 where  $Z_{\mathbf{w}}(s) = \sum_a \exp(\lambda^{-1} Q_{\mathbf{w}}(s, a))$  is the normalization constant. We then perform a soft policy  
20 improvement step to update  $\boldsymbol{\theta}$  by minimizing  $\mathbb{E}[D_{\text{KL}}(\pi_{\boldsymbol{\theta}}(s)\|\pi_{\mathbf{w}}(s))]$  where the expectation may be  
21 approximated by sampling  $s$  from a replay buffer  $D$ .

22 In [Haa+18c; Haa+18b], they show that the SAC method outperforms the off-policy DDPG  
23 algorithm (Section 37.3.5) and the on-policy PPO algorithm (Section 37.3.4) by a wide margin on  
24 various continuous control tasks. For more details, see [Haa+18c].

25 There is a variant of soft actor-critic, which only requires to model the action-value function. It is  
26 based on the observation that both the policy and soft value function can be induced by the soft  
27 action-value function as follows:

$$\frac{28}{29} V_{\mathbf{w}}(s) = \lambda \log \sum_a \exp(\lambda^{-1} Q_{\mathbf{w}}(s, a)) \quad (37.72)$$

$$\frac{30}{31} \pi_{\mathbf{w}}(a|s) = \exp(\lambda^{-1}(Q_{\mathbf{w}}(s, a) - V_{\mathbf{w}}(s))) \quad (37.73)$$

32 We then only need to learn  $\mathbf{w}$ , using approaches similar to DQN (Section 37.2.6). The resulting  
33 algorithm, **soft Q-learning** [SAC17], is convenient if the number of actions is small (when  $\mathcal{A}$  is  
34 discrete), or if the integral in obtaining  $V_{\mathbf{w}}$  from  $Q_{\mathbf{w}}$  is easy to compute (when  $\mathcal{A}$  is continuous).

35 It is interesting to see that algorithms derived in the maximum entropy RL framework bears a  
36 resemblance to PCL and SBEED in Section 37.5.3, both of which were to minimize an objective  
37 function resulting from the entropy-smoothed Bellman equation.  
38

### 39 37.6.2 Other approaches

40 **VIREL** is an alternative model to maximum entropy RL [Fel+19]. Similar to soft actor-critic, it uses  
41 an approximate action-value function,  $Q_{\mathbf{w}}$ , a stochastic policy,  $\pi_{\boldsymbol{\theta}}$ , and a binary optimality random  
42 variable  $o_t$  at time  $t$ . A different probability model for  $o_t$  is used

$$\frac{44}{45} p(o_t = 1|s_t, a_t) = \exp\left(\frac{Q_{\mathbf{w}}(s_t, a_t) - \max_a Q_{\mathbf{w}}(s_t, a)}{\lambda_{\mathbf{w}}}\right) \quad (37.74)$$

46

47

The temperature parameter  $\lambda_w$  is also part of the parameterization, and can be updated from data.  
An EM method can be used to maximize the objective

$$\mathcal{L}(w, \theta) = \mathbb{E}_{p(s)} \left[ \mathbb{E}_{\pi_\theta(a|s)} \left[ \frac{Q_w(s, a)}{\lambda_w} \right] + H(\pi_\theta(s)) \right] \quad (37.75)$$

for some distribution  $p$  that can be conveniently sampled from (e.g., in a replay buffer). The algorithm may be interpreted as an instance of actor-critic. In the E-step, the critic parameter  $w$  is fixed, and the actor parameter  $\theta$  is updated using gradient ascent with stepsize  $\eta_\theta$  (for policy improvement):

$$\theta \leftarrow \theta + \eta_\theta \nabla_\theta \mathcal{L}(w, \theta) \quad (37.76)$$

In the M-step, the actor parameter is fixed, and the critic parameter is updated (for policy evaluation):

$$w \leftarrow w + \eta_w \nabla_w \mathcal{L}(w, \theta) \quad (37.77)$$

Finally, there are other possibilities of reducing optimal control to probabilistic inference, in addition to MERL and VIREL. For example, we may aim to maximize the expectation of the trajectory return  $G$ , by optimizing the policy parameter  $\theta$ :

$$J(\pi_\theta) = \int G(\tau) p(\tau|\theta) d\tau \quad (37.78)$$

It can be interpreted as a pseudo-likelihood function, when the  $G(\tau)$  is treated as probability density, and solved (approximately) by a range of algorithms (see e.g., [PS07; Neu11; Abd+18]). Interestingly, some of these methods have a similar objective as MERL (Equation (37.67)), although the distribution involving  $\theta$  appears in the second argument of  $\text{KL}$ . As discussed in Section 2.9.1, this forward KL-divergence is mode-covering, which in the context of RL is argued to be less preferred than the mode-seeking, reverse KL-divergence used by MERL. For more details and references, see [Lev18].

Control as inference is also closely related to **active inference**; this is based on the **free energy principle** which is popular in neuroscience (see e.g., [Fri09; Buc+17; Ger19; Maz+22]). The FEP is equivalent to using variational inference (see Section 10.1) to perform state estimation (perception) and parameter estimation (learning). In particular, consider a latent variable model with hidden states  $s$ , observations  $y$  and parameters  $\theta$ . Following Section 10.1.1, we define the variational free energy to be  $\mathcal{F}(\theta) = D_{\text{KL}}(q(s, \theta|y) \| p(s, y, \theta))$ . State estimation corresponds to solving  $\min_{q(s|y)} \mathcal{F}(y)$ , and parameter estimation corresponds to solving  $\min_{q(\theta|y)} \mathcal{F}(y)$ , just as in variational Bayes EM (Section 10.2.5). (Minimizing the VFE for certain hierarchical Gaussian models also forms the foundation of predictive coding, which we discuss in Section 8.4.3.)

To extend this to decision making problems we can define the **expected free energy** as  $\bar{\mathcal{F}}(a) = \mathbb{E}_{q(y|a)} [\mathcal{F}(y)]$ , where  $q(y|a)$  is the posterior predictive distribution over observations given actions sequence  $a$ . The connection to control as inference is explained in [Mil+20; WIP20].

### 37.6.3 Imitation learning

In previous sections, an RL agent is to learn an optimal sequential decision making policy so that the total reward is maximized. **Imitation learning** (IL), also known as **apprenticeship learning** and **learning from demonstration** (LfD), is a different setting, in which the agent does not observe

1 rewards, but has access to a collection  $\mathcal{D}_{\text{exp}}$  of trajectories generated by an expert policy  $\pi_{\text{exp}}$ ; that  
2 is,  $\tau = (s_0, a_0, s_1, a_1, \dots, s_T)$  and  $a_t \sim \pi_{\text{exp}}(s_t)$  for  $\tau \in \mathcal{D}_{\text{exp}}$ . The goal is to learn a good policy by  
3 imitating the expert, in the absence of reward signals. IL finds many applications in scenarios where  
4 we have demonstrations of experts (often humans) but designing a good reward function is not easy,  
5 such as car driving and conversational systems. See [Osa+18] for a survey up to 2018.  
6

7

### 8 37.6.3.1 Imitation learning by behavior cloning

9 A natural method is **behavior cloning**, which reduces IL to supervised learning; see [Pom89] for  
10 an early application to autonomous driving. It interprets a policy as a classifier that maps states  
11 (inputs) to actions (labels), and finds a policy by minimizing the imitation error, such as  
12

$$\min_{\pi} \mathbb{E}_{p_{\pi_{\text{exp}}}^{\infty}(s)} [D_{\text{KL}}(\pi_{\text{exp}}(s) \| \pi(s))] \quad (37.79)$$

14

15 where the expectation wrt  $p_{\pi_{\text{exp}}}^{\infty}$  may be approximated by averaging over states in  $\mathcal{D}_{\text{exp}}$ . A challenge  
16 with this method is that the loss does not consider the sequential nature of IL: future state distribution  
17 is not fixed but instead depends on earlier actions. Therefore, if we learn a policy  $\hat{\pi}$  that has a low  
18 imitation error under distribution  $p_{\pi_{\text{exp}}}^{\infty}$ , as defined in Equation (37.79), it may still incur a large  
19 error under distribution  $p_{\hat{\pi}}^{\infty}$  (when the policy  $\hat{\pi}$  is actually run). Further expert demonstrations or  
20 algorithmic augmentations are often needed to handle the distribution mismatch (see e.g., [DLM09;  
21 RGB11]).

22

### 23 37.6.3.2 Imitation learning by inverse reinforcement learning

24 An effective approach to IL is **inverse reinforcement learning (IRL)** or **inverse optimal control**  
25 (IOC). Here, we first infer a reward function that “explains” the observed expert trajectories, and  
26 then compute a (near-)optimal policy against this learned reward using any standard RL algorithms  
27 studied in earlier sections. The key step of reward learning (from expert trajectories) is the opposite  
28 of standard RL, thus called inverse RL [NR00a].

29 It is clear that there are infinitely many reward functions for which the expert policy is optimal,  
30 for example by several optimality-preserving transformations [NHR99]. To address this challenge,  
31 we can follow the maximum entropy principle (Section 2.5.7), and use an energy-based probability  
32 model to capture how expert trajectories are generated [Zie+08]:  
33

$$\begin{aligned} p(\tau) &\propto \exp\left(\sum_{t=0}^{T-1} R_{\theta}(s_t, a_t)\right) \end{aligned} \quad (37.80)$$

34 where  $R_{\theta}$  is an unknown reward function with parameter  $\theta$ . Abusing notation slightly, we denote  
35 by  $R_{\theta}(\tau) = \sum_{t=0}^{T-1} R_{\theta}(s_t, a_t)$  the cumulative reward along the trajectory  $\tau$ . This model assigns  
36 exponentially small probabilities to trajectories with lower cumulative rewards. The partition function,  
37  $Z_{\theta} \triangleq \int_{\mathcal{T}} \exp(R_{\theta}(\tau))$ , is in general intractable to compute, and must be approximated. Here, we can  
38 take a sample-based approach. Let  $\mathcal{D}_{\text{exp}}$  and  $\mathcal{D}$  be the sets of trajectories generated by an expert, and  
39 by some known distribution  $q$ , respectively. We may infer  $\theta$  by maximizing the likelihood,  $p(\mathcal{D}_{\text{exp}}|\theta)$ ,  
40 or equivalently, minimizing the negative log-likelihood loss  
41

$$\mathcal{L}(\theta) = -\frac{1}{|\mathcal{D}_{\text{exp}}|} \sum_{\tau \in \mathcal{D}_{\text{exp}}} R_{\theta}(\tau) + \log \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \frac{\exp(R_{\theta}(\tau))}{q(\tau)} \quad (37.81)$$

42

The term inside the log of the loss is an importance sampling estimate of  $Z$  that is unbiased as long as  $q(\tau) > 0$  for all  $\tau$ . However, in order to reduce the variance, we can choose  $q$  adaptively as  $\theta$  is being updated. The optimal sampling distribution (Section 11.5),  $q_*(\tau) \propto \exp(R_\theta(\tau))$ , is hard to obtain. Instead, we may find a policy  $\hat{\pi}$  which induces a distribution that is close to  $q_*$ , for instance, using methods of maximum entropy RL discussed in Section 37.6.1. Interestingly, the process above produces the inferred reward  $R_\theta$  as well as an approximate optimal policy  $\hat{\pi}$ . This approach is used by **guided cost learning** [FLA16], and found effective in robotics applications.

### 37.6.3.3 Imitation learning by divergence minimization

We now discuss a different, but related, approach to IL. Recall that the reward function depends only on the state and action in an MDP. It implies that if we can find a policy  $\pi$ , so that  $p_\pi^\infty(s, a)$  and  $p_{\pi_{\text{exp}}}^\infty(s, a)$  are close, then  $\pi$  receives similar long-term reward as  $\pi_{\text{exp}}$ , and is a good imitation of  $\pi_{\text{exp}}$  in this regard. A number of IL algorithms find  $\pi$  by minimizing the divergence between  $p_\pi^\infty$  and  $p_{\pi_{\text{exp}}}^\infty$ . We will largely follow the exposition of [GZG19]; see [Ke+19b] for a similar derivation.

Let  $f$  be a convex function, and  $D_f$  the  $f$ -divergence (Section 2.9.1). From the above intuition, we want to minimize  $D_f(p_{\pi_{\text{exp}}}^\infty \| p_\pi^\infty)$ . Then, using a variational approximation of  $D_f$  [NWJ10a], we can solve the following optimization problem for  $\pi$ :

$$\min_{\pi} \max_{\mathbf{w}} \mathbb{E}_{p_{\pi_{\text{exp}}}^\infty(s, a)} [T_{\mathbf{w}}(s, a)] - \mathbb{E}_{p_\pi^\infty(s, a)} [f^*(T_{\mathbf{w}}(s, a))] \quad (37.82)$$

where  $T_{\mathbf{w}} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  is a function parameterized by  $\mathbf{w}$ . The first expectation can be estimated using  $\mathcal{D}_{\text{exp}}$ , as in behavior cloning, and the second can be estimated using trajectories generated by policy  $\pi$ . Furthermore, to implement this algorithm, we often use a parametric policy representation  $\pi_\theta$ , and then perform stochastic gradient updates to find a saddle-point to Equation (37.82).

With different choices of the convex function  $f$ , we can obtain many existing IL algorithms, such as **generative adversarial imitation learning** (GAIL) [HE16b] and **adversarial inverse RL** (AIRL) [FLL18], as well as new algorithms like **f-Divergence Max-Ent IRL** (*f*-MAX) and **forward adversarial inverse RL** (FAIRL) [GZG19; Ke+19b].

Finally, the algorithms above typically require running the learned policy  $\pi$  to approximate the second expectation in Equation (37.82). In risk- or cost-sensitive scenarios, collecting more data is not always possible. Instead, we are in the off-policy IL setting, working with trajectories collected by some policy other than  $\pi$ . Hence, we need to correct the mismatch between  $p_\pi^\infty$  and the off-policy trajectory distribution, for which techniques from Section 37.5 can be used. An example is **ValueDICE** [KNT20], which uses a similar distribution correction method of DualDICE (Section 37.5.2).



# 38 Causality

*This chapter was written by Victor Veitch and Alex D'Amour.*

## 38.1 Introduction

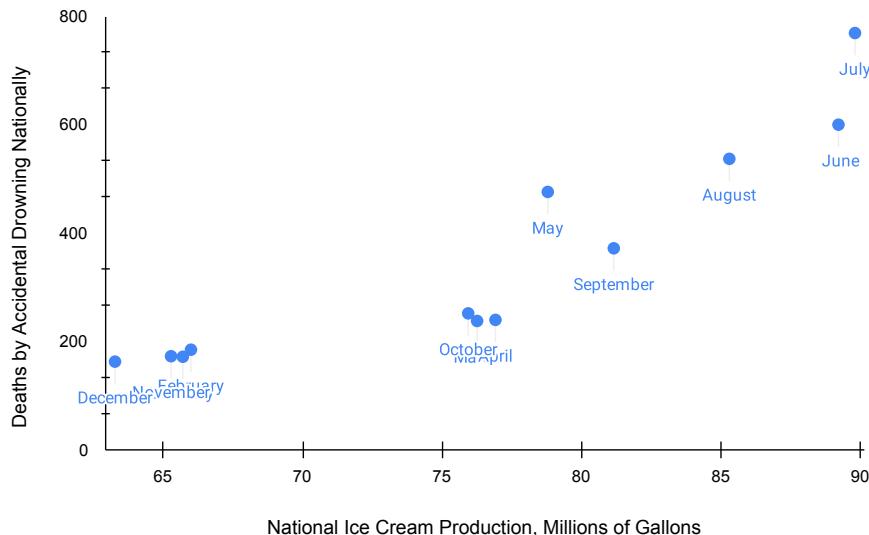
### 38.1.1 Why is causality different than other forms of ML?

The bulk of machine learning considers relationships between observed variables with the goal of summarizing these relationships in a manner that allows predictions on similar data. However, for many problems, our main interest is to predict how system would change if it were observed under different conditions. For instance, in healthcare, we are interested in whether a patient will recover if given a certain treatment (as opposed to whether treatment and recovery are associated in the observed data). **Causal inference** addresses how to formalize such problems, determine whether they can be solved, and, if so, how to solve them. This chapter covers the fundamentals of this subject. Code examples for the discussed methods are available at <https://github.com/vveitch/causality-tutorials>.

To make the gap between observed data modeling and causal inference concrete, consider the relationships depicted in Figure 38.1 and Figure 38.2. Figure 38.1 shows the relationship between deaths by drowning and ice cream production in the United States in 1931 (the pattern holds across most years). Figure 38.2 shows the relationship between smoking and lung cancer across various countries. In each case, there is a strong positive association. Faced with this association, we might ask: could we reduce drowning deaths by banning ice cream? Could we reduce lung cancer by banning cigarettes? We intuitively understand that these interventional questions have different answers, despite the fact that the observed associations are similar. Determining the causal effect of some intervention in the world requires some such causal hypothesis about the world.

For concreteness, consider three possible explanations for the association between ice cream and drowning. Perhaps eating ice cream does cause people to drown—due to stomach cramps or similar. Or, perhaps, drownings increase demand for ice cream—the survivors eat huge quantities of ice cream to handle their grief. Or, the association may be due (at least in part) to a common cause: warm weather makes people more likely to eat ice cream and more likely to go swimming (and, hence, to drown). Under all three scenarios, we can observe exactly the same data, but the implications for an ice cream ban are very different. Hence, answering questions about what will happen under an intervention requires us to incorporate some causal knowledge of the world—e.g., which of these scenarios is plausible?

1  
2  
3 Ice Cream Production and Deaths By Drowning in USA, 2020  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19



- **Causal Estimands:** The first step is to formally define the quantities we want to estimate. These are summaries of how the world would change under intervention, rather than summaries of the world as it has already been observed. E.g., we want to formalize “The expected number of drownings in the United States if we ban ice cream”.
- **Identification:** The next step is to identify the causal estimands with quantities that can, in principle, be estimated from observational data. This step involves codifying our causal knowledge of the world and translating this into a statement such as, “The causal effect is equal to the expected number of drownings after adjusting for month”. This step tells us what causal questions we could answer with perfect knowledge of the observed data distribution.
- **Estimation:** Finally, we must estimate the observable quantity using a finite data sample. The form of causal estimands favors certain efficient estimation procedures that allow us to exploit non-parametric (e.g., machine learning) predictive models.

In this chapter, we'll mainly focus on the estimation of the causal effect of an intervention averaged over all members of a population, known as the **Average Treatment Effect** or **ATE**. This is the

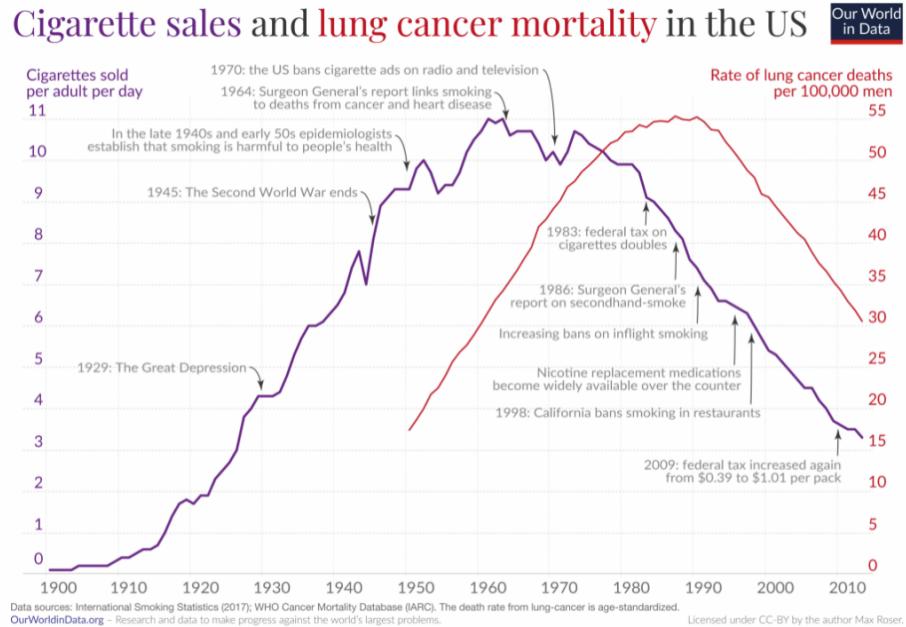


Figure 38.2: Smoking is strongly associated with lung cancer. Figure by Max Roser, [ourworldindata.org/smoking-big-problem-in-brief](http://ourworldindata.org/smoking-big-problem-in-brief)

most common problem in applied causal inference work. It is in some sense the simplest problem, and will allow us to concretely explain the use and importance of the fundamental causal concepts. These causal concepts include structural causal models, causal graphical models, the do-calculus, and efficient estimation using influence function techniques. This problem is also useful for understanding the role that standard predictive modeling and machine learning play in estimating causal quantities.

## 38.2 Causal Formalism

In causal inference, the goal is to use data to learn about how the outcome in the world would change under intervention. In order to make such inferences, we must also make use of our causal knowledge of the world. This requires a formalism that lets us make the notion of intervention precise and lets us encode our causal knowledge as assumptions.

### 38.2.1 Structural Causal Models

Consider a setting in which we observe four variables from a population of people:  $A_i$ , an indicator of whether or not person  $i$  smoked at a particular age,  $Y_i$ , an indicator of whether or not person  $i$  developed lung cancer at a later age,  $H_i$ , a “health consciousness” index that measures a person’s health-consciousness (perhaps constructed from a set of survey responses about attitudes toward

1 health), and  $G_i$ , an indicator for whether the person has a genetic predisposition towards cancer.  
2 Suppose we observe a dataset of these variables drawn independently and identically from a population,  
3  $(A_i, Y_i, H_i) \stackrel{\text{iid}}{\sim} P^{\text{obs}}$ , where “obs” stands for “observed”.

4 In standard practice, we model data like these using probabilistic models. Notably, there are many  
5 different ways to specify a probabilistic model for the same observed distribution. For example, we  
6 could write a probabilistic model for  $P^{\text{obs}}$  as

$$\underline{8} \quad A \sim P^{\text{obs}}(A) \quad (38.1)$$

$$\underline{9} \quad H|A \sim P^{\text{obs}}(H|A) \quad (38.2)$$

$$\underline{10} \quad Y|A, H \sim P^{\text{obs}}(Y|H, A) \quad (38.3)$$

$$\underline{11} \quad G|A, H, Y \sim P^{\text{obs}}(G|A, H, Y) \quad (38.4)$$

12 This is a valid factorization, and sampling variables in this order would yield valid samples from the  
13 joint distribution  $P^{\text{obs}}$ . However, this factorization does not map well to a mechanistic understanding  
14 of how these variables are causally related in the world. In particular, it is perhaps more plausible  
15 that health-consciousness  $H$  causally precedes smoking status  $A$ , since a person’s health-consciousness  
16 would influence their decision to smoke.

17 These intuitions about causal ordering are intimately tied to the notion of intervention. Here, we  
18 will focus on a notion of intervention that can be represented in terms of “structural” models that  
19 describe mechanistic relationships between variables. The fundamental objects that we will reason  
20 about are **structural causal models**, or SCM’s. SCM’s resemble probabilistic models, but they  
21 encode additional assumptions. Specifically, SCM’s serve two purposes: they describe a probabilistic  
22 model *and* they provide semantics for transforming the data-generating process through intervention.

23 Formally, SCM’s describe a mechanistic data generating process with an ordered sequence of  
24 equations that resemble assignment operations in a program. Each variable in a system is determined  
25 by combining other modeled variables (the causes) with exogenous “noise” according to some  
26 (unknown) deterministic function. For instance, a plausible SCM for  $P^{\text{obs}}$  might be

$$\underline{27} \quad G \leftarrow f_G(\xi_0) \quad (38.5)$$

$$\underline{28} \quad H \leftarrow f_H(\xi_1) \quad (38.6)$$

$$\underline{29} \quad A \leftarrow f_A(H, \xi_2) \quad (38.7)$$

$$\underline{30} \quad Y \leftarrow f_Y(G, H, A, \xi_3) \quad (38.8)$$

31 where the (unknown) functions  $f$  are fixed, and the variables  $\xi$  are unmeasured causes, modeled  
32 as independent random “noise” variables. Conceptually, the functions  $f_G, f_H, f_A, f_Y$  describe deter-  
33 ministic physical relationships in the real world, while the variables  $\xi$  are hidden causes that are  
34 sufficient to distinguish each unit  $i$  in the population. Because we assume that each observed unit  $i$   
35 is drawn at random from the population, we model  $\xi$  as random noise.

36 SCM’s imply probabilistic models, but not the other way around. For example, our example SCM  
37 implies probabilistic model for the observed data based on the factorization  $P^{\text{obs}}(G, H, A, Y) =$   
38  $P^{\text{obs}}(G)P^{\text{obs}}(H)P^{\text{obs}}(A | H)P^{\text{obs}}(Y | A, H)$ . Thus, we could sample from the SCM in the same way  
39 we would from a probabilistic model: draw a set of noise variables  $\xi$  and evaluate each assignment  
40 operation in the SCM in order.

41 Beyond the probabilistic model, an SCM encodes additional assumptions about the effects of  
42 interventions. In an SCM, interventions are represented by replacing assignment statements. For  
43

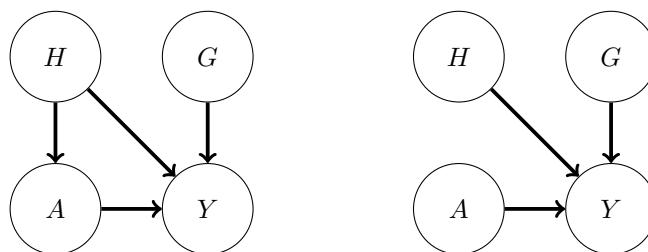


Figure 38.3: (Left) Causal graph illustrating relationships between smoking  $A$ , cancer  $Y$ , health consciousness  $H$ , and genetic cancer pre-disposition  $G$ . (Right) “Mutilated” causal graph illustrating relationships under an intervention on smoking  $A$ .

example, if we were interested in the distribution of  $Y$  in the hypothetical scenario that smoking were eliminated, we could set the second line of the SCM to be  $A \leftarrow 0$ . Because the  $f$  functions in the SCM are assumed to be invariant mechanistic relationships, the SCM encodes the assumption that this edited SCM generates data that we would see if we really applied this intervention in the world. Thus, the ordering of statements in an SCM are load-bearing: they imply substantive assumptions about how the world changes in response to interventions. This is in contrast to more standard probabilistic models where variables can be rearranged by applications of Bayes Rule without changing the substantive implications of the model.

We note that structural causal model may not incorporate all possible notions of causality. For example, laws based on conserved quantities or equilibria—e.g., the ideal gas law—do not trivially map to SCMs, though these are fundamental in disciplines such as physics and economics. Nonetheless, we will confine our discussion to SCMs.

### 38.2.2 Causal DAGs

SCM’s encode many details about the assumed generative process of a system, but often it is useful to reason about causal problems at a higher level of abstraction. In particular, it is often useful to separate the causal structure of a problem from the particular functional form of those causal relationships. **Causal graphs** provide this level of abstraction. A causal graph specifies which variables causally affect other variables, but leaves the parametric form of the structural equations  $f$  unspecified. Given an SCM, the corresponding causal graph can be drawn as follows: for each line of the SCM, draw arrows from the variables on the right hand side to variables on the left hand side. The causal DAG for our smoking-cancer example is shown in Figure 38.3. In this way, causal DAGs are related to SCMs in the same way that probabilistic graphical models (PGMs) are related to probabilistic models.

In fact, in the same way that SCMs imply a probabilistic model, causal DAGs imply a PGM. Functionally, causal graphs behave as probabilistic graphical models (Chapter 4). They imply conditional independence relationships between the variables in the observed data in same way. They obey the Markov property: If  $X \leftarrow Y \rightarrow Z$  then  $X \perp\!\!\!\perp Z|Y$ ; recall d-separation (Section 4.2.3.1). Additionally, if  $X \rightarrow Y \leftarrow Z$  then, usually,  $X \not\perp\!\!\!\perp Z|Y$  (even if  $X$  and  $Z$  are marginally independent). In this case,  $Y$  is called a **collider** for  $X$  and  $Z$ .

1 Conceptually, the difference between causal DAGs and PGMs is that probabilistic graphical models  
2 encode our assumptions about statistical relationships, whereas causal graphs encode our (stronger)  
3 assumptions about causal relationships. Such causal relationships can be used to derive how statistical  
4 relationships would change under intervention.

5 Causal graphs also allow us to reason about the causal and non-causal origins of statistical  
6 dependencies in observed data without specifying a full SCM. In a causal graph, two variables—say,  
7  $A$  and  $D$ —can be statistically associated in different ways. First, there can be a directed path from  
8 (ancestor)  $A$  to (descendant)  $D$ . In this case,  $A$  is a causal ancestor of  $D$  and interventions on  $A$  will  
9 propagate through to change  $D$ ;  $P(D|do(A = a)) \neq P(D|do(A = a'))$ . For example, smoking is a  
10 causal ancestor of cancer in our example. Alternatively,  $A$  and  $D$  could share a common cause—there  
11 is some variable  $C$  such that there is a directed path from  $C$  to  $A$  and from  $C$  to  $D$ . If  $A$  and  $D$   
12 are associated only through such a path then interventions on  $A$  will not change the distribution of  
13  $D$ . However, it is still the case that  $P(D|A = a) \neq P(D|A = a')$ —observing different values of  $A$   
14 changes our guess for the value of  $D$ . The reason is that  $A$  carries information about  $C$ , which carries  
15 information about  $D$ . For example, suppose we lived in a world where there was no effect of smoking  
16 on developing cancer (e.g., everybody vapes), there would nevertheless be an association between  
17 smoking and cancer because of the path  $A \leftarrow H \rightarrow Y$ . The existence of such “backdoor paths” is one  
18 core reason that statistical and causal association are not the same. Of course, more complicated  
19 variants of these associations are possible—e.g.,  $C$  is itself only associated with  $A$  through a backdoor  
20 path—but this already captures the key distinction between causal and non-causal paths.

21 Recall that our aim in introducing SCMs and causal graphs is to enable us to formalize our causal  
22 knowledge of the world and to make precise what interventional quantities we'd like to estimate.  
23 Writing down a causal graph gives a simple formal way to encode our knowledge of the causal  
24 structure of a problem. Usefully, this causal structure is sufficient to directly reason about the  
25 implications of interventions without fully specifying the underlying SCM. The key observation is that  
26 if a variable  $A$  is intervened on then, after intervention, none of the other variables are causes of  $A$ .  
27 That is, when we replace a line of an SCM with a statement directly assigning a variable a particular  
28 value, we cut off all dependencies that variable had on its causal parents. Accordingly, in the causal  
29 graph, the intervened on variable has no parents. This leads us to the **graph surgery** notion of  
30 intervention: an intervention that sets  $A$  to  $a$  is the operation that deletes all incoming edges to  $A$  in  
31 the graph, and then conditions on  $A = a$  in the resulting probability distribution (which is defined  
32 by the conditional independence structure of the post-surgery graph). We'll use Pearl's do notation  
33 to denote this operation.  $P(\mathbf{X}|do(A = a))$  is the distribution of  $\mathbf{X}$  given  $A = a$  under the mutilated  
34 graph that results from deleting all edges going into  $A$ . Similarly,  $\mathbb{E}[\mathbf{X}|do(A = a)] \triangleq \mathbb{E}_{P(\mathbf{X}|do(A=a))}[\mathbf{X}]$ .  
35 Thus, we can formalize statements such as “The average effect of receiving drug  $A$ ” as

$$\text{ATE} = E[Y|do(A = 1)] - \mathbb{E}[Y|do(A = 0)], \quad (38.9)$$

36 where ATE stands for Average Treatment Effect.

37 For concreteness, consider our running example. We contrast the distribution that results by  
38 conditioning on  $A$  with the distribution that results from intervening on  $A$ :

$$P(Y, H, G|A = a) = P(Y|H, G, A = a)P(G)P(H|A = a) \quad (38.10)$$

$$P(Y, H, G|do(A = a)) = P(Y|H, G, A = a)P(G)P(H) \quad (38.11)$$

39 The key difference between these two distributions is that the standard conditional distribution  
40 describes a population where health consciousness  $H$  has the distribution that we observe among  
41

1 individuals with smoking status  $A = a$ , while the interventional distribution described a population  
 2 where health consciousness  $H$  follows the marginal distribution among all individuals. For example,  
 3 we would expect  $P(H | A = \text{smoker})$  to put more mass on lower values of  $H$  than the marginal  
 4 health consciousness distribuiton than the marginal distribution  $P(H)$ , which would also include  
 5 non-smokers. The intervention distribution thus incorporates a hypothesis of how smoking would  
 6 affect the subpopulation individuals who tend to be too health conscious to smoke in the observed  
 7 data.  
 8

### 10 38.2.3 Identification

11 A central challenge in causal inference is that many different SCM's can produce identical distributions  
 12 of observed data. This means that, on the basis of observed data alone, we cannot uniquely identify  
 13 the SCM that generated it. This is true no matter how large of a data sample is available to us.

14 For example, consider the setting where there is a treatment  $A$  that may or may not have an  
 15 effect on outcome  $Y$ , and where both the treatment and outcome are known to be affected by  
 16 some *unobserved* common binary cause  $U$ . Now, we might be interested in the causal estimand  
 17  $E[Y|\text{do}(A = 1)]$ . In general, we can't learn this quantity from the observed data. The problem is  
 18 that, we can't tell apart the case where the treatment has a strong effect from the case where the  
 19 treatment has no effect, but  $U = 1$  both causes people to tend to be treated and causes increases the  
 20 probability of a positive outcome. The same observation shows we can't learn the (more complicated)  
 21 interventional distribution  $P(Y|\text{do}(A = 1))$  (if we could learn this, then we'd get the average effect  
 22 automatically).

23 Thus, an important part of causal inference is to augment the observed data with knowledge about  
 24 the underlying causal structure of the process under consideration. Often, these assumptions can  
 25 narrow the space of SCM's sufficiently so that there is only one value of the causal estimand that is  
 26 compatible with the observed data. We say that the causal estimand is **identified** or **identifiable**  
 27 under a given set of assumptions if those assumptions are sufficient to provide a unique answer.  
 28 There are many different sets of sufficient conditions that yield identifiable causal effects; we call  
 29 each set of sufficient conditions an **identification strategy**.

30 Given a set of assumptions about the underlying SCM, the most common way to show that a  
 31 causal estimand is identified is by construction. Specifically, if the causal estimand can be written  
 32 entirely in terms of observable probability distributions, then it is identified. We call such a function of  
 33 observed distributions a **statistical estimand**. Once such a statistical estimand has been recovered,  
 34 we can then construct and analyze an estimator for that quantity using standard statistical tools.  
 35 As an example of a statistical estimand, in the SCM above, it can be shown the ATE as defined in  
 36 Equation (38.9), is equal to the following statistical estimand  
 37

$$38 \quad \text{ATE} \stackrel{(*)}{=} \tau^{\text{ATE}} \triangleq \mathbb{E}[\mathbb{E}[Y|H, A = 1] - \mathbb{E}[Y|H, A = 0]], \quad (38.12)$$

39 where the equality (\*) only holds because of some specific properties of the SCM. Note that the RHS  
 40 above only involves conditional expectations between observed variables (there are no do operators),  
 41 so  $\tau^{\text{ATE}}$  is only a function of observable probability distributions.

42 There are many kinds of assumptions we might make about the SCM governing the process under  
 43 consideration. For example, the following are assertions we might make about the system in our  
 44 running example:

1 1. The probability of developing cancer is additive on the logit scale in  $A$ ,  $G$ , and  $H$  (i.e., logistic  
2 regression is a well-specified model).

3 2. For each individual, smoking can never decrease the probability of developing cancer.

4 3. Whether someone smokes is influenced by their health consciousness  $H$ , but not by their genetic  
5 predisposition to cancer  $G$ .

6 These assumptions range from strong parametric assumptions fully specifying the form of the SCM  
7 equations, to non-parametric assumptions that only specify what the inputs to each equation are,  
8 leaving the form fully unspecified. Typically, assumptions that fully specify the parametric form are  
9 very strong, and would require far more detailed knowledge of the system under consideration than  
10 we actually have. The goal in identification arguments is to find a set of assumptions that are weak  
11 enough that they might be plausibly true for the system under consideration, but which are also  
12 strong enough to allow for identification of the causal effect.

13 If we are not willing to make any assumptions about the functional form of the SCM, then our  
14 assumptions are just about which variables affect (and do not affect) the other variables. In this sense,  
15 such which-affects-which assumptions are minimal. These assumptions are exactly the assumptions  
16 captured by writing down a (possibly incomplete) causal DAG, showing which variables are parents  
17 of each other variable. The graph may be incomplete because we may not know whether each possible  
18 edge is present in the physical system. For example, we might be unsure whether the gene  $G$  actually  
19 has a causal effect on health consciousness  $H$ . It is natural to ask to what extent we can identify  
20 causal effects only on the basis of partially specified causal DAGs. It turns out much progress can be  
21 made based on such non-parametric assumptions; we discuss this in detail in Section 38.8.

22 We will also discuss certain assumptions that cannot be encoded in a causal graph, but that are  
23 still weaker than assuming that full functional forms are known. For example, we might assume that  
24 the outcome is affected additively by the treatment and any confounders, with no interaction terms  
25 between them. These weaker assumptions can enable causal identification even when assuming the  
26 causal graph alone does not.

27 It is worth emphasizing that every causal identification strategy relies on assumptions that have  
28 some content that cannot be validated in the observed data. This follows directly from the ill-posedness  
29 of causal problems: if the assumptions used to identify causal quantities could be validated, that  
30 would imply that the causal estimand was identifiable from the observed data alone. However, since  
31 we know that there are many values of the causal estimand that are compatible with observed data,  
32 it follows that the assumptions in our identification strategy must have unobservable implications.  
33

### 34 38.2.4 Counterfactuals and the Causal Hierarchy

35 Structural causal models let us formalize and study a hierarchy of different kinds of query about the  
36 system under consideration. The most familiar is observational queries: questions that are purely  
37 about statistical associations (e.g., “Are smoking and lung cancer associated in the population this  
38 sample was drawn from?”). Next is interventional queries: questions about causal relationships at  
39 the population level (e.g., “How much does smoking increase the probability of cancer in a given  
40 population?”). The rest of this chapter is focused on the definition, identification, and estimation of  
41 interventional queries. Finally, there are counterfactual queries: questions about causal relationships  
42 at the level of specific individuals, had something been different (e.g., “Would Alice have developed  
43 cancer had she not smoked?”). This causal hierarchy was popularized by [Pea09a, Ch. 1].  
44

45

**Interventional queries** concern the prospective effect of an intervention on an outcome; for example, if we intervene and prevent a randomly sampled individual from smoking, what is the probability they develop lung cancer? Ultimately, the probability statement here is about our uncertainty about the “noise” variables  $\xi$  in the SCM. These are the unmeasured factors specific to the randomly selected individual. The distribution is determined by the population from which that individual is sampled. Thus, interventional queries are statements about populations. Interventional queries can be written in terms of conditional distributions using **do-notation**, e.g.

$$P(Y|\text{do}(A = 0)) \quad (38.13)$$

where conditioning on the “do” clause means that the line defining  $A$  in the underlying SCM was changed to an intervention setting  $A \leftarrow 0$ . In our example, this represents the distribution of lung cancer outcomes for an individual selected at random and prevented from smoking.

**Counterfactual queries** concern how an observed outcome might have been different had an intervention been applied in the past. Counterfactual queries are often framed in terms of attributing a given outcome to a particular cause. For example, would Alice have developed cancer had she not smoked? Did most smokers with lung cancer develop cancer because they smoked? Counterfactual queries are so called because they require a comparison of counterfactual outcomes within individuals. In the formalism of SCM’s, counterfactual outcomes for an individual  $i$  are generated by running the same values of  $\xi_i$  through differently intervened SCM’s. Counterfactual outcomes are often written in terms of *potential outcomes* notation. In our running smoking example, this would look like:

$$Y_i(a) \triangleq f_Y(G_i, H_i, a, \xi_{3,i}). \quad (38.14)$$

That is,  $Y_i(a)$  is the outcome we would have seen had  $A$  been set to  $a$  while all of  $G_i, H_i, \xi_{3,i}$  were kept fixed.

It is important to understand what distinguishes interventional and fundamentally counterfactual queries. Just because a query can be written in terms of potential outcomes does not make it a counterfactual query. For example, the average treatment effect, which is the canonical interventional query, is easy to write in potential outcomes notation:

$$\text{ATE} = \mathbb{E}[Y_i(1) - Y_i(0)]. \quad (38.15)$$

Instead, the key dividing line between counterfactual and interventional queries is whether the query requires knowing the joint distribution of potential outcomes within individuals, or whether marginal distributions of potential outcomes across individuals will suffice. An important signature of a counterfactual query is conditioning on the value of one potential outcome. For example, “the lung cancer rate among smokers who developed cancer, had they not smoked” is a counterfactual query, and can be written as:

$$\mathbb{E}[Y_i(0) | Y_i(1) = 1, A_i = 1] \quad (38.16)$$

Answering this query requires knowing how individual-level cancer outcomes are related (through  $\xi_{3,i}$ ) across the worlds where the each individual  $i$  did and did not smoke. Notably, this query cannot be rewritten using do-notation, because it requires a distinction between  $Y(0)$  and  $Y(1)$  while the ATE can:  $\mathbb{E}[Y | \text{do}(A = 1)] - \mathbb{E}[Y | \text{do}(A = 0)]$ .

<sup>1</sup> Counterfactual queries require categorically more assumptions for identification than interventional  
<sup>2</sup> ones. For identifying interventional queries, knowing the DAG structure of an SCM is often sufficient,  
<sup>3</sup> while for counterfactual queries, some assumptions about the functional forms in the SCM are  
<sup>4</sup> necessary. This is because only one potential outcome is ever observed for each individual, so the  
<sup>5</sup> dependence between potential outcomes within individuals is not observable. For example, the data  
<sup>6</sup> in our running example provide no information on how individual-level smoking and non-smoking  
<sup>7</sup> cancer risk are related. Thus, answering a question like “Did smokers who developed cancer have lower  
<sup>8</sup> non-smoking cancer risk than smokers who did not develop cancer?”, requires additional assumptions  
<sup>9</sup> about how characteristics encoded in  $\xi_i$  are translated to cancer outcomes. To answer this question  
<sup>10</sup> without such assumptions, we would need to observe smokers who developed cancer in the alternate  
<sup>11</sup> world where they did not smoke. Because they compare how individuals would have turned out under  
<sup>12</sup> different generating processes, counterfactual queries are often referred to as “cross-world” quantities.  
<sup>13</sup> On the other hand, interventional queries only require understanding the marginal distributions of  
<sup>14</sup> potential outcomes  $Y_i(0)$  and  $Y_i(1)$  across individuals; thus, no cross-world information is necessary  
<sup>15</sup> at the individual level.

<sup>16</sup> We conclude this section by noting that counterfactual outcomes and potential outcomes notation  
<sup>17</sup> are often conceptually useful, even if they are not used to explicitly answer counterfactual queries.  
<sup>18</sup> Many causal queries are more intuitive to formalize in terms of potential outcomes. E.g., “Would I  
<sup>19</sup> have smoked if I was more health conscious?” may be more intuitive than “Would a randomly sampled  
<sup>20</sup> individual from the same population have smoked had they been subject to an intervention that made  
<sup>21</sup> them more health conscious?”. In fact, some schools of causal inference use potential outcomes, rather  
<sup>22</sup> than DAGs, as their primary conceptual building block [See IR15]. Causal graphs and potential  
<sup>23</sup> outcomes both provide ways to formalize interventional queries and causal assumptions. Ultimately,  
<sup>24</sup> these are mathematically equivalent. Nevertheless, practically, they have different strengths. The  
<sup>25</sup> main advantage of potential outcomes is that counterfactual statements often map more directly to  
<sup>26</sup> our mechanistic understanding of the world. This can make it easier to articulate causal desiderata  
<sup>27</sup> and causal assumptions we may wish to use. On the other hand, the potential outcomes notation  
<sup>28</sup> does not automatically distinguish between interventional and counterfactual queries. Additionally,  
<sup>29</sup> causal graphs often give an intuitive and easy way of articulating assumptions about structural  
<sup>30</sup> causal models involving many variables—potential outcomes get quickly unwieldy. In short: both  
<sup>31</sup> formalizations have distinct advantages, and those advantages are simply about how easy it is to  
<sup>32</sup> translate our causal understanding of the world into crisp mathematical assumptions.  
<sup>33</sup>

<sup>34</sup>

<sup>35</sup>

### <sup>36</sup> 38.3 Randomized Control Trials

<sup>37</sup>

<sup>38</sup> We now turn to the business of estimating causal effects from data. We begin with **randomized**  
<sup>39</sup> **control trials**, which are experiments designed to make the causal concerns as simple as possible.

<sup>40</sup> The simplest situation for causal estimation is when there are no common causes of  $A$  and  $Y$ . The  
<sup>41</sup> world is rarely so obliging as to make this the case. However, sometimes we can design an experiment  
<sup>42</sup> to enforce the no-common-causes structure. In randomized control trials we assign each participant  
<sup>43</sup> to either the treatment or control group at random. Because random assignment does not depend on  
<sup>44</sup> any property of the units in the study, there are no causes of treatment assignment, and hence also  
<sup>45</sup> no common causes of  $Y$  and  $A$ .

<sup>46</sup> In this case, it’s straightforward to see that  $P(Y|do(A = a)) = P(Y|a)$ . This is essentially by  
<sup>47</sup>

definition of the graph surgery: since  $A$  has no parents, the mutilated graph is the same as the original graph. Indeed, the graph surgery definition is chosen to make this true: any sensible formalization of causality should have this identification result.

It is common to use RCTs to study the average treatment effect,

$$\text{ATE} = E[Y|\text{do}(A = 1)] - E[Y|\text{do}(A = 0)]. \quad (38.17)$$

This is the expected difference between being assigned treatment and assigned no treatment for a randomly chosen member of the population. It's easy to see that in an RCT this causal quantity is identified as a parameter  $\tau^{\text{RCT}}$  of the observational distribution:

$$\tau^{\text{RCT}} = E[Y|A = 1] - E[Y|A = 0].$$

Then, a natural estimator is:

$$\hat{\tau}^{\text{RCT}} \triangleq \frac{1}{n_A} \sum_{i:A_i=1} Y_i - \frac{1}{n - n_A} \sum_{i:A_i=0} Y_i, \quad (38.18)$$

where  $n_A$  is the number of units who received treatment. That is, we estimate the average treatment effect as the difference between the average outcome of the treated group and the average outcome of the untreated (control) group.<sup>1</sup>

Randomized control trials are the gold standard for estimating causal effects. This is because we know *by design* that there are no confounders that can produce alternative causal explanations of the data. In particular, the assumption of the triangle DAG—there are no unobserved confounders—is enforced by design. However, there are limitations. Most obviously, randomized control trials are sometimes infeasible to conduct. This could be due to expense, regulatory restrictions, or more fundamental difficulties (e.g., in developmental economics, the response of interest is sometimes collected decades after treatment). Additionally, it may be difficult to ensure that the participants in an RCT are representative of the population where the treatment will be deployed. For instance, participants in drug trials may skew younger and poorer than the population of patients who will ultimately take the drug.

## 38.4 Confounder Adjustment

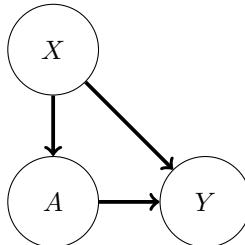
We now turn to the problem of estimating causal effects using observational (i.e., not experimental) data. The most common application of causal inference is estimating the average treatment effect (ATE) of an intervention. The ATE is also commonly called the **average causal effect**, or ACE. Here, we focus on the important special case where the treatment  $A$  is binary, and we observe the outcome  $Y$  as well as a set of common causes  $X$  that influence both  $A$  and  $Y$ .

### 38.4.1 Causal Estimand, Statistical Estimand, and Identification

Consider a problem where we observe treatment  $A$ , outcome  $Y$ , and covariates  $X$ , which are drawn i.i.d. from some unknown distribution  $P$ . We wish to learn the average treatment effect: the expected

<sup>1</sup> There is a literature on efficient estimation of causal effects in RCT's going back to Fisher [Fis25] that employ more sophisticated estimators. See also Lin [Lin13a] and Bloniarz et al. [Blo+16] for more modern treatments.

1  
2  
3  
4  
5  
6  
7  
8  
9  
10



11Figure 38.4: A causal DAG illustrating a situation where treatment  $A$  and outcome  $Y$  are both influenced by  
12observed confounders  $X$ .

13  
14

15difference between being assigned treatment and assigned no treatment for a randomly chosen member  
16of the population. Following the discussion in the introduction, there are three steps to learning this  
17quantity: mathematically formalize the causal estimand, give conditions for the causal estimand to  
18be identified as a statistical estimand, and, finally, estimate this statistical estimand from data. We  
19now turn to the first two steps.

20 The average treatment effect is defined to be the difference between the average outcome if we  
21intervened and set  $A$  to be 0, versus the average outcome if we intervened and set  $A$  to be 1. Using  
22the do notation, we can write this formally as

23

$$\text{ATE} = \mathbb{E}[Y|\text{do}(A = 1)] - \mathbb{E}[Y|\text{do}(A = 0)]. \quad (38.19)$$

25

26 The next step is to articulate sufficient conditions for the ATE to be identified as a statistical  
27estimand (a parameter of distribution  $P$ ). The key issue is the possible presence of **confounders**.  
28Confounders are “common cause” variables that affect both the treatment and outcome. When there  
29are confounding variables in observed data, the sub-population of people who are observed to have  
30received one level of the treatment  $A$  will differ from the rest of the population in ways that are  
31relevant to their observed  $Y$ . For example, there is a strong positive association between horseback  
32riding in childhood (treatment) and healthiness as an adult (outcome) [RB16]. However, both of these  
33quantities are influenced by wealth  $X$ . The population of people who rode horses as children ( $A = 1$ )  
34is wealthier than the population of people who did not. Accordingly, horseback-riding population  
35will have better health outcomes even if there is no actual causal benefit of horseback riding for adult  
36health.

37 We’ll express the assumptions required for causal identification in the form of a causal DAG.  
38Namely, we consider the simple triangle DAG in Figure 38.4, where the treatment and outcome  
39are influenced by *observed* confounders  $X$ . It turns out that the assumption encoded by this DAG  
40suffices for identification. To understand why this is so, recall that the target causal effect is defined  
41according to the distribution we would see if the edge from  $X$  to  $A$  was removed (that’s the meaning  
42of do). The key insight is that because the intervention only modifies the relationship between  $X$   
43and  $A$ , the structural equation that generates outcomes  $Y$  given  $X$  and  $A$ , illustrated in Figure 38.4  
44as the  $A \rightarrow Y \leftarrow X$ , is the same even after the  $X \rightarrow Y$  edge is removed. For example, we might  
45believe that the physiological processes by which smoking status  $A$  and confounders  $X$  produce  
46lung cancer  $Y$  remain the same, regardless of how the decision to smoke or not smoke was made.

47

Secondly, because the intervention does not change the composition of the population, we would also expect the distribution of background characteristics  $X$  to be the same between the observational and intervened processes.

With these insights about invariances between observed and interventional data, we can derive a statistical estimand for the ATE as follows.

**Theorem 2** (Adjustment with No Unobserved Confounders). *We observe  $A, Y, X \sim P$ . Suppose that*

1. *(Confounders observed) The data obeys the causal structure in Figure 38.4. In particular,  $X$  contains all common causes of  $A$  and  $Y$  and no variable in  $X$  is caused by  $A$  or  $Y$ .*
2. *(Overlap)  $0 < P(A = 1|X = x) < 1$  for all values of  $x$ . That is, there are no individuals for whom treatment is always or never assigned.*

Then, the average treatment effect is identified as  $\text{ATE} = \tau$ , where

$$\tau = \mathbb{E}[\mathbb{E}[Y|A = 1, X]] - \mathbb{E}[\mathbb{E}[Y|A = 0, X]]. \quad (38.20)$$

*Proof.* First, we expand the ATE using the tower property of expectation, conditioning on  $X$ . Then, we apply the invariances discussed above:

$$\text{ATE} = \mathbb{E}[Y|\text{do}(A = 1)] - \mathbb{E}[Y|\text{do}(A = 0)] \quad (38.21)$$

$$= \mathbb{E}[\mathbb{E}[Y|\text{do}(A = 1), X]] - \mathbb{E}[\mathbb{E}[Y|\text{do}(A = 0), X]] \quad (38.22)$$

$$= \mathbb{E}[\mathbb{E}[Y|A = 1, X]] - \mathbb{E}[\mathbb{E}[Y|A = 0, X]] \quad (38.23)$$

The final equality is the key to passing from a causal to observational quantity. This follows because, from the causal graph, the conditional distribution of  $Y$  given  $A, X$  is the same in both the original graph, and in the mutilated graph created by removing the edge from  $X$  to  $A$ . This mutilated graph defines  $P(Y|\text{do}(A = 1), X)$ , so the equality holds.

The condition that  $0 < P(A = 1|X = x) < 1$  is required for the first equality (the tower property) to be well defined.  $\square$

Note that Equation (38.20) is a function of only conditional expectations and distributions that appear in the observed data distribution (in particular, it contains no “do” operators). Thus, if we can fully characterize the observed data distribution  $P$ , we can map that distribution to a unique ATE.

It is useful to note how  $\tau$  differs from the naive estimand  $\mathbb{E}[Y|A = 1] - \mathbb{E}[Y|A = 0]$  that just reports the treatment-outcome association without adjusting for confounding. The comparison is especially clear when we write out the outer expectation in  $\tau$  explicitly as an integral over  $X$ :

$$\tau = \int \mathbb{E}[Y | A = 1, X]P(X)dX - \int \mathbb{E}[Y | A = 0, X]P(X)dX \quad (38.24)$$

We can write the naive estimand in a similar form by applying the tower property of expectation:

$$\mathbb{E}[Y | A = 1] - \mathbb{E}[Y | A = 0] = \int \mathbb{E}[Y | A = 1, X]P(X | A = 1)dX - \int \mathbb{E}[Y | A = 0, X]P(X | A = 0)dX$$

123

4 The key difference is the probability distribuiton over  $X$  that is being integrated over. The observational difference in means integrates over the over distinct conditional distributions of confounders 5  $X$ , depending on the value of  $A$ . On the other hand, in the ATE estimand  $\tau$ , we integrate over the 6 same distribution  $P(X)$  for both levels of the treatment.

78

9 **Overlap** In addition to the assumption on the causal structure, identification requires that there is 10 sufficient random variation in how treatments are assigned.

11

12 **Definition 1.** A distribution  $P$  on  $A, X$  satisfies **overlap** if  $0 < P(A = 1|x) < 1$  for all  $x$ . It 13 satisfies **strict overlap** if  $\epsilon < P(A = 1|x) < 1 - \epsilon$  for all  $x$  and some  $\epsilon > 0$ .

14

15 Overlap is the requirement that any unit could have either recieived the treatment or not.

16 To see the necessity of overlap, consider estimating the effectiveness of a drug in a study where 17 patient sex is a confounder, but the drug was only ever prescribed to male patients. Then, conditional 18 on a patient being female, we would know that patient was assigned to control. Without further 19 assumptions, it's impossible to know the effect of the drug on a population with female patients, 20 because there would be no data to inform the expected outcome for treated female patients, that is, 21  $E[Y | A = 1, X = \text{female}]$ . In this case, the statistical estimand (38.20) would not be identifiable. In 22 the same vein, strict overlap ensures that the conditional distributions at each stratum of  $X$  can be 23 estimated in finite samples.

24 Overlap can be particularly limiting in settings where we are adjusting for a large number of 25 covariates (in an effort to satisfy no unobserved confounding). Then, certain combinations of traits 26 may be very highly predictive of treatment assignment, even if individual traits are not. E.g., male 27 patients over age 70 with BMI greater than 25 are very rarely assigned the drug. If such groups 28 represent a significant fraction of the target population, or have significantly different treatment 29 effects, then this issue can be problematic. In this case, the strict overlap assumption puts very strong 30 restrictions on observational studies: for an observational study to satisfy overlap, most dimensions 31 of the confounders  $X$  would need to closely mimic the balance we would expect in an RCT [D'A+21].

32

### 33 38.4.2 ATE Estimation with Observed Confounders

34 We now return to estimating the ATE using observed—i.e., not experimental—data. We've shown 35 that in the case where we observe all common causes of the treatment and outcome, the ATE is 36 causally identified with a statistical estimand  $\tau$ . We now consider several strategies for estimating this 37 quantity using a finite data sample. Broadly, these techniques are known as backdoor adjustment.<sup>2</sup>

38 Recall that the defining characteristic of a confounding variable is that it affects both treatment 39 and outcome. Thus, an adjustment strategy may aim to account for the influence of confounders on 40 the observed outcome, the influence of confounders on treatment, or both. We discuss each of these 41 strategies in turn.

42

43 2. As we discuss in Section 38.8, this backdoor adjustment references the estimand returned by the do-calculus to 44 eliminate confounding from a backdoor path. This also generalizes the approaches discussed here to some cases where 45 we do not observe all common causes.

4647

---

### 38.4.2.1 Outcome Model Adjustment

We begin with an approach to covariate adjustment that relies on modeling the conditional expectation of the outcome  $Y$  given treatment  $A$  and confounders  $X$ . This strategy is often referred to as g-computation or outcome adjustment.<sup>3</sup> To begin, we define

**Definition 2.** *The conditional expected outcome is the function  $Q$  given by*

$$Q(a, x) = \mathbb{E}[Y|A = a, X = x]. \quad (38.26)$$

Substituting this definition into the definition of our estimand  $\tau$ , Equation (38.20), we have  $\tau = \mathbb{E}[Q(1, x) - Q(0, x)]$ . This suggests a procedure for estimating  $\tau$ : fit a model  $\hat{Q}$  for  $Q$  and then report

$$\hat{\tau}^Q \triangleq \frac{1}{n} \sum_i \hat{Q}(1, x_i) - \hat{Q}(0, x_i). \quad (38.27)$$

To fit  $\hat{Q}$ , recall that  $E[Y|a, x] = \operatorname{argmin}_Q \mathbb{E}[(Y - Q(A, X))^2]$ . That is, the minimizer (among all functions) of the squared loss risk is the conditional expected outcome.<sup>4</sup> So, to approximate  $Q$ , we simply use mean squared error to fit a predictor that predicts  $Y$  from  $A$  and  $X$ .

The estimation procedure takes several steps. We first fit a model  $\hat{Q}$  to predict  $Y$ . Then, for each unit  $i$ , we predict that unit's outcome had they received treatment  $\hat{Q}(1, x_i)$  and we predict their outcome had they not received treatment  $\hat{Q}(0, x_i)$ .<sup>5</sup> If the unit actually did receive treatment ( $a_i = 1$ ) then  $\hat{Q}(0, x_i)$  is our guess about what would have happened in the counterfactual case that they did not. The estimated expected gain from treatment for this individual is  $\hat{Q}(1, x_i) - \hat{Q}(0, x_i)$ —the difference in expected outcome between being treated and not treated. Finally, we estimate the outer expectation with respect to  $P(X)$ —the true population distribution of the confounders—using the empirical distribution  $\hat{P}(X) = 1/n \sum_i \delta_{x_i}$ . In effect, this means we substitute the expectation (over an unknown distribution) by an average over the observed data.

**Linear regression** It's worth saying something more about the special case where  $Q$  is modeled as a linear function of both the treatment and all the covariates. That is, the case where we assume the identification conditions of Theorem 2 and we additionally assume that the true, causal law (the SCM) governing  $Y$  yields:  $Q(A, X) = \mathbb{E}[Y|A, X] = \mathbb{E}[f_Y(A, X, \xi)|A, X] = \beta_0 + \beta_A A + \beta_X X$ . Plugging in, we see that  $Q(1, X) - Q(0, X) = \beta_A$  (and so also  $\tau = \beta_A$ ). Then, the estimator for the average treatment effect reduces to the estimator for the regression coefficient  $\beta_A$ . This “fit linear regression and report the regression coefficient” remains a common way of estimating the association between two variables in practice. The expected-outcome-adjustment procedure here may be viewed as a generalization of this procedure that removes the linear parametric assumption.

### 38.4.2.2 Propensity Score Adjustment

Outcome model adjustment relies on modeling the relationship between the confounders and the outcome. A popular alternative is to model the relationship between the confounders and the

3. The “g” stands for generalized, for now-inscrutable historical reasons [Rob86].

4. To be precise, this definition applies when  $X$  and  $Y$  are square-integrable, and the minimization taken over measurable functions.

5. this interpretation is justified by the same conditions as Theorem 2

<sup>1</sup> treatment. This strategy adjusts for confounding by directly addressing sampling bias in the treated  
<sup>2</sup> and control groups. This bias arises from the relationship between the confounders and the treatment.  
<sup>3</sup> Intuitively, the effect of confounding may be viewed as due to the difference between  $P(X|A = 1)$   
<sup>4</sup> and  $P(X|A = 0)$ —e.g., the population of people who rode horses as children is wealthier than the  
<sup>5</sup> population of people who did not. When we observe all confounding variables  $X$ , this degree of over-  
<sup>6</sup> or under-representation can be adjusted away by reweighting samples such that the confounders  $X$   
<sup>7</sup> have the same distribution in the treated and control groups. When the confounders are balanced  
<sup>8</sup> between the two groups, then any differences between them must be attributable to the treatment.  
<sup>9</sup> A key quantity for balancing treatment and control groups is the **propensity score**, which  
<sup>10</sup> summarises the relationship between confounders and treatment.

<sup>12</sup> **Definition 3.** *The propensity score is the function  $g$  given by  $g(x) = P(A = 1|X = x)$ .*

<sup>13</sup> To make use of the propensity score in adjustment, we first rewrite the estimand  $\tau$  in a suggestive  
<sup>14</sup> form:

$$\tau = \mathbb{E}\left[\frac{YA}{g(X)} - \frac{Y(1-A)}{1-g(X)}\right]. \quad (38.28)$$

<sup>18</sup> This identity can be verified by noting that  $\mathbb{E}[YA|X] = \mathbb{E}[Y|A = 1, X]P(A = 1|X) + 0$ , rearranging  
<sup>19</sup> for  $\mathbb{E}[Y|A = 1, X]$ , doing the same for  $\mathbb{E}[Y|A = 0, X]$ , and substituting in to Equation (38.20). Note  
<sup>20</sup> that the identity is just a mathematical fact about the statistical estimand—it does not rely on any  
<sup>21</sup> causal assumptions, and holds whether or not  $\tau$  can be interpreted as a causal effect.

<sup>22</sup> This expression suggests the **inverse probability of treatment weighted estimator**, or IPTW  
<sup>23</sup> estimator:

$$\hat{\tau}^{\text{IPTW}} \triangleq \frac{1}{n} \sum_i \frac{Y_i A_i}{\hat{g}(X_i)} - \frac{Y_i(1-A_i)}{1-\hat{g}(X_i)}. \quad (38.29)$$

<sup>27</sup> Here,  $\hat{g}$  is a estimate of the propensity score function. Recall from Section 14.2.1 that if a model is well-  
<sup>28</sup> specified and the loss function is a proper scoring rule then risk minimizer  $g^* = \operatorname{argmin}_g \mathbb{E}[L(A, g(X))]$   
<sup>29</sup> will be  $g^*(X) = P(A = 1|X)$ . That is, we can estimate the propensity score by fitting a model that  
<sup>30</sup> predicts  $A$  from  $X$ . Cross-entropy and squared loss are both proper scoring rules, so we may use  
<sup>31</sup> standard pipelines.

<sup>32</sup> In summary, the procedure is to estimate the propensity score function (with machine learning),  
<sup>33</sup> and then to plug the estimated propensity scores  $\hat{g}(x_i)$  into Equation (38.29). The IPTW estimator  
<sup>34</sup> computes a difference of weighted averages between the treated and untreated group. The effect is to  
<sup>35</sup> upweight the outcomes of units who were unlikely to be treated but who nevertheless actually, by  
<sup>36</sup> chance, received treatment (and similarly for untreated). Intuitively, such units are typical for the  
<sup>37</sup> untreated population. So, their outcomes under treatment are informative about what would have  
<sup>38</sup> happened had a typical untreated unit received treatment.

<sup>39</sup> A word of warning is in order. Although the IPTW is asymptotically valid and popular in practice,  
<sup>40</sup> it can be very unstable in finite samples. If estimated propensity scores are extreme for some values  
<sup>41</sup> of  $x$  (that is, very close to 0 or 1), then the corresponding IPTW weights can be very large, resulting  
<sup>42</sup> in a high-variance estimator. In some cases, this instability can be mitigated by instead using the  
<sup>43</sup> Hajek version of the estimator.

$$\hat{\tau}^{\text{h-IPTW}} \triangleq \sum_i Y_i A_i \frac{1/\hat{g}(X_i)}{\sum_i A_i/\hat{g}(X_i)} - \sum_i Y_i(1-A_i) \frac{1/(1-\hat{g}(X_i))}{\sum_i (1-A_i)/(1-\hat{g}(X_i))}. \quad (38.30)$$

<sup>44</sup>

However, extreme weights can persist even after self-normalization, either because there are truly strata of  $X$  where treatment assignment is highly imbalanced, or because the propensity score estimation method has overfit. In such cases, it is common to apply heuristics such as weight clipping.

See Khan and Ugander [KU21] for a longer discussion of inverse-propensity type estimators, including some practical improvements.

### 38.4.2.3 Double Machine Learning

We have seen how to estimate the average treatment effect using either the relationship between confounders and outcome, or the relationship between confounders and treatment. In each case, we follow a two step estimation procedure. First, we fit models for the expected outcome or the propensity score. Second, we plug these fitted models into a downstream estimator of the effect.

Unsurprisingly, the quality of the estimate of  $\tau$  depends on the quality of the estimates  $\hat{Q}$  or  $\hat{g}$ . This is problematic because  $Q$  and  $g$  may be complex functions that require large numbers of samples to estimate. Even though we're only interested in the 1-dimensional parameter  $\tau$ , the naive estimators described thus far can have very slow rates of convergence. This leads to unreliable inference or very large confidence intervals.

Remarkably, there are strategies for combining  $Q$  and  $g$  in estimators that, in principle, do better than using either  $Q$  or  $g$  alone. The **Augmented Inverse Probability of Treatment Weighted Estimator (AIPTW)** is one such estimator. It is defined as

$$\hat{\tau}^{\text{AIPTW}} \triangleq \frac{1}{n} \sum_i \hat{Q}(1, X_i) - \hat{Q}(0, X_i) + A_i \frac{Y_i - \hat{Q}(1, X_i)}{\hat{g}(x_i)} - (1 - A_i) \frac{Y_i - \hat{Q}(0, X_i)}{1 - \hat{g}(X_i)}. \quad (38.31)$$

That is,  $\hat{\tau}^{\text{AIPTW}}$  is the outcome adjustment estimator plus a stabilization term that depends on the propensity score. This estimator is a particular case of a broader class of estimators that are referred to as **semi-parametrically efficient** or **double machine-learning** estimators [Che+17e; Che+17d]. We'll use the later terminology here.

We now turn to understanding the sense in which double machine learning estimators are robust to misestimation of the **nuisance functions**  $Q$  and  $g$ . To this end, we define the **influence curve** of  $\tau$  to be the function  $\phi$  defined by<sup>6</sup>

$$\phi(X_i, A_i, Y_i; Q, g, \tau) \triangleq Q(1, X_i) - Q(0, X_i) + A_i \frac{Y_i - Q(1, X_i)}{g(x_i)} - (1 - A_i) \frac{Y_i - Q(0, X_i)}{1 - g(X_i)} - \tau. \quad (38.32)$$

By design,  $\hat{\tau}^{\text{AIPTW}} - \tau = \frac{1}{n} \sum_i \phi(\mathbf{X}_i; \hat{Q}, \hat{g}, \tau)$ . We begin by considering what would happen if we simply knew  $Q$  and  $g$ , and didn't have to estimate them. In this case, the estimator would be  $\tau^{\text{ideal}} = \frac{1}{n} \sum_i \phi(\mathbf{X}_i; Q, g, \tau)$  and, by the central limit theorem, we would have:

$$\sqrt{n}(\hat{\tau}^{\text{ideal}} - \tau) \xrightarrow{d} \text{Normal}(0, \mathbb{E}[\phi(\mathbf{X}_i; Q, g, \tau)^2]). \quad (38.33)$$

<sup>6</sup> Influence curves are the foundation of what follows, and the key to generalizing the analysis beyond the ATE. Unfortunately, going into the general mathematics would require a major digression, so we omit it. However, see references at the end of the chapter for some pointers to the relevant literature.

1 This result characterizes the estimation uncertainty in the best possible case. If we knew  $Q$  and  $g$ ,  
2 we could rely on this result for, e.g., finding confidence intervals for our estimate.

4 The question is: what happens when  $Q$  and  $g$  need to be estimated? For general estimators and  
5 nuisance function models, we don't expect the  $\sqrt{n}$ -rate of Equation (38.33) to hold. For instance,  
6  $\sqrt{n}(\hat{\tau}^Q - \tau)$  only converges if  $\sqrt{n}\mathbb{E}[(\hat{Q} - Q)^2]^{\frac{1}{2}} \rightarrow 0$ . That is, for the naive estimator we only get the  
7  $\sqrt{n}$  rate for estimating  $\tau$  if we can also estimate  $Q$  at the  $\sqrt{n}$  rate—a much harder task! This is the  
8 issue that the double machine learning estimator helps with.

9 To understand how, we decompose the error in estimating  $\tau$  as follows:

$$\begin{aligned} \underline{10} \quad & \sqrt{n}(\hat{\tau}^{\text{AIPTW}} - \tau) \\ \underline{11} \end{aligned} \tag{38.34}$$

$$\begin{aligned} \underline{12} \quad & = \frac{1}{\sqrt{n}} \sum_i \phi(\mathbf{X}_i; Q, g, \tau) \\ \underline{13} \end{aligned} \tag{38.35}$$

$$\begin{aligned} \underline{14} \quad & + \frac{1}{\sqrt{n}} \sum_i \phi(\mathbf{X}_i; \hat{Q}, \hat{g}, \tau) - \phi(\mathbf{X}_i; Q, g, \tau) - \mathbb{E}[\phi(\mathbf{X}; \hat{Q}, \hat{g}, \tau) - \phi(\mathbf{X}; Q, g, \tau)] \\ \underline{15} \end{aligned} \tag{38.36}$$

$$\begin{aligned} \underline{16} \quad & + \sqrt{n}\mathbb{E}[\phi(\mathbf{X}; \hat{Q}, \hat{g}, \tau) - \phi(\mathbf{X}; Q, g, \tau)] \\ \underline{17} \end{aligned} \tag{38.37}$$

18 We recognize the first term, Equation (38.35), as  $\sqrt{n}(\hat{\tau}^{\text{ideal}} - \tau)$ , the estimation error in the optimal  
19 case where we know  $Q$  and  $g$ . Ideally, we'd like the error of  $\hat{\tau}^{\text{AIPTW}}$  to be asymptotically equal to  
20 this ideal case—which will happen if the other two terms go to 0.

22 The second term, Equation (38.36), is a penalty we pay for using the same data to estimate  $Q$ ,  $g$   
23 and to compute  $\tau$ . For many model classes, it can be shown that such “empirical process” terms go  
24 to 0. This can also be guaranteed in general by using different data for fitting the nuisance functions  
25 and for computing the estimator (see the next section).

26 The third term, Equation (38.37), captures the penalty we pay for misestimating the nuisance  
27 functions. This is where the particular form of the AIPTW is key. With a little algebra, we can show  
28 that

$$\begin{aligned} \underline{29} \quad \mathbb{E}[\phi(\mathbf{X}; \hat{Q}, \hat{g}) - \phi(\mathbf{X}; Q, g)] &= \mathbb{E}\left[\frac{1}{g(X)}(\hat{g}(X) - g(X))(\hat{Q}(1, X) - Q(1, X))\right] \\ \underline{30} \end{aligned} \tag{38.38}$$

$$\begin{aligned} \underline{31} \quad & + \frac{1}{1 - g(X)}(\hat{g}(X) - g(X))(\hat{Q}(0, X) - Q(0, X)). \\ \underline{32} \end{aligned} \tag{38.39}$$

33 The important point is that estimation errors of  $Q$  and  $g$  are multiplied together. Using the Cauchy-  
34 Schwarz inequality, we find that  $\sqrt{n}\mathbb{E}[\phi(\mathbf{X}; \hat{Q}, \hat{g}) - \phi(\mathbf{X}; Q, g)] \rightarrow 0$  as long as  $\sqrt{n} \max_a \mathbb{E}[(\hat{Q}(a, X) -$   
35  $Q(a, X))^2]^{\frac{1}{2}} \mathbb{E}[(\hat{g}(X) - g(X))^2]^{\frac{1}{2}} \rightarrow 0$ . That is, the misestimation penalty will vanish so long as the  
36 product of the misestimation errors is  $o(\sqrt{n})$ . For example, this means that that  $\tau$  can be estimated at  
37 the (optimal)  $\sqrt{n}$  rate even when the estimation error of each of  $Q$  and  $g$  only decreases as  $o(n^{-1/4})$ .

39 The upshot here is that the double machine learning estimator has the special property that the  
40 weak condition  $\sqrt{n}\mathbb{E}[(\hat{Q}(T, X) - Q(T, X))^2 \mathbb{E}(\hat{g}(X) - g(X))^2] \rightarrow 0$  suffices to imply that

$$\begin{aligned} \underline{41} \quad \sqrt{n}(\hat{\tau}^{\text{AIPTW}} - \tau) &\xrightarrow{d} \text{Normal}(0, \mathbb{E}[\phi(\mathbf{X}_i; Q, g, \tau)^2]) \\ \underline{42} \end{aligned} \tag{38.40}$$

44(though strictly speaking this requires some additional technical conditions we haven't discussed).  
45This is *not* true for the earlier estimators we discussed, which require a much faster rate of convergence  
46for the nuisance function estimation.

47

The AIPTW estimator has two further nice properties that are worth mentioning. First, it is **non-parametrically efficient**. This means that this estimator has the smallest possible variance of any estimator that does not make parametric assumptions; namely,  $\mathbb{E}[\phi(\mathbf{X}_i; Q, g, \tau)^2]$ . This means, for example, that this estimator yields the smallest confidence intervals of any approach that does not rely on parametric assumptions. Second, it is **double robust**: the estimator is consistent (converges to the true  $\tau$  as  $n \rightarrow \infty$ ) as long as at least one of either  $\hat{Q}$  or  $\hat{g}$  is consistent.

#### 38.4.2.4 Cross Fitting

The term Equation (38.36) in the error decomposition above is the penalty we pay for reusing the same data to both fit  $Q, g$  and to compute the estimator. For many choices of model for  $Q, g$ , this term goes to 0 quickly as  $n$  gets large and we achieve the (best case)  $\sqrt{n}$  error rate. However, this property doesn't always hold.

As an alternative, we can always randomly split the available data and use one part for model fitting, and the other to compute the estimator. Effectively, this means the nuisance function estimation and estimator computation are done using independent samples. It can then be shown that the reuse penalty will vanish. However, this comes at the price of reducing the amount of data available for each of nuisance function estimation and estimator computation.

This strategy can be improved upon by a **cross fitting** approach. We divide the data into  $K$  folds. For each fold  $j$  we use the other  $K - 1$  folds to fit the nuisance function models  $\hat{Q}^{-j}, \hat{g}^{-j}$ . Then, for each datapoint  $i$  in fold  $j$ , we take  $\hat{Q}(a_i, x_i) = \hat{Q}^{-j}(a_i, x_i)$  and  $\hat{g}(x_i) = \hat{g}^{-j}(x_i)$ . That is, the estimated conditional outcomes and propensity score for each datapoint are predictions from a model that was not trained on that datapoint. Then, we estimate  $\tau$  by plugging  $\{\hat{Q}(a_i, x_i), \hat{g}(x_i)\}_i$  into Equation (38.31). It can be shown that this cross fitting procedure has the same asymptotic guarantee—the central limit theorem at the  $\sqrt{n}$  rate—as described above.

#### 38.4.3 Uncertainty Quantification

In addition to the point estimate  $\hat{\tau}$  of the average treatment effect, we'd also like to report a measure of the uncertainty in our estimate. For example, in the form of a confidence interval. The asymptotic normality of  $\sqrt{n}\hat{\tau}$  (Equation (38.40)) provides a means for this quantification. Namely, we could base confidence intervals and similar on the limiting variance  $\mathbb{E}[\phi(\mathbf{X}_i; Q, g, \tau)^2]$ . Of course, we don't actually know any of  $Q, g$ , or  $\tau$ . However, it turns out that it suffices to estimate the asymptotic variance with  $\frac{1}{n} \sum_i \phi(\mathbf{X}_i; \hat{Q}, \hat{g}, \hat{\tau})^2$  [Che+17e]. That is, we can estimate the uncertainty by simply plugging in our fitted nuisance models and our point estimate of  $\tau$  into

$$\hat{\mathbb{V}}[\hat{\tau}] = 1/n \sum_i \phi(\mathbf{X}_i; \hat{Q}, \hat{g}, \hat{\tau})^2. \quad (38.41)$$

This estimated variance can then be used to compute confidence intervals in the usual manner. E.g., we'd report a 95% confidence interval for  $\tau$  as  $\hat{\tau} \pm 1.96\sqrt{\hat{\mathbb{V}}[\hat{\tau}]/n}$ .

Alternatively, we could quantify the uncertainty by bootstrapping. Note, however, that this would require refitting the nuisance functions with each bootstrap model. Depending on the model and data, this can be prohibitively computationally expensive.

### <sup>1</sup> 38.4.4 Matching

<sup>3</sup> One particularly popular approach to adjustment-based causal estimation is **matching**. Intuitively,  
<sup>4</sup> the idea is to match each treated to unit to an untreated unit that has the same (or at least similar)  
<sup>5</sup> values of the confounding variables and then compare the observed outcomes of the treated unit and  
<sup>6</sup> its matched control. If we match on the full set of common causes, then the difference in outcomes is,  
<sup>7</sup> intuitively, a noisy estimate of the effect the treatment had on that treated unit. We'll now build  
<sup>8</sup> this up a bit more carefully. In the process we'll see that matching can be understood as, essentially,  
<sup>9</sup> a particular kind of outcome model adjustment.

<sup>10</sup> For simplicity, consider the case where  $X$  is a discrete random variable. Define  $\mathcal{A}_x$  to be the set of  
<sup>11</sup>treated units with covariate value  $x$ , and  $\mathcal{C}_x$  to be the set of untreated units with covariate value  $x$ .  
<sup>12</sup>In this case, the matching estimator is:

<sup>13</sup>

$$\hat{\tau}^{\text{matching}} = \sum_x \hat{P}(x) \left( \frac{1}{|\mathcal{A}_x|} \sum_{i \in \mathcal{A}_x} Y_i - \frac{1}{|\mathcal{C}_x|} \sum_{j \in \mathcal{C}_x} Y_j \right), \quad (38.42)$$

<sup>17</sup> where  $\hat{P}(x)$  is an estimator of  $P(X = x)$ —e.g., the fraction of units with  $X = x$ . Now, we can rewrite  
<sup>18</sup>  $Y_i = Q(A_i, X_i) + \xi_i$  where  $\xi_i$  is a unit-specific noise term defined by the equation. In particular, we  
<sup>19</sup> have that  $E[\xi_i | A_i, X_i] = 0$ . Subbing this in, we have:  
<sup>20</sup>

$$\hat{\tau}^{\text{matching}} = \sum_x \hat{P}(x) (Q(1, x) - Q(0, x)) + \sum_x \frac{1}{|\mathcal{A}_x|} \sum_{i \in \mathcal{A}_x} \xi_i - \frac{1}{|\mathcal{C}_x|} \sum_{j \in \mathcal{C}_x} \xi_j. \quad (38.43)$$

<sup>21</sup>We can recognize the first term as an estimator of usual target parameter  $\tau$  (it will be equal to  $\tau$  if  
<sup>22</sup> $\hat{P}(x) = P(x)$ ). The second term is a difference of averages of random variables with expectation 0,  
<sup>23</sup> and so each term will converge to 0 as long as  $|\mathcal{A}_x|$  and  $|\mathcal{C}_x|$  each go to infinity as we see more and  
<sup>24</sup> more data. Thus, we see that the matching estimator is a particular way of estimating the parameter  
<sup>25</sup> $\tau$ . The procedure can be extended to continuous covariates by introducing some notion of values of  
<sup>26</sup> $X$  being close, and then matching close treatment and control variables.  
<sup>27</sup>

<sup>28</sup> There are two points we should emphasize here. First, notice that the argument here has nothing  
<sup>29</sup> to do with causal identification. Matching is a particular technique for estimating the observational  
<sup>30</sup> parameter  $\tau$ . Whether or not  $\tau$  can be interpreted as an average treatment effect is determined by  
<sup>31</sup> the conditions of Theorem 2—the particular estimation strategy doesn't say anything about this.  
<sup>32</sup> Second, notice that in essence matching amounts to a particular choice of model for  $\hat{Q}$ . Namely,  
<sup>33</sup>  $\hat{Q}(1, x) = \frac{1}{|\mathcal{A}_x|} \sum_{i \in \mathcal{A}_x} Y_i$  and similarly for  $\hat{Q}(0, x)$ . That is, we estimate the conditional expected  
<sup>34</sup> outcome as a sample mean over units with the same covariate value. Whether this is a good idea  
<sup>35</sup> depends on how good of a model for  $Q$  this is. In situations where better models are possible (e.g.,  
<sup>36</sup>a machine-learning model fits the data well), we might expect to get a more accurate estimate by  
<sup>37</sup> using the conditional expected outcome predictor directly.  
<sup>38</sup>

<sup>39</sup> There is another important case we mention in passing. In general, when using adjustment based  
<sup>40</sup> identification, it suffices to adjust for any function  $\phi(X)$  of  $X$  such that  $A \perp\!\!\!\perp X | \phi(X)$ . To see that  
<sup>41</sup> adjusting for only  $\phi(X)$  suffices, first notice that  $g(X) = P(A = 1 | X) = P(A = 1 | \phi(X))$  only depends  
<sup>42</sup> on  $\phi(X)$ , and then recall that can write the target parameter as  $\tau = E[\frac{YA}{g(X)} - \frac{Y(1-A)}{1-g(X)}]$ , whence  
<sup>43</sup> $\tau$  only depends on  $X$  through  $g(X)$ . That is: replacing  $X$  by a reduced version  $\phi(X)$  such that  
<sup>44</sup> $g(X) = P(A = 1 | \phi(X))$  can't make any difference to  $\tau$ . Indeed, the most popular choice of  $\phi(X)$  is  
<sup>45</sup> $g(X) = P(A = 1 | \phi(X))$   
<sup>46</sup>

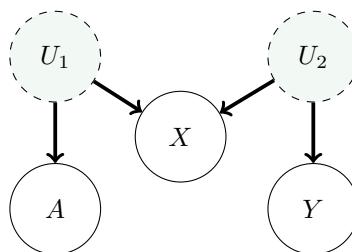


Figure 38.5: The *M*-bias causal graph. Here,  $A$  and  $Y$  are not confounded. However, conditioning on the covariate  $X$  opens a backdoor path, passing through  $U_1$  and  $U_2$  (because  $X$  is a collider). Thus, adjusting for  $X$  creates bias. This is true even though  $X$  need not be a pre-treatment variable.

the propensity score itself,  $\phi(X) = g(X)$ . This leads to **propensity score matching**, a two step procedure where we first fit a model for the propensity score, and then run matching based on the estimated propensity score values for each unit. Again, this is just a particular estimation procedure for the observational parameter  $\tau$ , and says nothing about whether it's valid to interpret  $\tau$  as a causal effect.

### 38.4.5 Practical Considerations and Procedures

Lots of issues can arise in practice.

#### 38.4.5.1 What to adjust for

Choosing which variables to adjust for is a key detail in estimating causal effects using covariate adjustment. The criterion is clear when one has a full causal graph relating  $A$ ,  $Y$ , and all covariates  $X$  to each other. Namely, adjust for all variables that are actually causal parents of  $A$  and  $Y$ . In fact, with access to the full graph, this criteria can be generalized somewhat—see Section 38.8.

In practice, we often don't actually know the full causal graph relating all of our variables. As a result, it is common to apply simple heuristics to determine which variables to adjust for. Unfortunately, these heuristics have serious limitations. However, exploring these are instructive.

A key condition in Theorem 2 is that the covariates  $X$  that we adjust for must include all the common causes. In the absence of a full causal graph, it is tempting to condition on as many observed variables as possible to try to ensure this condition holds. However, this can be problematic. For instance, suppose that  $M$  is a mediator of the effect of  $A$  on  $Y$ —i.e.,  $M$  lies on one of the directed paths between  $A$  and  $Y$ . Then, conditioning on  $M$  will block this path, removing some of the causal effect. Note that this does not always result in an attenuated, or smaller-magnitude, effect estimate. The effect through a given mediator may run in the opposite direction of other causal pathways from the treatment; thus conditioning on a mediator can inflate or even flip the sign of a treatment effect. Alternatively, if  $C$  is a collider between  $A$  and  $Y$ —a variable that is caused by both—then conditioning on  $C$  will induce an extra statistical dependency between  $A$  and  $Y$ .

Both pitfalls of the “condition on everything” heuristic discussed above both involve conditioning

<sup>1</sup> on variables that are downstream of the treatment  $A$ . A natural response is to this is to limit  
<sup>2</sup> conditioning to all pre-treatment variables, or those that are causally upstream of the treatment.  
<sup>3</sup> Importantly, if there is a valid adjustment set in the observed covariates  $X$ , then there will also be a  
<sup>4</sup> valid adjustment set among the pre-treatment covariates. This is because any open backdoor path  
<sup>5</sup> between  $A$  and  $Y$  must include a parent of  $A$ , and the set of pre-treatment covariates includes these  
<sup>6</sup> parents. However, it is still possible that conditioning on the full set of pre-treatment variables can  
<sup>7</sup> induce new backdoor paths between  $A$  and  $Y$  through colliders. In particular, if there is a covariate  
<sup>8</sup>  $D$  that is separately confounded with the treatment  $A$  and the outcome  $Y$  then  $D$  is a collider, and  
<sup>9</sup> conditioning on  $D$  opens a new backdoor path. This phenomenon is known as m-bias because of the  
<sup>10</sup> shape of the graph [Pea09c], see Figure 38.5.

<sup>11</sup> A practical refinement of the pre-treatment variable heuristic is given in VanderWeele TJ [VT11].  
<sup>12</sup> Their heuristic suggests conditioning on all pre-treatment variables that are causes of the treatment,  
<sup>13</sup> outcome, or both. The essential qualifier in this heuristic is that the variable is causally upstream of  
<sup>14</sup> treatment and/or outcome. This eliminates the possibility of conditioning on covariates that are  
<sup>15</sup> only confounded with treatment and outcome, avoiding m-bias. Notably, this heuristic requires more  
<sup>16</sup> causal knowledge than the above heuristics, but does not require detailed knowledge of how different  
<sup>17</sup> covariates are causally related to each other.

<sup>18</sup> The VanderWeele TJ [VT11] criterion is a useful rule of thumb, but other practical considerations  
<sup>19</sup> often arise. For example, if one has more knowledge about the causal structure among covariates, it  
<sup>20</sup> is possible to optimize adjustment sets to minimize the variance of the resulting estimator [RS20].  
<sup>21</sup> One important example of reducing variance by pruning adjustment sets is the exclusion of variables  
<sup>22</sup> that are known to only be a parent of the treatment, and not of the outcome (so called instruments,  
<sup>23</sup> as discussed in Section 38.5).

<sup>24</sup> Finally, adjustment set selection criteria operate under the assumption that there actually exists a  
<sup>25</sup> valid adjustment set among observed covariates. When there is no set of observed covariates in  $X$   
<sup>26</sup> that block all backdoor paths, then any adjusted estimate will be biased. Importantly, in this case,  
<sup>27</sup> the bias does not necessarily decrease as one conditions on more variables. For example, conditioning  
<sup>28</sup> on an instrumental variable often results in an estimate that has higher bias, in addition to the  
<sup>29</sup> higher variance discussed above. This phenomenon is known as bias amplification or z-bias; see  
<sup>30</sup> Section 38.7.2. A general rule of thumb is that variables that explain away much more variation in  
<sup>31</sup> the treatment than in the outcome can potentially amplify bias, and should be treated with caution.

<sup>32</sup>

### <sup>33</sup> 38.4.5.2 Overlap

<sup>34</sup>

<sup>35</sup> Recall that in addition to no-unobserved-confounders, identification of the average treatment effect  
<sup>36</sup> requires overlap: the condition that  $0 < P(A = 1|x) < 1$  for the population distribution  $P$ . With  
<sup>37</sup> infinite data, any amount of overlap will suffice for estimating the causal effect. In realistic settings,  
<sup>38</sup> even near failures can be problematic. Equation (38.40) gives an expression for the (asymptotic)  
<sup>39</sup> variance of our estimate:  $\mathbb{E}[\phi(\mathbf{X}_i; \hat{Q}, \hat{g}, \hat{\tau})^2]/n$ . Notice that  $\phi(\mathbf{X}_i; \hat{Q}, \hat{g}, \hat{\tau})^2$  involves terms that are  
<sup>40</sup> proportional to  $1/g(x)$  and  $1/(1 - g(x))$ . Accordingly, the variance of our estimator will balloon  
<sup>41</sup> if there are units where  $g(x) \approx 0$  or  $g(x) \approx 1$  (unless such units are rare enough that they don't  
<sup>42</sup> contribute much to the expectation).

<sup>43</sup> In practice, a simple way to deal with potential overlap violation is to fit a model  $\hat{g}$  for the  
<sup>44</sup> treatment assignment probability—which we need to do anyways—and check that the values  $\hat{g}(x)$   
<sup>45</sup> are not too extreme. In the case that some values are too extreme, the simplest resolution is to cheat.

<sup>46</sup>

We can simply exclude all the data with extreme values of  $\hat{g}(x)$ . This is equivalent to considering the average treatment effect over only the subpopulation where overlap is satisfied. This changes the interpretation of the estimand. The restricted subpopulation ATE may or may not provide a satisfactory answer to the real-world problem at hand, and this needs to be justified based on knowledge of the real-world problem.

### 38.4.5.3 Choice of Estimand and Average Treatment Effect on the Treated

Usually, our goal in estimating a causal effect is qualitative. We want to know what the sign of the effect is, and whether it's large or small. The utility of the ATE is that it provides a concrete query we can use to get a handle on the qualitative question. However, it is not sacrosanct; sometimes we're better off choosing an alternative causal estimand that still answers the qualitative question but which is easier to estimate statistically. The **average treatment effect on the treated** or ATT,

$$\text{ATT} \triangleq \mathbb{E}_{X|A=1}[\mathbb{E}[Y|X, \text{do}(A=1)] - \mathbb{E}[Y|X, \text{do}(A=0)]], \quad (38.44)$$

is one such an estimand that is frequently useful.

The ATT is useful when many members of the population are very unlikely to receive treatment, but the treated units had a reasonably high probability of receiving the control. This can happen if, e.g., we sample control units from the general population, but the treatment units all self-selected into treatment from a smaller subpopulation. In this case, it's not possible to (non-parametrically) determine the treatment effect for the control units where no similar unit took treatment. The ATT solves this obstacle by simply omitting such units from the average.

If we have the causal structure Figure 38.4, and the overlap condition  $P(A=1|X=x) < 1$  for all  $X = x$  then the ATT is causally identified as

$$\tau^{\text{ATT}} = \mathbb{E}_{X|A=1}[\mathbb{E}[Y|A=1, X] - \mathbb{E}[Y|A=0, X]]. \quad (38.45)$$

Note that the required overlap condition here is weaker than for identifying the ATE. (The proof is the same as Theorem 2.)

The estimation strategies for the ATE translate readily to estimation strategies for the ATT. Namely, estimate the nuisance functions the same way and then simply replace averages over all data points by averages over the treated datapoints only. In principle, it's possible to do a little better than this by making use of the untreated datapoints as well. A corresponding double machine learning estimator is

$$\hat{\tau}^{\text{ATT-AIPTW}} \triangleq \frac{1}{n} \sum_i \frac{A_i}{P(A=1)} (Y - \hat{Q}(0, X_i)) - \frac{(1 - A_i)g(X)}{P(A=1)(1 - g(X))} (Y - \hat{Q}(0, X_i)). \quad (38.46)$$

. The variance of this estimator can be estimated by

$$\phi^{\text{ATT}}(\mathbf{X}_i; Q, g, \tau) \triangleq \frac{1}{n} \sum_i \frac{A_i}{P(A=1)} (Y - \hat{Q}(0, X_i)) - \frac{(1 - A_i)g(X)}{P(A=1)(1 - g(X))} (Y - \hat{Q}(0, X_i)) - \frac{A\tau}{P(A=1)} \quad (38.47)$$

$$\hat{\mathbb{V}}[\hat{\tau}^{\text{ATT-AIPTW}}] \triangleq \frac{1}{n} \sum_i \phi^{\text{ATT}}(\mathbf{X}_i; \hat{Q}, \hat{g}, \hat{\tau}^{\text{ATT-AIPTW}}). \quad (38.48)$$

<sup>1</sup> Notice that the estimator for the ATT doesn't require estimating  $Q(1, X)$ . This can be a considerable  
<sup>2</sup> advantage when the treated units are rare.  
<sup>3</sup> See Chernozhukov et al. [Che+17e] for details.

<sup>5</sup>

#### <sup>6</sup> 38.4.6 Summary and Practical Advice

<sup>7</sup> We have seen a number of estimators that follow the general procedure:

- <sup>9</sup> 1. fit statistical or machine-learning models  $\hat{Q}(a, x)$  as a predictor for  $Y$ , and/or  $\hat{g}(x)$  as a predictor  
<sup>10</sup> for  $A$
- <sup>11</sup> 2. compute the predictions  $\hat{Q}(0, x_i), \hat{Q}(1, x_i), \hat{g}(x_i)$  for each data point, and
- <sup>13</sup> 3. combine these predictions into an estimate of the average treatment effect.

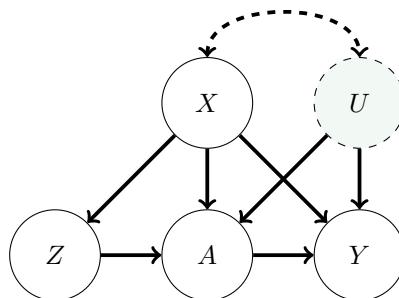
<sup>14</sup> Importantly, no single estimation approach is a silver bullet. For example, the double machine-  
<sup>16</sup>learning estimator has appealing theoretical properties, such as asymptotic efficiency guarantees and  
<sup>17</sup>a recipe for estimating uncertainty without needing to bootstrap the model fitting. However, in  
<sup>18</sup>terms of the quality of point estimates, the double ML estimators can sometimes underperform their  
<sup>19</sup>more naive counterparts [KS07]. In fact, there are cases where each of outcome regression, propensity  
<sup>20</sup>weighting, or doubly robust methods will outperform the others.

<sup>21</sup> One difficulty in choosing an estimator in practice is that there are fewer guardrails in causal  
<sup>22</sup>inference than there are in standard predictive modeling. In predictive modeling, we construct a  
<sup>23</sup>train-test split and validate our prediction models using the true labels or outcomes in the held-out  
<sup>24</sup>dataset. However, for causal problems, the causal estimands are functionals of a different data-  
<sup>25</sup>generating process from the one that we actually observed. As a result, it is impossible to empirically  
<sup>26</sup>validate many aspects of causal estimation using standard techniques.

<sup>27</sup> The effectiveness of a given approach is often determined by how much we trust the specification of  
<sup>28</sup>our propensity score or outcome regression models  $\hat{g}(x)$  and  $\hat{Q}(a, x)$ , and how well the treatment and  
<sup>29</sup>control groups overlap in the dataset. Using flexible models for the nuisance functions  $g$  and  $Q$  can  
<sup>30</sup>alleviate some of the concerns about model misspecification, but our freedom to use such models is  
<sup>31</sup>often constrained by dataset size. When we have the luxury of large data, we can use flexible models;  
<sup>32</sup>on the other hand, when the dataset is relatively small, we may need to use a smaller parametric  
<sup>33</sup>family or stringent regularization to obtain stable estimates of  $Q$  and  $g$ . Similarly, if overlap is poor  
<sup>34</sup>in some regions of the covariate space, then flexible models for  $Q$  may be highly variable, and inverse  
<sup>35</sup>propensity score weights may be large. In these cases, IPTW or AIPTW estimates may fluctuate  
<sup>36</sup>wildly as a function of large weights. Meanwhile, outcome regression estimates will be sensitive to  
<sup>37</sup>the specification of the  $Q$  model and its regularization, and can incur bias that is difficult to measure  
<sup>38</sup>if the specification or regularization does not match the true outcome process.

<sup>39</sup> There are a number of practical steps that we can take to sanity-check causal estimates. The  
<sup>40</sup>simplest check is to compute many different ATE estimators (e.g., outcome regression, IPTW, doubly  
<sup>41</sup>robust) using several comparably complex estimators of  $Q$  and  $g$ . We can then check whether they  
<sup>42</sup>agree, at least qualitatively. If they do agree then this can provide some peace of mind (although it  
<sup>43</sup>is not a guarantee of accuracy). If they disagree, caution is warranted, particularly in choosing the  
<sup>44</sup>specification of the  $Q$  and  $g$  models.

<sup>45</sup> It is also important to check for failures of overlap. Often, issues such as disagreement between  
<sup>46</sup>alternative estimators can be traced back to poor overlap. A common way to do this, particularly  
<sup>47</sup>



*Figure 38.6: Causal graph illustrating the Instrumental Variable setup. The treatment  $A$  and outcome  $Y$  are both influenced by unobserved confounder  $U$ . Nevertheless, identification is sometimes possible due to the presence of the instrument  $Z$ . We also allow for observed covariates  $X$  that we may need to adjust for. The dashed arrow between  $U$  and  $X$  indicates a statistical dependency where we remain agnostic to the particular causal relationship.*

with high-dimensional data, is to examine the estimated (ideally cross-fitted) propensity scores  $\hat{g}(x_i)$ . This is a useful diagnostic, even if the intention is to use an outcome regression approach that only incorporates and estimated outcome regression function  $\hat{Q}(a, x_i)$ . If overlap issues are relevant, it may be better to instead estimate either the average treatment effect on the treated, or the “trimmed” estimand given by discarding units with extreme propensities.

Uncertainty quantification is also an essential part of most causal analyses. This frequently take the form of an estimate of the estimator’s variance, or a confidence interval. This may be important for downstream decision-making, and can also be a useful diagnostic. We can calculate variance either by bootstrapping the entire procedure (including refitting the models in each bootstrap replicate), or computing analytical variance estimates from the AIPTW estimator. Generally, large variance estimates may indicate issues with the analysis. For example, poor overlap will often (although not always) manifest as extremely large variances under either of these methods. Small variance estimates should be treated with caution, unless other checks, such as overlap checks, or stability across different  $Q$  and  $g$  models, also pass.

The previous advice only addresses the statistical problem of estimating  $\tau$  from a data sample. It does not speak to whether or not  $\tau$  can reasonably be interpreted as an average treatment effect. Considerable care should be devoted to whether or not the assumption that there are no unobserved confounders is reasonable. There are several methods for assessing the sensitivity of the ATE estimate to violations of this assumption. See Section 38.7. Bias due to unobserved confounding can be substantial in practice—often overwhelming bias due to estimation error—so it is wise to conduct such an analysis.

## 38.5 Instrumental Variable Strategies

Adjustment-based methods rely on observing all confounders affecting the treatment and outcome. In some situations, it is possible to identify interesting causal effects even when there are unobserved confounders. We now consider strategies based on **instrumental variables**. The instrumental

variable graph is shown in Figure 38.6. The key ingredient is the instrumental variable  $Z$ , a variable that has a causal effect on  $Y$  only through its causal effect on  $A$ . Informally, the identification strategy is to determine the causal effect of  $Z$  on  $Y$ , the causal effect of  $Z$  on  $A$ , and then combine these into an estimate of the causal effect of  $A$  on  $Y$ .

For this identification strategy to work the instrument must satisfy three conditions. There are observed variables (confounders)  $X$  such that:

1. **Instrument Relevance**  $Z \not\perp\!\!\!\perp A|X$ : the instrument must actually affect the treatment assignment.
2. **Instrument Unconfoundedness** Any backdoor path between  $Z$  and  $Y$  is blocked by  $X$ , even conditional on  $A$ .
3. **Exclusion Restriction** All directed paths from  $Z$  to  $Y$  pass through  $A$ . That is, the instrument affects the outcome *only* through its effect on  $A$ .

(It may help conceptually to first think through the case where  $X$  is the empty set—i.e., where the only confounder is the unobserved  $U$ ). These assumptions are necessary for using instrumental variables for causal identification, but they are not quite sufficient. In practice, they must be supplemented by an additional assumption that depends more closely on the details of the problem at hand. Historically, this additional assumption was usually that both the instrument-treatment and treatment-outcome relationship are linear. We'll examine some less restrictive alternatives below.

Before moving on to how to use instrumental variables for identification, let's consider how we might encounter instruments in practice. The key is that it's often possible to find, and measure, variables that affect treatment and that are assigned (as if) at random. For example, suppose we are interested in measuring the effect of taking a drug  $A$  on some health outcome  $Y$ . The challenge is that whether a study participant actually takes the drug can be confounded with  $Y$ —e.g., sicker people may be more likely to take their medication, but have worse outcomes. However, the assignment of treatments to patients can be randomized and this random assignment can be viewed as an instrument. This **random assignment with non-compliance** scenario is common in practice. The random assignment—the instrument—satisfies relevance (so long as assigning the drug affects the probability of the patient taking the drug). It also satisfies unconfoundedness (because the instrument is randomized). And, it plausibly satisfies exclusion restriction: telling (or not telling) a patient to take a drug has no effect on their health outcome except through influencing whether or not they actually take the drug. As a second example, the **judge fixed effects** research design uses the identity of the judge assigned to each criminal case to infer the effect of incarceration on some life outcome of interest (e.g., total lifetime earnings). Relevance will be satisfied so long as different judges have different propensities to hand out severe sentences. The assignment of trial judges to cases is randomized, so unconfoundedness will also be satisfied. And, exclusion restriction is also plausible: the particular identity of the judge assigned to your case has no bearing on your years-later life outcomes, except through the particular sentence that you're subjected to.

It's important to note that these assumptions require some care, particularly exclusion restriction. Relevance can be checked directly from the data, by fitting a model to predict the treatment from the instrument (or vice versa). Unconfoundedness is often satisfied by design: the instrument is randomly assigned. Even when literal random assignment doesn't hold, we often restrict to instruments where unconfoundedness is “obviously” satisfied—e.g., using number of rainy days in a month as an instrument for sun exposure. Exclusion restriction is trickier. For example, it might fail in the drug assignment case if patients who are not told to take a drug respond by seeking out alternative

treatment. Or, it might fail in the judge fixed effects case if judges hand out additional, unrecorded, punishments in addition to incarceration. Assessing the plausibility of exclusion restriction requires careful consideration based on domain expertise.

We now return to the question of how to make use of an instrument once we have it in hand. As previously mentioned, getting causal identification using instrumental variables requires supplementing the IV assumptions with some additional assumption about the causal process.

### 38.5.1 Additive Unobserved Confounding

We first consider **additive unobserved confounding**. That is, we assume that the structural causal model for the outcome has the form:<sup>7</sup>

$$Y \leftarrow f(A, X) + f_U(U). \quad (38.49)$$

In words, we assume that there are no interaction effects between the treatment and the unobserved confounder—everyone responds to treatment in the same way. With this additional assumption, we see that  $\mathbb{E}[Y|X, \text{do}(A = a)] - \mathbb{E}[Y|X, \text{do}(A = a')] = f(a, X) - f(a', X)$ . In this setting, our goal is to learn this contrast.

**Theorem 3** (Additive Confounding Identification). *If the instrumental variables assumptions hold and also additive unobserved confounding holds, then there is a function  $\tilde{f}(a, x)$  where*

$$\mathbb{E}[Y|x, \text{do}(A = a)] - \mathbb{E}[Y|x, \text{do}(A = a')] = \tilde{f}(a, x) - \tilde{f}(a', x), \quad (38.50)$$

for all  $x, a, a'$  and such that  $\tilde{f}$  satisfies

$$\mathbb{E}[Y|z, x] = \int \tilde{f}(a, x)p(a|z, x)da. \quad (38.51)$$

Here,  $p(a|z, x)$  is the conditional probability density of treatment.

In particular, if there is a unique function  $g$  that satisfies

$$\mathbb{E}[Y|z, x] = \int g(a, x)p(a|z, x)da, \quad (38.52)$$

then  $g = \tilde{f}$  and this relation identifies the target causal effect.

Before giving the proof, let's understand the point of this identification result. The key insight is that both the left hand side of Equation (38.52) and  $p(a|z, x)$  (appearing in the integrand) are identified by the data, since they involve only observational relationships between observed variables. So,  $\tilde{f}$  is identified implicitly as one of the functions that makes Equation (38.52) true. If there is a unique such function, then this fully identifies the causal effect.

*Proof.* With the additive unobserved confounding assumption, the instrument unconfoundedness implies that  $U \perp\!\!\!\perp Z|X$ . Then, we have that:

$$\mathbb{E}[Y|Z, X] = \mathbb{E}[f(A, X)|Z, X] + \mathbb{E}[f_U(U)|Z, X] \quad (38.53)$$

$$= \mathbb{E}[f(A, X)|Z, X] + \mathbb{E}[f_U(U)|X] \quad (38.54)$$

$$= \mathbb{E}[\tilde{f}(A, X)|Z, X], \quad (38.55)$$

7. We roll the unit-specific variables  $\xi$  into  $U$  to avoid notational overload.

<sup>1</sup> where  $\tilde{f} = f(A, X) + \mathbb{E}[f_U(U)|X]$ . Now, identifying just  $\tilde{f}$  would suffice for us, because we could then  
<sup>2</sup> identify contrasts between treatments:  $f(a, x) - f(a', x) = \tilde{f}(a, x) - \tilde{f}(a', x)$ . (The term  $\mathbb{E}[f_U(U)|x]$   
<sup>3</sup> cancels out). Accordingly, we rewrite Equation (38.55) as:  
<sup>4</sup>

$$\mathbb{E}[Y|z, x] = \int \tilde{f}(a, x)p(a|z, x)da. \quad (38.56)$$

□

<sup>5</sup> It's worth dwelling briefly on how the IV assumptions come into play here. The exclusion restriction  
<sup>6</sup> is implied by the additive unobserved confounding assumption, which we use explicitly. We also use  
<sup>7</sup> the unconfoundedness assumption to conclude  $U \perp\!\!\!\perp Z|X$ . However, we do not use relevance. The  
<sup>8</sup> role of relevance here is in ensuring that few functions solve the relation Equation (38.52). Informally,  
<sup>9</sup> the solution  $g$  is constrained by the requirement that it hold for all values of  $Z$ . However, different  
<sup>10</sup> values of  $Z$  only add non-trivial constraints if  $p(a|z, x)$  differ depending on the value of  $z$ —this is  
<sup>11</sup> exactly the relevance condition.  
<sup>12</sup>

<sup>13</sup> **Estimation** The basic estimation strategy is to fit models for  $\mathbb{E}[Y|z, x]$  and  $p(a|z, x)$  from the data,  
<sup>14</sup> and then solve the implicit equation Equation (38.52) to find  $g$  consistent with the fitted models.  
<sup>15</sup> The procedures for doing this can vary considerably depending on the particulars of the data (e.g., if  
<sup>16</sup>  $Z$  is discrete or continuous) and the choice of modeling strategy. We omit a detailed discussion, but  
<sup>17</sup> [see e.g. NP03; Dar+11; Har+17; SSG19; BKS19; Mua+20; Dik+20] for various concrete approaches.  
<sup>18</sup>

<sup>19</sup> It's also worth mentioning an additional nuance to the general procedure. Even if relevance holds,  
<sup>20</sup> there will often be more than one function that satisfies Equation (38.52). So, we have only identified  
<sup>21</sup>  $\tilde{f}$  as a member of this set of functions. In practice, this ambiguity is defeated by making some  
<sup>22</sup> additional structural assumption about  $\tilde{f}$ . For example, we model  $\tilde{f}$  with a neural network, and then  
<sup>23</sup> choose the network satisfying Equation (38.52) that has minimum  $l_2$ -norm on the parameters (i.e.,  
<sup>24</sup> we pick the  $l_2$ -regularized solution).  
<sup>25</sup>

### <sup>26</sup> 38.5.2 Instrument Monotonicity and Local Average Treatment Effect

<sup>27</sup> We now consider an alternative assumption to additive unobserved confounding that is applicable  
<sup>28</sup> when both the instrument and treatment are binary. It will be convenient to conceptualize the  
<sup>29</sup> instrument as assignment-to-treatment. Then, the population divides into four subpopulations:  
<sup>30</sup>

- <sup>31</sup> 1. Compliers, who take the treatment if assigned to it, and who don't take the treatment otherwise.
- <sup>32</sup> 2. Always takers, who take the treatment no matter their assignment
- <sup>33</sup> 3. Never takers, who refuse the treatment no matter their assignment
- <sup>34</sup> 4. Defiers, who refuse the treatment if assigned to it, and who take the treatment if not assigned.

<sup>35</sup> Our goal in this setting will be to identify the average treatment effect among the compliers. The  
<sup>36</sup> local average treatment effect (or complier average treatment effect) is defined to be<sup>8</sup>

$$\text{LATE} = \mathbb{E}[Y|\text{do}(A = 1), \text{complier}] - \mathbb{E}[Y|\text{do}(A = 0), \text{complier}]. \quad (38.57)$$

<sup>43</sup> 8. We follow the econometrics literature in using “LATE” because “CATE” is already commonly used for conditional  
<sup>44</sup> average treatment effect.  
<sup>45</sup>

The LATE requires an additional assumption for identification. Namely, **instrument monotonicity**: being assigned (not assigned) the treatment only increases (decreases) the probability that each unit will take the treatment. Equivalently,  $P(\text{defier}) = 0$ .

We can then write down the identification result.

**Theorem 4.** *Given the instrumental variable assumptions and instrument monotonicity, the local average treatment is identified as a parameter  $\tau^{\text{LATE}}$  of the observational distributional; that is,  $\text{LATE} = \tau^{\text{LATE}}$ . Namely,*

$$\tau^{\text{LATE}} = \frac{\mathbb{E}[\mathbb{E}[Y|X, Z = 1] - \mathbb{E}[Y|X, Z = 0]]}{\mathbb{E}[P(A = 1|X, Z = 1) - P(A = 1|X, Z = 0)]}. \quad (38.58)$$

*Proof.* We now show that, given the IV assumptions and monotonicity,  $\text{LATE} = \tau^{\text{LATE}}$ . First, notice that

$$\tau^{\text{LATE}} = \frac{\mathbb{E}[Y|\text{do}(Z = 1)] - \mathbb{E}[Y|\text{do}(Z = 0)]}{P(A = 1|\text{do}(Z = 1)) - P(A = 1|\text{do}(Z = 0))}. \quad (38.59)$$

This follows from backdoor adjustment, Theorem 2, applied to the numerator and denominator separately. Our strategy will be to decompose  $\mathbb{E}[Y|\text{do}(Z = z)]$  into the contributions from the compliers, the units that ignore the instrument (the always/never takers), and the defiers. To that end, note that  $P(\text{complier}|\text{do}(Z = z)) = P(\text{complier})$  and similarly for always/never takers and defiers—interventions on the instrument don’t change the composition of the population. Then,

$$\mathbb{E}[Y|\text{do}(Z = 1)] - \mathbb{E}[Y|\text{do}(Z = 0)] \quad (38.60)$$

$$= (\mathbb{E}[Y|\text{complier}, \text{do}(Z = 1)] - \mathbb{E}[Y|\text{complier}, \text{do}(Z = 0)])P(\text{complier}) \quad (38.61)$$

$$+ (\mathbb{E}[Y|\text{always/never}, \text{do}(Z = 1)] - \mathbb{E}[Y|\text{always/never}, \text{do}(Z = 0)])P(\text{always/never}) \quad (38.62)$$

$$+ (\mathbb{E}[Y|\text{defier}, \text{do}(Z = 1)] - \mathbb{E}[Y|\text{defier}, \text{do}(Z = 0)])P(\text{defier}). \quad (38.63)$$

The key is the effect on the complier subpopulation, Equation (38.61). First, by definition of the complier population, we have that:

$$\mathbb{E}[Y|\text{complier}, \text{do}(Z = z)] = \mathbb{E}[Y|\text{complier}, \text{do}(A = z)]. \quad (38.64)$$

That is, the causal effect of the treatment is the same as the causal effect of the instrument in this subpopulation—this is the core reason why access to an instrument allows identification of the local average treatment effect. This means that

$$\text{LATE} = \mathbb{E}[Y|\text{complier}, \text{do}(Z = 1)] - \mathbb{E}[Y|\text{complier}, \text{do}(Z = 0)]. \quad (38.65)$$

Further, we have that  $P(\text{complier}) = P(A = 1|\text{do}(Z = 1)) - P(A = 1|\text{do}(Z = 0))$ . The reason is simply that, by definition of the subpopulations,

$$P(A = 1|\text{do}(Z = 1)) = P(\text{complier}) + P(\text{always taker}) \quad (38.66)$$

$$P(A = 1|\text{do}(Z = 0)) = P(\text{always taker}). \quad (38.67)$$

1 Now, plugging the expression for  $P(\text{complier})$  and Equation (38.65) into Equation (38.61) we have  
2 that:

$$\begin{aligned} \underline{4} \quad & (\mathbb{E}[Y|\text{complier}, \text{do}(Z=1)] - \mathbb{E}[Y|\text{complier}, \text{do}(Z=0)])P(\text{complier}) \\ \underline{5} \quad & = \text{LATE} \times (P(A=1|\text{do}(Z=1)) - P(A=1|\text{do}(Z=0))) \end{aligned} \quad (38.68)$$

$$\underline{6} \quad = \text{LATE} \times (P(A=1|\text{do}(Z=1)) - P(A=1|\text{do}(Z=0))) \quad (38.69)$$

7 This gives us an expression for the local average treatment effect in terms of the effect of the instrument  
8 on the compliers and the probability that a unit takes the treatment when assigned/not-assigned.  
9

10 The next step is to show that the remaining instrument effect decomposition terms, Equa-  
11 tions (38.62) and (38.63), are both 0. Equation (38.62) is the causal effect of the instrument on the  
12 always/never takers. It's equal to 0 because, by definition of this subpopulation, the instrument  
13 has no causal effect in the subpopulation—they ignore the instrument! Mathematically, this is just  
14  $\mathbb{E}[Y|\text{always/never}, \text{do}(Z=1)] = \mathbb{E}[Y|\text{always/never}, \text{do}(Z=0)]$ . Finally, Equation (38.63) is 0 by the  
15 instrument monotonicity assumption: we assumed that  $P(\text{defier}) = 0$ .

16 In totality, we now have that Equations (38.61) to (38.63) reduces to:

$$\begin{aligned} \underline{17} \quad & \mathbb{E}[Y|\text{do}(Z=1)] - \mathbb{E}[Y|\text{do}(Z=0)] \\ \underline{18} \quad & = \text{LATE} \times (P(A=1|\text{do}(Z=1)) - P(A=1|\text{do}(Z=0))) + 0 + 0 \end{aligned} \quad (38.70)$$

$$\begin{aligned} \underline{19} \quad & = \text{LATE} \times (P(A=1|\text{do}(Z=1)) - P(A=1|\text{do}(Z=0))) + 0 + 0 \\ \underline{20} \quad & \text{Rearranging for LATE and plugging in to Equation (38.59) gives claimed identification result. } \quad \square \end{aligned} \quad (38.71)$$

### 22 38.5.2.1 Estimation

24 For estimating the local average treatment effect under the monotone instrument assumption, there  
25 is a double-machine learning approach that works with generic supervised learning approaches. Here,  
26 we want an estimator  $\hat{\tau}^{\text{LATE}}$  for the parameter

$$\begin{aligned} \underline{27} \quad & \tau^{\text{LATE}} = \frac{\mathbb{E}[\mathbb{E}[Y|X, Z=1] - \mathbb{E}[Y|X, Z=0]]}{\mathbb{E}[P(A=1|X, Z=1) - P(A=1|X, Z=0)]}. \end{aligned} \quad (38.72)$$

31 To define the estimator, it's convenient to introduce some additional notation. First, we define the  
32 nuisance functions:

$$\begin{aligned} \underline{33} \quad & \mu(z, x) = \mathbb{E}[Y|z, x] \\ \underline{34} \quad & m(z, x) = P(A=1|x, z) \end{aligned} \quad (38.73) \quad (38.74)$$

$$\begin{aligned} \underline{35} \quad & p(x) = P(Z=1|x). \\ \underline{36} \quad & \end{aligned} \quad (38.75)$$

38 We also define the score  $\phi$  by:

$$\begin{aligned} \underline{39} \quad & \phi_{Z \rightarrow Y}(\mathbf{X}; \mu, p) \triangleq \mu(1, X) - \mu(0, X) + \frac{Z(Y - \mu(1, X))}{p(X)} - \frac{(1-Z)(Y - \mu(0, X))}{1-p(X)} \\ \underline{40} \quad & \end{aligned} \quad (38.76)$$

$$\begin{aligned} \underline{42} \quad & \phi_{Z \rightarrow A}(\mathbf{X}; m, p) \triangleq m(1, X) - m(0, X) + \frac{Z(A - m(1, X))}{p(X)} - \frac{(1-Z)(A - m(0, X))}{1-p(X)} \\ \underline{43} \quad & \end{aligned} \quad (38.77)$$

$$\begin{aligned} \underline{44} \quad & \phi(\mathbf{X}; \mu, m, p, \tau) \triangleq \phi_{Z \rightarrow Y}(\mathbf{X}; \mu, p) - \phi_{Z \rightarrow A}(\mathbf{X}; m, p) \times \tau \\ \underline{45} \quad & \end{aligned} \quad (38.78)$$

46 Then, the estimator is defined by a two stage procedure:

47

1    1. Fit models  $\hat{\mu}, \hat{m}, \hat{p}$  for each of  $\mu, m, p$  (using supervised machine learning).

2    2. Define  $\hat{\tau}^{\text{LATE}}$  as the solution to  $\frac{1}{n} \sum_i \phi(\mathbf{X}_i; \hat{\mu}, \hat{m}, \hat{p}, \hat{\tau}^{\text{LATE}}) = 0$ . That is,

$$\hat{\tau}^{\text{LATE}} = \frac{\frac{1}{n} \sum_i \phi_{Z \rightarrow Y}(\mathbf{X}_i; \hat{\mu}, \hat{p})}{\frac{1}{n} \sum_i \phi_{Z \rightarrow A}(\mathbf{X}_i; \hat{m}, \hat{p})} \quad (38.79)$$

It may help intuitions to notice that the double machine learning estimator of the LATE is effectively the double machine learning estimator of the average treatment effect of  $Z$  on  $Y$  divided by the double machine learning estimator of the average treatment effect of  $Z$  on  $A$ .

Similarly to Section 38.4, the nuisance functions can be estimated by:

1. fit a model  $\hat{\mu}$  that predicts  $Y$  from  $Z, X$  by minimizing mean square error

2. fit a model  $\hat{m}$  that predicts  $A$  from  $Z, X$  by minimizing mean cross-entropy

3. fit a model  $\hat{p}$  that predicts  $Z$  from  $X$  by minimizing mean cross-entropy.

As in Section 38.4, reusing the same data for model fitting and computing the estimator can potentially cause problems. This can be avoided with use a cross-fitting procedure as described in Section 38.4.2.4. In this case, we split the data into  $K$  folds and, for each fold  $k$ , use all the but the  $k$ th fold to compute estimates  $\hat{\mu}_{-k}, \hat{m}_{-k}, \hat{p}_{-k}$  of the nuisance parameters. Then we compute the nuisance estimates for each datapoint  $i$  in fold  $k$  by predicting the required quantity using the nuisance model fit on the other folds. That is, if unit  $i$  is in fold  $k$ , we compute  $\hat{\mu}(z_i, x_i) \triangleq \hat{\mu}^{-k}(z_i, x_i)$  and so forth.

The key result is that if we use the cross-fit version of the estimator and the estimators for the nuisance functions converge to their true values in the sense that

1.  $\mathbb{E}(\hat{\mu}(Z, X) - \mu(Z, X))^2 \rightarrow 0$ ,  $\mathbb{E}(\hat{m}(Z, X) - m(Z, X))^2 \rightarrow 0$ , and  $\mathbb{E}(\hat{p}(X) - p(X))^2 \rightarrow 0$

2.  $\sqrt{\mathbb{E}[(\hat{p}(X) - p(X))^2]} \times (\sqrt{\mathbb{E}[(\hat{\mu}(Z, X) - \mu(Z, X))^2]} + \sqrt{\mathbb{E}[(\hat{m}(Z, X) - m(Z, X))^2]}) = o(\sqrt{n})$

then (with some omitted technical conditions) we have asymptotic normality at the  $\sqrt{n}$ -rate:

$$\sqrt{n}(\hat{\tau}^{\text{LATE}-\text{cf}} - \tau^{\text{LATE}}) \xrightarrow{d} \text{Normal}(0, \frac{\mathbb{E}[\phi(\mathbf{X}; \mu, m, p, \tau^{\text{LATE}})^2]}{\mathbb{E}[m(1, X) - m(0, X)]^2}). \quad (38.80)$$

As with double machine learning for the confounder adjustment strategy, the key point here is that we can achieve the (optimal)  $\sqrt{n}$  rate for estimating the LATE under a relatively weak condition on how well we estimate the nuisance functions—what matters is the *product* of the error in  $p$  and the errors in  $\mu, m$ . So, for example, a very good model for how the instrument is assigned ( $p$ ) can make up for errors in the estimation of the treatment-assignment ( $m$ ) and outcome ( $\mu$ ) models.

The double machine learning estimator also gives a recipe for quantifying uncertainty. To that end, define

$$\hat{\tau}_{Z \rightarrow A} \triangleq \frac{1}{n} \sum_i \phi_{Z \rightarrow A}(\mathbf{X}_i; \hat{m}, \hat{p}) \quad (38.81)$$

$$\hat{\mathbb{V}}[\hat{\tau}^{\text{LATE}}] \triangleq \frac{1}{\hat{\tau}_{Z \rightarrow A}^2} \frac{1}{n} \sum_i \phi(\mathbf{X}_i; \hat{\mu}, \hat{m}, \hat{p}, \hat{\tau}^{\text{LATE}})^2. \quad (38.82)$$

<sup>1</sup> Then, subject to suitable technical conditions,  $\hat{\mathbb{V}}[\hat{\tau}^{\text{LATE}-\text{cf}}]$  can be used as an estimate of the variance  
<sup>2</sup> of the estimator. More precisely,

<sup>4</sup>

$$\sqrt{n}(\hat{\tau}^{\text{LATE}} - \tau^{\text{LATE}}) \xrightarrow{d} \text{Normal}(0, \hat{\mathbb{V}}[\hat{\tau}^{\text{LATE}}]). \quad (38.83)$$

<sup>5</sup>

<sup>6</sup> Then, confidence intervals or  $p$ -values can be computed using this variance in the usual way. The main  
<sup>7</sup> extra condition required for the variance estimator to be valid is that the nuisance parameters must  
<sup>8</sup> all converge at rate  $O(n^{-1/4})$  (so an excellent estimator for one can't fully compensate for terrible  
<sup>9</sup> estimators of the others). In fact, even this condition is unnecessary in certain special cases—e.g.,  
<sup>10</sup> when  $p$  is known exactly, which occurs when the instrument is randomly assigned. See Chernozhukov  
<sup>11</sup> et al. [Che+17e] for technical details.  
<sup>12</sup>

<sup>13</sup>

### <sup>14</sup> 38.5.3 Two Stage Least Squares

<sup>15</sup> Commonly, the IV assumptions are supplemented with the following linear model assumptions:  
<sup>16</sup>

<sup>17</sup>

$$A_i \leftarrow \alpha_0 + \alpha Z_i + \delta_A X_i + \gamma_A X_i + \xi_i^A \quad (38.84)$$

<sup>18</sup>

$$Y_i \leftarrow \beta_0 + \beta A_i + \delta_Y X_i + \gamma_Y X_i + \xi_i^Y \quad (38.85)$$

<sup>19</sup>

<sup>20</sup> That is, we assume that the real-world process for treatment assignment and the outcome are both  
<sup>21</sup> linear. In this case, plugging Equation (38.84) into Equation (38.85) yields  
<sup>22</sup>

<sup>23</sup>

$$Y_i \leftarrow \tilde{\beta}_0 + \beta \alpha Z_i + \tilde{\delta} X_i + \tilde{\gamma} X_i + \tilde{\xi}_i. \quad (38.86)$$

<sup>24</sup>

<sup>25</sup> The point is that  $\beta$ , the average treatment effect of  $A$  on  $Y$ , is equal to the coefficient  $\beta \alpha$  of the  
<sup>26</sup> instrument in the outcome-instrument model divided by the coefficient  $\alpha$  of the instrument in the  
<sup>27</sup> treatment-instrument model. So, to estimate the treatment effect, we simply fit both linear models  
<sup>28</sup> and divide the estimated coefficients. This procedure is called **two stage least squares**.

<sup>29</sup> The simplicity of this procedure is seductive. However, the required linearity assumptions are hard  
<sup>30</sup> to satisfy in practice and frequently lead to severe issues. A particularly pernicious version of this  
<sup>31</sup> is that linear-model misspecification together with weak relevance can yield standard errors for the  
<sup>32</sup> estimate that are far too small. In practice, this can lead us to find large, significant estimates from  
<sup>33</sup> two stage least squares when the truth is actually a weak or null effect. See [Rei16; You19; ASS19;  
<sup>34</sup> Lal+21] for critical evaluations of two stage least squares in practice.

<sup>35</sup>

## <sup>36</sup> 38.6 Difference in Differences

<sup>37</sup> Unsurprisingly, time plays an important role in causality. Causes precede effects, and we should be  
<sup>38</sup> able to incorporate this knowledge into causal identification. We now turn to a particular strategy  
<sup>39</sup> for causal identification that relies on observing each unit at multiple time points. Data of this kind  
<sup>40</sup> is sometimes called **panel data**. We'll consider the simplest case. There are two time periods. In  
<sup>41</sup> the first period, none of the units are treated, and we observe an outcome  $Y_{0i}$  for each unit. Then,  
<sup>42</sup> a subset of the units are treated, denoted by  $A_i = 1$ . In the second time period, we again observe  
<sup>43</sup> the outcomes  $Y_{1i}$  for each unit, where now the outcomes of the treated units are affected by the  
<sup>44</sup> treatment. Our goal is to determine the average effect receiving the treatment had on the treated  
<sup>45</sup> units. That is, we want to know the average difference between the outcomes we actually observed  
<sup>46</sup>

<sup>47</sup>

for the treated units, and the outcomes we would have observed on those same units if they had not been treated. The general strategy we look at is called **difference in differences**.

As a concrete motivating example, consider trying to determine the effect raising minimum wage on employment. The concern here is that, in an efficient labor market, increasing the price of workers will reduce the demand for them, thereby driving down employment. As such, it seems increasing minimum wage may hurt the people the policy is nominally intended to help. The question is: how strong is this effect in practice? Card and Krueger [CK94a] studied this effect using difference in differences. The Philadelphia metropolitan area includes regions in both Pennsylvania and New Jersey (different US states). On April 1st 1992, New Jersey raised its minimum wage from \$4.25 to \$5.05. In Pennsylvania, the wage remained constant at \$4.25. The strategy is to collect employment data from fast food restaurants (which pay many employees minimum wage) in each state before and after the change in minimum wage. In this case, for restaurant  $i$ , we have  $Y_{0i}$ , the number of full time employees in February 1992, and  $Y_{1i}$ , the number of full time employees in November 1992. The treatment is simply  $A_i = 1$  if the restaurant was located in New Jersey, and  $A_i = 0$  if located in Pennsylvania. Our goal is to estimate the average effect of the minimum wage hike on employment in the restaurants affected by it (i.e., the ones in New Jersey).

The assumption in classical difference-in-differences is the following structural equation:

$$Y_{ti} \leftarrow W_i + S_t + \tau A_i 1[t = 1] + \xi_{ti}, \quad (38.87)$$

with  $\mathbb{E}[\xi_{ti}|W_i, S_t, A_i] = 0$ . Here,  $W_i$  is a unit specific effect that is constant across time (e.g., the location of the restuarant or competence of the management) and  $S_t$  is a time-specific effect that applies to all units (e.g., the state of the US economy at each time). Both of these quantities are treated as unobserved, and not explicitly accounted for. The parameter  $\tau$  captures the target causal effect. The (strong) assumption here is that unit, time, and treatment effects are all additive. This assumption is called **parallel trends**, because it is equivalent to assuming that, in the absence of treatment, the trend over time would be the same in both groups. It's easy to see that under this assumption, we have:

$$\tau = \mathbb{E}[Y_{1i} - Y_{0i}|A = 1] - \mathbb{E}[Y_{1i} - Y_{0i}|A = 0]. \quad (38.88)$$

That is, the estimand first computes the difference across time for both the treated and untreated group, and then computes the difference between these differences across the groups. The obvious estimator is then

$$\hat{\tau} = \frac{1}{n_A} \sum_{i:A_i=1} Y_{1i} - Y_{0i} - \frac{1}{n - n_A} \sum_{i:A_i=0} Y_{1i} - Y_{0i}, \quad (38.89)$$

where  $n_A$  is the number of treated units.

The root identification problem addressed by difference-in-differences is that  $\mathbb{E}[W_i|A_i = 1] \neq \mathbb{E}[W_i|A_i = 0]$ . That is, restaurants in New Jersey may be systematically different from restuarants in Pennsylvania in unobserved ways that affect employment.<sup>9</sup> This is why we can't simply compare average outcomes for the treated and untreated. The identification assumption is that this unit-specific effect is the only source of statistical association with treatment; in particular we assume the

<sup>9</sup> 9. This is similar to the issue that arises from unobserved confounding, except  $W_i$  need not be a cause of the treatment assignment.

1 time-specific effect has no such issue:  $\mathbb{E}[S_{1i} - S_{0i}|A_i = 1] = \mathbb{E}[S_{1i} - S_{0i}|A_i = 0]$ . Unfortunately, this  
2 assumption can be too strong. For instance, administrative data shows employment in Pennsylvania  
3 falling relative to employment in New Jersey between 1993 and 1996 [AP08, §5.2]. Although this  
4 doesn't directly contradict the parallel trends assumption used for identification, which needs to  
5 hold only in 1992, it does make it seem less credible.  
6

7 To weaken the assumption, we'll look at a version that requires parallel trends to hold only after  
8 adjusting for covariates. To motivate this, we note that there were several different types of fast  
9 food restaurant included in the employment data. These vary, e.g., in the type of food they serve,  
10 and in cost per meal. Now, it seems reasonable the trend in employment may depend on the type  
11 of restaurant. For example, more expensive chains (such as Kentucky Fried Chicken) might be  
12 more affected by recessions than cheaper chains (such as McDonald's). If expensive chains are more  
13 common in New Jersey than in Pennsylvania, this effect can create a violation of parallel trends—if  
14 there's recession affecting both states, we'd expect employment to go down more in New Jersey than  
15 in Pennsylvania. However, we may find it credible that McDonald's restaurants in New Jersey have  
16 the same trend as McDonald's in Pennsylvania, and similarly for Kentucky Fried Chicken.

17 The next step is to give a definition of the target causal effect that doesn't depend on a parametric  
18 model, and a non-parametric statement of the identification assumption to go with it. In words, the  
19 causal estimand will be the average treatment effect on the units that received the treatment. To  
20 make sense of this mathematically, we'll introduce a new piece of notation:

$$\mathbb{P}^{A=1}(Y|do(A=a)) \triangleq \int P(Y|A=a, \text{parents of } Y)dP(\text{parents of } Y|A=1) \quad (38.90)$$

$$\mathbb{E}^{A=1}[Y|do(A=a)] \triangleq \mathbb{E}_{\mathbb{P}^{A=1}(Y|do(A=a))}[Y]. \quad (38.91)$$

25 In words: recall that the ordinary do operator works by replacing  $P(\text{parents}|A=a)$  by the marginal  
26 distribution  $P(\text{parents})$ , thereby breaking the backdoor associations. Now, we're replacing the  
27 distribution  $P(\text{parents}|A=a)$  by  $P(\text{parents}|A=1)$ , irrespective of the actual treatment value. This  
28 still breaks all backdoor associations, but is a better match for our target of estimating the treatment  
29 effect only among the treated units.

30 To formalize a causal estimand using the do calculus, we need to assume some partial causal  
31 structure. We'll use the graph in Figure 38.7. With this in hand, our causal estimand is the average  
32 treatment effect on the units that received the treatment, namely:

$$\text{ATT}^{\text{DiD}} = \mathbb{E}^{A=1}[Y_1 - Y_0|do(A=1)] - \mathbb{E}^{A=1}[Y_1 - Y_0|do(A=0)] \quad (38.92)$$

35 In the minimum wage example, this is the average effect of the minimum wage hike on employment  
36 in the restaurants affected by it (i.e., the ones in New Jersey).

37 Finally, we formalize the identification assumption that, conditional on  $X$ , the trends in the treated  
38 and untreated groups are the same. The **conditional parallel trends** assumption is:

$$\mathbb{E}^{A=1}[Y_1 - Y_0|X, do(A=0)] = \mathbb{E}[Y_1 - Y_0|X, A=0]. \quad (38.93)$$

40 In words, this says that for treated units with covariates  $X$ , the trend we would have seen had we not  
41 assigned treatment is the same as the trend we actually saw for the untreated units with covariates  
42  $X$ . That is, if New Jersey had not raised its minimum wage, then McDonald's in New Jersey would  
43 have the same expected change in employment as McDonald's in Pennsylvania.

44 With this in hand, we can give the main identification result:

45

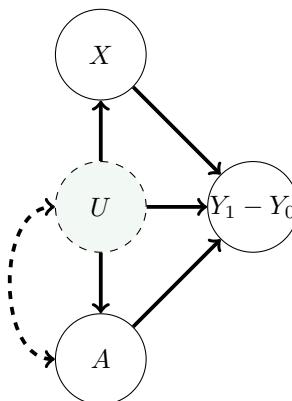


Figure 38.7: Causal graph assumed for the difference-in-differences setting. Here, the outcome of interest is the difference between the pre- and post-treatment period,  $Y_1 - Y_0$ . This difference is influenced by the treatment, unobserved factors  $U$ , and observed covariates  $X$ . The dashed arrow between  $U$  and  $A$  indicates a statistical dependency between the variables, but where we remain agnostic to the precise causal mechanism. For example, in the minimum wage example,  $U$  might be the average income in restaurant's neighbourhood, which is dependent on the state, and hence also the treatment.

**Theorem 5** (Difference in Differences Identification). We observe  $A, Y_0, Y_1, X \sim P$ . Suppose that

1. (Causal Structure) The data follows the causal graph in Figure 38.7.

2. (Conditional Parallel Trends)  $\mathbb{E}^{A=1}[Y_1 - Y_0|X, \text{do}(A = 0)] = \mathbb{E}[Y_1 - Y_0|X, A = 0]$ .

3. (Overlap)  $P(A = 1) > 0$  and  $P(A = 1|X = x) < 1$  for all values of  $x$  in the sample space. That is, there are no covariate values that only exist in the treated group.

Then, the average treatment effect on the treated is identified as  $\text{ATT}^{\text{DiD}} = \tau^{\text{DiD}}$ , where

$$\tau^{\text{DiD}} = \mathbb{E}[\mathbb{E}[Y_1 - Y_0|A = 1, X] - \mathbb{E}[Y_1 - Y_0|A = 0, X]|A = 1]. \quad (38.94)$$

*Proof.* First, by unrolling definitions, we have that

$$\mathbb{E}^{A=1}[Y_1 - Y_0|\text{do}(A = 1), X] = \mathbb{E}[Y_1 - Y_0|A = 1, X]. \quad (38.95)$$

The interpretation is the near-tautology that the average effect among the treated under treatment is equal to the actually observed average effect among the treated. Next,

$$\mathbb{E}^{A=1}[Y_1 - Y_0|\text{do}(A = 0), X] = \mathbb{E}[Y_1 - Y_0|A = 0, X]. \quad (38.96)$$

is just the conditional parallel trends assumption. The result follows immediately.

(The overlap assumption is required to make sure all the conditional expectations are well defined).  $\square$

1 **38.6.1 Estimation**

2 With the identification result in hand, the next task is to estimate the observational estimand  
3 Equation (38.94). To that end, we define  $\tilde{Y} \triangleq Y_1 - Y_0$ . Then, we've assumed that  $\tilde{Y}, X, A \stackrel{\text{iid}}{\sim} P$  for  
4 some unknown distribution  $P$ , and our target estimand is  $\mathbb{E}[\mathbb{E}[\tilde{Y}|A=1, X] - \mathbb{E}[\tilde{Y}|A=0, X]|A=1]$ .  
5 We can immediately recognize this as the observational estimand that occurs in estimating the  
6 average treatment effect through adjustment, described in Section 38.4.5.3. That is, even though  
7 the causal situation and the identification argument are different between the adjustment setting  
8 and the difference in differences setting, the statistical estimation task we end up with is the same.  
9 Accordingly, we can use all of the estimation tools we developed for adjustment. That is, all of the  
10 techniques there—expected outcome modeling, propensity score methods, double machine learning,  
11 and so forth—were purely about the *statistical* task, which is the same between the two scenarios.  
12

13 So, we're left with the same general recipe for estimation we saw in Section 38.4.6. Namely,  
14

15 1. fit statistical or machine-learning models  $\hat{Q}(a, x)$  as a predictor for  $\tilde{Y} = Y_1 - Y_0$ , and/or  $\hat{g}(x)$  as a  
16 predictor for  $A$

17 2. compute the predictions  $\hat{Q}(0, x_i), \hat{Q}(1, x_i), \hat{g}(x_i)$  for each data point, and

18 3. combine these predictions into an estimate of the average treatment effect on the treated.

19 The estimator in the third step can be the expected outcome model estimator, the propensity weighted  
20 estimator, the double machine learning estimator, or any other strategy that's valid in the adjustment  
21 setting.

22

23 **38.7 Credibility Checks**

24

25 Once we've chosen an identification strategy, fit our models, and produced an estimate, we're faced  
26 with a basic question: should we believe it? Whether the reported estimate succeeds in capturing  
27 the true causal effect depends on whether the assumptions required for causal identification hold, the  
28 quality of the machine learning models, and the variability in the estimate due to only having access  
29 to a finite data sample. The latter two problems are already familiar from machine learning and  
30 statistical practice. We should, e.g., assess our models by checking performance on held out data,  
31 examining feature importance, and so forth. Similarly, we should report measures of the uncertainty  
32 due to finite sample (e.g., in the form of confidence intervals). Because these procedures are already  
33 familiar practice, we will not dwell on them further. However, model evaluation and uncertainty  
34 quantification are key parts of any credible causal analysis.

35 Assessing the validity of identification assumptions is trickier. First, there are assumptions that  
36 can in fact be checked from data. For example, overlap should be checked in analysis using backdoor  
37 adjustment or difference in differences, and relevance should be checked in the instrumental variable  
38 setting. Again, checking these conditions is absolutely necessary for a credible causal analysis. But,  
39 again, this involves only familiar data analysis, so we will not discuss it further. Next, there are the  
40 causal assumptions that cannot be verified from data; e.g., no unobserved confounding in backdoor  
41 adjustment, the exclusion restriction in IV, and conditional parallel trends in DiD. Ultimately, the  
42 validity of these assumptions must be assessed using substantive causal knowledge of the particular  
43 problem under consideration. However, it is possible to conduct some supplementary analyses that  
44 make the required judgement easier. We now discuss two such techniques.

45

---

### **38.7.1 Placebo Checks**

In many situations we may be able to find a variable that can be interpreted as a “treatment” that is known to have no effect on the outcome, but which we expect to be confounded with the outcome in a very similar fashion to the true treatment of interest. For example, if we’re trying to estimate the efficacy of a COVID vaccine in preventing symptomatic COVID, we might take our placebo treatment to be vaccination against HPV. We do not expect that there’s any causal effect here. However, it seems plausible that latent factors that cause an individual to seek (or avoid) HPV vaccination and COVID vaccination are similar; e.g., health conscientiousness, fear of needles, and so forth. Then, if our identification strategy is valid for the COVID vaccine, we’d also expect it to be valid for HPV vaccination. Accordingly, our estimation procedure we use for estimating the COVID effect should, when applied to HPV, yield  $\hat{\tau} \approx 0$ . Or, more precisely, the confidence interval should contain 0. If this does not happen, then we may suspect that there are still some confounding factors lurking that are not adequately handled by the identification procedure.

A similar procedure works when there is a variable that can be interpreted as an outcome which is known to not be affected by the treatment, but that shares confounders with the outcome we’re actually interested in. For example, in the COVID vaccination case, we might take the null outcome to be symptomatic COVID within 7 days of vaccination [Dag+21]. Our knowledge of both the biological mechanism of vaccination and the amount of time it takes to develop symptoms after COVID infection (at least 2 days) lead us to conclude that it’s unlikely that the treatment has a causal effect on the outcome. However, the properties of the treated people that affect how likely they are to develop symptomatic COVID are largely the same in the 7 day and, e.g., 6 month window. That includes factors such as risk aversion, baseline health, and so forth. Again, we can apply our identification strategy to estimate the causal effect of the treatment on the null outcome. If the confidence interval does not include 0, then we should doubt the credibility of the analysis.

### **38.7.2 Sensitivity Analysis to Unobserved Confounding**

We now specialize to the case of estimating the average causal effect of a binary treatment by adjusting for confounding variables, as described in Section 38.4. In this case, causal identification is based on the assumption of ‘no unobserved confounding’; i.e., the assumption that the observed covariates include all common causes of the treatment assignment and outcome. This assumption is fundamentally untestable from observed data, but its violation can induce bias in the estimation of the treatment effect—the unobserved confounding may completely or in part explain the observed association. Our aim in this part is to develop a sensitivity analysis tool to aid in reasoning about potential bias induced by unobserved confounding.

Intuitively, if we estimate a large positive effect then we might expect the real effect is also positive, even in the presence of mild unobserved confounding. For example, consider the association between smoking and lung cancer. One could argue that this association arises from a hormone that both predisposes carriers to both an increased desire to smoke and to a greater risk of lung cancer. However, the association between smoking and lung cancer is large—is it plausible that some unknown hormonal association could have a strong enough influence to explain the association? Cornfield et al. [Cor+59] showed that, for a particular observational dataset, such an unmeasured hormone would need to increase the probability of smoking by at least a factor of nine. This is an unreasonable effect size for a hormone, so they conclude it’s unlikely the causal effect can be

1  
2 explained away.

3 We would like a general procedure to allow domain experts to make judgments about whether  
4 plausible confounding is “mild” relative to the “large” effect. In particular, the domain expert must  
5 translate judgments about the strength of the unobserved confounding into judgments about the  
6 bias induced in the estimate of the effect. Accordingly, we must formalize what is meant by strength  
7 of unobserved confounding, and to show how to translate judgments about confounding strength into  
8 judgments about bias.

9 A prototypical example, due to Imbens [Imb03] (building on [RR83]), illustrates the broad approach.  
10 As above, the observed data consists of a treatment  $A$ , an outcome  $Y$ , and covariates  $X$  that may  
11 causally affect the treatment and outcome. Imbens [Imb03] then posits an additional unobserved  
12 binary confounder  $U$  for each patient, and supposes that the observed data and unobserved confounder  
13 were generated according to the following assumption, known as **Imbens’ Sensitivity Model**:

$$\begin{aligned} \text{14} \\ \text{15} \quad U_i &\stackrel{\text{iid}}{\sim} \text{Bern}(1/2) \end{aligned} \tag{38.97}$$

$$\begin{aligned} \text{16} \\ \text{17} \quad A_i | X_i, U_i &\stackrel{\text{ind}}{\sim} \text{Bern}(\text{sig}(\gamma X_i + \alpha U_i)) \end{aligned} \tag{38.98}$$

$$\begin{aligned} \text{18} \\ \text{19} \quad Y_i | X_i, A_i, U_i &\stackrel{\text{ind}}{\sim} \mathcal{N}(\tau A_i + \beta X_i + \delta U_i, \sigma^2). \end{aligned} \tag{38.99}$$

20 where  $\text{sig}$  is the sigmoid function.

21 If we had observed  $U_i$ , we could estimate  $(\hat{\tau}, \hat{\gamma}, \hat{\beta}, \hat{\alpha}, \hat{\delta}, \hat{\sigma}^2)$  from the data and report  $\hat{\tau}$  as the  
22 estimate of the average treatment effect. Since  $U_i$  is not observed, it is not possible to identify  
23 the parameters from the data. Instead, we make (subjective) judgments about plausible values of  
24  $\alpha$ —how strongly  $U_i$  affects the treatment assignment—and  $\delta$ —how strongly  $U_i$  affects the outcome.  
25 Contingent on plausible  $\alpha = \alpha^*$  and  $\delta = \delta^*$ , the other parameters can be estimated. This yields an  
26 estimate of the treatment effect  $\hat{\tau}(\alpha^*, \delta^*)$  under the presumed values of the sensitivity parameters.

27 The approach just outlined has a major drawback: it relies on a parametric model for the full data  
28 generating process. The assumed model is equivalent to assuming that, had  $U$  been observed, it  
29 would have been appropriate to use logistic regression to model treatment assignment, and linear  
30 regression to model the outcome. This assumption also implies a simple, parametric model for the  
31 relationships governing the observed data. This restriction is out of step with modern practice, where  
32 we use flexible machine-learning methods to model these relationships. For example, the assumption  
33 forbids the use of neural networks or random forests, though such methods are often state-of-the-art  
34 for causal effect estimation.

35

36 **Austen plots** We now turn to developing an alternative an adaptation of Imbens’ approach that  
37 fully decouples sensitivity analysis and modeling of the observed data. Namely, the **Austen plots**  
38 of [VZ20]. An example Austen plot is shown in Figure 38.8. The high-level idea is to posit a  
39 generative model that uses a simple, interpretable parametric form for the influence of the unobserved  
40 confounder, but that *puts no constraints on the model for the observed data*. We then use the  
41 parametric part of the model to formalize “confounding strength” and to compute the induced bias  
42 as a function of the confounding.

43 Austen plots further adapt two strategies pioneered by Imbens [Imb03]. First, we find a parameterization  
44 of the model so that the sensitivity parameters, measuring strength of confounding, are on  
45 a standardized, unitless scale. This allows us to compare the strength of hypothetical unobserved  
46 confounding to the strength of observed covariates, measured from data. Second, we plot the curve  
47

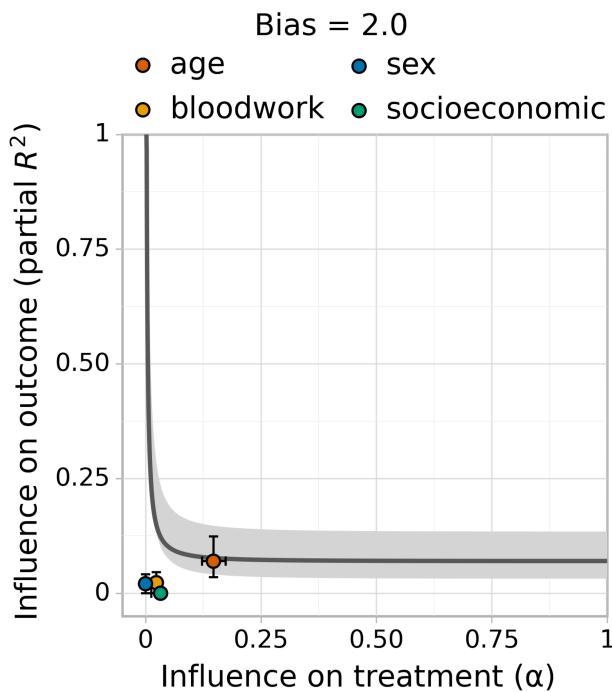


Figure 38.8: Austen plot showing how strong an unobserved confounder would need to be to induce a bias of 2 in an observational study of the effect of combination blood pressure medications on diastolic blood pressure [Dor+16]. We chose this bias to equal the nominal average treatment effect estimated from the data. We model the outcome with Bayesian Additive Regression Trees and the treatment assignment with logistic regression. The curve shows all values treatment and outcome influence that would induce a bias of 2. The colored dots show the influence strength of (groups of) observed covariates, given all other covariates. For example, an unobserved confounder with as much influence as the patient's age might induce a bias of about 2.

of all values of the sensitivity parameter that would yield given level of bias. This moves the analyst judgment from “what are plausible values of the sensitivity parameters?” to “are sensitivity parameters this extreme plausible?”

Figure 38.8, an Austen plot for an observational study of the effect of combination medications on diastolic blood pressure, illustrates the idea. A bias of 2 would suffice to undermine the qualitative conclusion that the blood-pressure treatment is effective. Examining the plot, an unobserved confounder as strong as age could induce this amount of confounding, but no other (group of) observed confounders has so much influence. Accordingly, if a domain expert thinks an unobserved confounder as strong as age is unlikely then they may conclude that the treatment is likely effective. Or, if such a confounder is plausible, they may conclude that the study fails to establish efficacy.

---

2 **Setup** The data are generated independently and identically  $(Y_i, A_i, X_i, U_i) \stackrel{\text{iid}}{\sim} P$ , where  $U_i$  is not observed and  $P$  is some unknown probability distribution. The approach in Section 38.4 assumes that the observed covariates  $X$  contain all common causes of  $Y$  and  $A$ . If this ‘no unobserved confounding’ assumption holds, then the ATE is equal to parameter,  $\tau$ , of the observed data distribution, where

$$\underline{7} \quad \tau = \mathbb{E}[\mathbb{E}[Y|X, A = 1] - \mathbb{E}[Y|X, A = 0]]. \quad (38.100)$$

9 This observational parameter is then estimated from a finite data sample. Recall from Section 38.4 that 10 this involves estimating the conditional expected outcome  $Q(A, X) = \mathbb{E}[Y|A, X]$  and the propensity 11 score  $g(X) = P(A = 1|X)$ , then plugging these into an estimator  $\hat{\tau}$ .

12 We are now concerned with the case of possible unobserved confounding. That is, where  $U$  causally 13 affects  $Y$  and  $A$ . If there is unobserved confounding then the parameter  $\tau$  is not equal to the 14 ATE, so  $\hat{\tau}$  is a biased estimate. Inference about the ATE then divides into two tasks. First, the 15 statistical task: estimating  $\tau$  as accurately as possible from the observed data. And, second, the 16 causal (domain-specific) problem of assessing  $\text{bias} = \text{ATE} - \tau$ . We emphasize that our focus here is 17 bias due to causal misidentification, not the statistical bias of the estimator. Our aim is to reason 18 about the bias induced by unobserved confounding—the second task—in a way that imposes no 19 constraints on the modeling choices for  $\hat{Q}$ ,  $\hat{g}$  and  $\hat{\tau}$  used in the statistical analysis.

21 **Sensitivity Model** Our sensitivity analysis should impose no constraints on how the *observed* 22 data is modeled. However, sensitivity analysis demands some assumption on the relationship between 23 the observed data and the *unobserved* confounder. It is convenient to formalize such assumptions 24 by specifying a probabilistic model for how the data is generated. The strength of confounding is 25 then formalized in terms of the parameters of the model (the sensitivity parameters). Then, the 26 bias induced by the confounding can be derived from the assumed model. Our task is to posit a 27 generative model that both yields a useful and easily interpretable sensitivity analysis, and that 28 avoids imposing any assumptions about the observed data.

29 To begin, consider the functional form of the sensitivity model used by Imbens [Imb03].

$$\underline{30} \quad \text{logitP}(A = 1|x, u) = h(x) + \alpha u \quad (38.101)$$

$$\underline{32} \quad \mathbb{E}[Y|a, x, u] = l(a, x) + \delta u, \quad (38.102)$$

34 for some functions  $h$  and  $l$ . That is, the propensity score is logit-linear in the unobserved confounder, 35 and the conditional expected outcome is linear.

36 By rearranging Equation (38.101) to solve for  $u$  and plugging in to Equation (38.102), we see 37 that it’s equivalent to assume  $\mathbb{E}[Y|t, x, u] = \tilde{l}(t, x) + \tilde{\delta} \text{logitP}(A = 1|x, u)$ . That is, the unobserved 38 confounder  $u$  only influences the outcome through the propensity score. Accordingly, by positing a 39 distribution on  $P(A = 1|x, u)$  directly, we can circumvent the need to explicitly articulate  $U$  (and  $h$ ).

40 **Definition 38.7.1.** Let  $\tilde{g}(x, u) = P(A = 1|x, u)$  denote the propensity score given observed covariates 41  $x$  and the unobserved confounder  $u$ .

43 The insight is that we can posit a sensitivity model by defining a distribution on  $\tilde{g}$  directly. We 44 choose:

$$\underline{46} \quad \tilde{g}(X, U)|X \sim \text{Beta}(g(X)(1/\alpha - 1), (1 - g(X))(1/\alpha - 1)).$$

That is, the full propensity score  $\tilde{g}(X, U)$  for each unit is assumed to be sampled from a Beta distribution centered at the observed propensity score  $g(X)$ . The sensitivity parameter  $\alpha$  plays the same role as in Imbens' model: it controls the influence of the unobserved confounder  $U$  on treatment assignment. When  $\alpha$  is close to 0 then  $\tilde{g}(X, U)|X$  is tightly concentrated around  $g(X)$ , and the unobserved confounder has little influence. That is,  $U$  minimally affects our belief about who is likely to receive treatment. Conversely, when  $\alpha$  is close to 1 then  $\tilde{g}$  concentrates near 0 and 1; i.e., knowing  $U$  would let us accurately predict treatment assignment. Indeed, it can be shown that  $\alpha$  is the change in our belief about how likely a unit was to have gotten the treatment, given that they were actually observed to be treated (or not):

$$\alpha = \mathbb{E}[\tilde{g}(X, U)|A = 1] - \mathbb{E}[\tilde{g}(X, U)|A = 0]. \quad (38.103)$$

With the  $\tilde{g}$  model in hand, we define the **Austen Sensitivity Model** as follows:

$$\tilde{g}(X, U)|X \sim \text{Beta}(g(X)(1/\alpha - 1), (1 - g(X))(1/\alpha - 1)) \quad (38.104)$$

$$A|X, U \sim \text{Bern}(\tilde{g}(X, U)) \quad (38.105)$$

$$\mathbb{E}[Y|A, X, U] = Q(A, X) + \delta(\text{logit}\tilde{g}(X, U) - \mathbb{E}[\text{logit}\tilde{g}(X, U)|A, X]). \quad (38.106)$$

This model has been constructed to satisfy the requirement that the propensity score and conditional expected outcome are the  $g$  and  $Q$  actually present in the observed data:

$$\mathbb{P}(A = 1|X) = \mathbb{E}[\mathbb{E}[T|X, U]|X] = \mathbb{E}[\tilde{g}(X, U)|X] = g(X)$$

$$\mathbb{E}[Y|A, X] = \mathbb{E}[\mathbb{E}[Y|A, X, U]|A, X] = Q(A, X).$$

The sensitivity parameters are  $\alpha$ , controlling the dependence between the unobserved confounder the treatment assignment, and  $\delta$ , controlling the relationship with the outcome.

**Bias** We now turn to calculating the bias induced by unobserved confounding. By assumption,  $X$  and  $U$  together suffice to render the average treatment effect identifiable as:

$$\text{ATE} = \mathbb{E}[\mathbb{E}[Y|A = 1, X, U] - \mathbb{E}[Y|A = 0, X, U]].$$

Plugging in our sensitivity model yields,

$$\text{ATE} = \mathbb{E}[Q(1, X) - Q(0, X)] + \delta(\mathbb{E}[\text{logit}\tilde{g}(X, U)|X, A = 1] - \mathbb{E}[\text{logit}\tilde{g}(X, U)|X, A = 0]).$$

The first term is the observed-data estimate  $\tau$ , so

$$\text{bias} = \delta(\mathbb{E}[\text{logit}\tilde{g}(X, U)|X, A = 1] - \mathbb{E}[\text{logit}\tilde{g}(X, U)|X, A = 0]).$$

Then, by invoking Beta-Bernoulli conjugacy and standard Beta identities,<sup>10</sup> we arrive at,

**Theorem 6.** *Under the Austen sensitivity model, Equation (38.106), an unobserved confounder with influence  $\alpha$  and  $\delta$  induces bias in the estimated treatment effect equal to*

$$\text{bias} = \frac{\delta}{1/\alpha - 1} \mathbb{E}\left[\frac{1}{g(X)} + \frac{1}{1 - g(X)}\right].$$

<sup>10</sup> We also use the recurrence relation  $\psi(x+1) - \psi(x) = 1/x$ , where  $\psi$  is the digamma function.

1 That is, the amount of bias is determined by the sensitivity parameters and by the *realized*  
 2 propensity score. Notice that more extreme propensity scores lead to more extreme bias in response  
 3 to unobserved confounding. This means, in particular, that conditioning on a covariate that affects  
 4 the treatment but that does not directly affect the outcome (an instrument) will increase any bias  
 5 due to unobserved confounding. This general phenomena is known as **z-bias**.  
 6

7

8 **Sensitivity Parameters** The Austen model provides a formalization of confounding strength  
 9 in terms of the parameters  $\alpha$  and  $\delta$  and tells us how much bias is induced by a given strength of  
 10 confounding. This lets us translate judgments about confounding strength to judgments about bias.  
 11 However, it is not immediately obvious how to translate qualitative judgements such as “I think any  
 12 unobserved confounder would be much less important than Age” to judgements about the possible  
 13 values of the sensitivity parameters.

14 First, because the scale of  $\delta$  is not fixed, it may be difficult to compare the influence of potential  
 15 unobserved confounders to the influence of reference variables. To resolve this, we reexpress the  
 16 outcome-confounder strength in terms of the (non-parametric) partial coefficient of determination:

$$17 \quad R_{Y,\text{par}}^2(\alpha, \delta) = 1 - \frac{\mathbb{E}(Y - \mathbb{E}[Y|A, X, U])^2}{\mathbb{E}(Y - Q(A, X))^2}.$$

20 The key to computing the reparameterization is the following result

21 **Theorem 7.** Under the Austen sensitivity model, Equation (38.106), the outcome influence is  
 22

$$23 \quad R_{Y,\text{par}}^2(\alpha, \delta) = \delta^2 \sum_{a=0}^1 \frac{\mathbb{E}[\psi_1(g(X)^a(1-g(X))^{1-a}(1/\alpha - 1) + 1[A=a])]}{\mathbb{E}[(Y - Q(A, X))^2]},$$

26 where  $\psi_1$  is the trigamma function.

27 See Veitch and Zaveri [VZ20] for the proof.

28 By design,  $\alpha$ —the strength of confounding influence on on treatment assignment—is already on  
 29 a fixed, unitless scale. However, because the measure is tied to the model it may be difficult to  
 30 interpret, and it is not obvious how to compute reference confounding strength values from the  
 31 observed data. The next result clarifies these issues.

32

33 **Theorem 8.** Under the Austen sensitivity model, Equation (38.106),

$$34 \quad \alpha = 1 - \frac{\mathbb{E}[\tilde{g}(X, U)(1 - \tilde{g}(X, U))]}{\mathbb{E}[g(X)(1 - g(X))]}.$$

37 See Veitch and Zaveri [VZ20] for the proof. That is, the sensitivity parameter  $\alpha$  is measures how  
 38 much more extreme the propensity scores become when we condition on  $U$ . That is,  $\alpha$  is a measure  
 39 of the extra predictive power  $U$  adds for  $A$ , above and beyond the predictive power in  $X$ . It may  
 40 also be insightful to notice that

$$41 \quad \alpha = R_{A,\text{par}}^2 = 1 - \frac{\mathbb{E}[(A - \tilde{g}(X, U))^2]}{\mathbb{E}[(A - g(X))^2]}. \quad (38.107)$$

44 That is,  $\alpha$  is just the (non-parametric) partial coefficient of determination of  $U$  on  $A$ —the same  
 45 measure used for the outcome influence. (To see this, just expand the expectations conditional on  
 46  $A = 1$  and  $A = 0$ ).

47

**Estimating bias** In combination, Theorems 6 and 7 yield an expression for the bias in terms of  $\alpha$  and  $R_{Y,\text{par}}^2$ . In practice, we can estimate the bias induced by confounding by fitting models for  $\hat{Q}$  and  $\hat{g}$  and replacing the expectations by means over the data.

### 38.7.2.1 Calibration using observed data

The analyst must make judgments about the influence a hypothetical unobserved confounder might have on treatment assignment and outcome. To calibrate such judgments, we'd like to have a reference point for how much the observed covariates influence the treatment assignment and outcome. In the sensitivity model, the degree of influence is measured by partial  $R_Y^2$  and  $\alpha$ . We want to measure the degree of influence of an observed covariate  $Z$  given the other observed covariates  $X \setminus Z$ .

For the outcome, this can be measured as:

$$R_{Y \cdot Z|T, X \setminus Z}^2 \triangleq 1 - \frac{\mathbb{E}(Y - Q(A, X))^2}{\mathbb{E}(Y - \mathbb{E}[Y|A, X \setminus Z])^2}.$$

In practice, we can estimate the quantity by fitting a new regression model  $\hat{Q}_Z$  that predicts  $Y$  from  $A$  and  $X \setminus Z$ . Then we compute

$$R_{Y \cdot Z|T, X \setminus Z}^2 = 1 - \frac{\frac{1}{n} \sum_i (y_i - \hat{Q}(t_i, x_i))^2}{\frac{1}{n} \sum_i (y_i - \hat{Q}_Z(t_i, x_i \setminus z_i))^2}.$$

Using Theorem 8, we can measure influence of observed covariate  $Z$  on treatment assignment given  $X \setminus Z$  in an analogous fashion to the outcome. We define  $g_{X \setminus Z}(X \setminus Z) = P(A = 1|X \setminus Z)$ , then fit a model for  $g_{X \setminus Z}$  by predicting  $A$  from  $X \setminus Z$ , and estimate

$$\hat{\alpha}_{Z|X \setminus Z} = 1 - \frac{\frac{1}{n} \sum_i \hat{g}(x_i)(1 - \hat{g}(x_i))}{\frac{1}{n} \sum_i \hat{g}_{X \setminus Z}(x_i \setminus z_i)(1 - \hat{g}_{X \setminus Z}(x_i \setminus z_i))}.$$

**Grouping covariates** The estimated values  $\hat{\alpha}_{X \setminus Z}$  and  $R_{Y \cdot X \setminus Z}^2$  measure the influence of  $Z$  conditioned on all the other confounders. In some cases, this can be misleading. For example, if some piece of information is important but there are multiple covariates providing redundant measurements, then the estimated influence of each covariate will be small. To avoid this, group together related or strongly dependent covariates and compute the influence of the entire group in aggregate. For example, grouping income, location, and race as ‘socioeconomic variables’.

### 38.7.2.2 Practical Use

We now have sufficient results to produce Austen plots such as Figure 38.8. At a high level, the procedure is:

1. Produce an estimate  $\hat{\tau}$  using any modeling tools. As a component of this, estimate the propensity score  $\hat{g}$  and conditional outcome model  $\hat{Q}$
2. Pick a level of bias that would suffice to change the qualitative interpretation of the estimate (e.g., the lower bound of a 95% confidence interval).

- <sup>1</sup>
- <sup>2</sup> 3. Plot the values of  $\alpha$  and  $R_{Y,\text{par}}^2$  that would suffice to induce that much bias. This is the black  
<sup>3</sup> curve on the plot. To calculate these values, use Theorems 6 and 7 together with the estimated  $\hat{g}$   
<sup>4</sup> and  $\hat{Q}$ .
- <sup>5</sup>
- <sup>6</sup> 4. Finally, compute reference influence level for (groups of) observed covariates. In particular, this  
<sup>7</sup> requires fitting reduced models for the conditional expected outcome and propensity that do not  
<sup>8</sup> use the reference covariate as a feature.

<sup>9</sup> In practice, an analyst only needs to do the model fitting parts themselves. The bias calculations,  
<sup>10</sup> reference value calculations, and plotting can be done automatically with standard libraries.<sup>11</sup>

<sup>11</sup> Austen plots are predicated on Equation (38.106). This assumption replaces the purely parametric  
<sup>12</sup> Equation (38.99) with a version that eliminates any parametric requirements on the observed data.  
<sup>13</sup> However, we emphasize that Equation (38.106) does, implicitly, impose some parametric assumption  
<sup>14</sup> on the structural causal relationship between  $U$  and  $A, Y$ . Ultimately, any conclusion drawn from  
<sup>15</sup> the sensitivity analysis depends on this assumption, which is not justified on any substantive grounds.  
<sup>16</sup> Accordingly, such sensitivity analyses can only be used to informally guide domain experts. They  
<sup>17</sup> do not circumvent the need to thoroughly adjust for confounding. This reliance on a structural  
<sup>18</sup> assumption is a generic property of sensitivity analysis.<sup>12</sup> Indeed, there are now many sensitivity  
<sup>19</sup> analysis models that allow the use of any machine learning model in the data analysis [e.g., RRS00;  
<sup>20</sup> FDF19; She+11; HS13; BK19; Ros10; Yad+18; ZSB19; Sch+21a]. However, none of these are yet in  
<sup>21</sup> routine use in practice. We have presented Austen plots here not because they make an especially  
<sup>22</sup> virtuous modeling assumption, but because they are (relatively) easy to understand and interpret.  
<sup>23</sup> Austen plots are most useful in situations where the conclusion from the plot would be ‘obvious’  
<sup>24</sup> to a domain expert. For instance, in Figure 38.8, we can be confident that an unobserved confounder  
<sup>25</sup> similar to socioeconomic status would not induce enough bias to change the qualitative conclusion.  
<sup>26</sup> By contrast, Austen plots should not be used to draw conclusions such as, “I think a latent confounder  
<sup>27</sup> could only be 90% as strong as ‘age’, so there is evidence of a small non-zero effect”. Such nuanced  
<sup>28</sup> conclusions might depend on issues such as the particular sensitivity model we use, or finite-sample  
<sup>29</sup> variation of our bias and influence estimates, or on incautious interpretation of the calibration dots.  
<sup>30</sup> These issues are subtle, and it would be difficult resolve them to a sufficient degree that a sensitivity  
<sup>31</sup> analysis would make an analysis credible.

<sup>32</sup>

<sup>33</sup> **Calibration using observed data** The interpretation of the observed-data calibration requires  
<sup>34</sup> some care. The sensitivity analysis requires the analyst to make judgements about the strength  
<sup>35</sup> of influence of the unobserved confounder  $U$ , *conditional on the observed covariates  $X$* . However,  
<sup>36</sup> we report the strength of influence of observed covariate(s)  $Z$ , *conditional on the other observed*  
<sup>37</sup> *covariates  $X \setminus Z$* . The difference in conditioning sets can have subtle effects.

<sup>38</sup> Cinelli and Hazlett [CH20] give an example where  $Z$  and  $U$  are identical variables in the true  
<sup>39</sup> model, but where influence of  $U$  given  $A, X$  is larger than the influence of  $Z$  given  $A, X \setminus Z$ . (The  
<sup>40</sup> influence of  $Z$  given  $X \setminus Z, U$  would be the same as the influence of  $U$  given  $X$ ). Accordingly, an  
<sup>41</sup> analyst is *not* justified in a judgement such as, “I know that  $U$  and  $Z$  are very similar. I see  $Z$  has  
<sup>42</sup> substantial influence, but the dot is below the line. Thus,  $U$  will not undo the study conclusions.” In  
<sup>43</sup>

<sup>44</sup> 11. See [github.com/vveitch/causality-tutorials/blob/main/Sensitivity\\_Analysis.ipynb](https://github.com/vveitch/causality-tutorials/blob/main/Sensitivity_Analysis.ipynb)

<sup>45</sup> 12. In extreme cases, there can be so little unexplained variation in  $A$  or  $Y$  that only a very weak confounder could be  
<sup>46</sup> compatible with the data. In this case, essentially assumption free sensitivity analysis is possible [Man90].

<sup>47</sup>

essence, if the domain expert suspects a strong interaction between  $U$  and  $Z$  then naively eyeballing the dot-vs-line position may be misleading. A particular subtle case is when  $U$  and  $Z$  are independent variables that both strongly influence  $A$  and  $Y$ . The joint influence on  $A$  creates an interaction effect between them when  $A$  is conditioned on (the treatment is a collider). This affects the interpretation of  $R^2_{Y \cdot U|X,A}$ . Indeed, we should generally be skeptical of sensitivity analysis interpretation when it is expected that a strong confounder has been omitted. In such cases, our conclusions may depend substantively on the particular form of our sensitivity model, or other unjustifiable assumptions.

Although the interaction problem is conceptually important, its practical significance is unclear. We often expect the opposite effect: if  $U$  and  $Z$  are dependent (e.g., race and wealth) then omitting  $U$  should increase the apparent importance of  $Z$ —leading to a conservative judgement (a dot artificially towards the top right part of the plot).

## 38.8 The Do Calculus

We have seen several strategies for identifying causal effects as parameters of observational distributions. Confounder adjustment (Section 38.4) relied only on the assumed causal graph (and overlap), which specified that we observe all common causes of  $A$  and  $Y$ . On the other hand, instrumental variable methods and difference-in-differences each relied on both an assumed causal graph and partial functional form assumptions about the underlying structural causal model. Because functional form assumptions can be quite difficult to justify on substantive grounds, it's natural to ask when causal identification is possible from the causal graph alone. That is, when can we be agnostic to the particular functional form of the structural causal models?

There is a general “calculus of intervention”, known as the **do-calculus**, that gives a general recipe for determining when the causal assumptions expressed in a causal graph can be used to identify causal effects [Pea09c]. The do-calculus is a set of three rewrite rules that allows us to replace statements where we condition on variables being set by intervention, e.g.  $P(Y|\text{do}(A = a))$ , with statements involving only observational quantities, e.g.  $\mathbb{E}_X[P(Y|A = a, X)]$ . When causal identification is possible, we can repeatedly apply the three rules to boil down our target causal parameter into an expression involving only the observational distribution.

### 38.8.1 The three rules

To express the rules, let  $X$ ,  $Y$ ,  $Z$ , and  $W$  be arbitrary disjoint sets of variables in a causal DAG  $G$ .

**Rule 1** The first rule allows us to insert or delete observations  $z$ :

$$p(y|\text{do}(x), z, w) = p(y|\text{do}(x), w) \text{ if } (Y \perp Z|X, W)_{G_{\overline{X}}} \quad (38.108)$$

where  $G_{\overline{X}}$  denotes cutting edges going into  $X$ , and  $(Y \perp Z|X, W)_{G_{\overline{X}}}$  denotes conditional independence in the mutilated graph. The rule follows from d-separation in the mutilated graph. This rule just says that conditioning on irrelevant variables leaves the distribution invariant (as we would expect).

**Rule 2** The second rule allows us to replace  $\text{do}(z)$  with conditioning on (seeing)  $z$ . The simplest case where we can do this is: if  $Z$  is a root of the causal graph (i.e., it has no causal parents) then  $p(y|\text{do}(z)) = p(y|z)$ . The reason is that the do operator is equivalent to conditioning in the mutilated

1 causal graph where all the edges into  $Z$  are removed, but, because  $Z$  is a root, the mutilated graph  
2 is just the original causal graph. The general form of this rule is:  
3

$$\frac{4}{5} \quad p(y|\text{do}(x), \text{do}(z), w) = p(y|\text{do}(x), z, w) \text{ if } (Y \perp Z|X, W)_{G_{\overline{XZ}}} \quad (38.109)$$

6 where  $G_{\overline{XZ}}$  cuts edges going into  $X$  and out of  $Z$ . Intuitively, we can replace  $\text{do}(z)$  by  $z$  as long as  
7 there are no backdoor (non-directed) paths between  $z$  and  $y$ . If there are in fact no such paths, then  
8 cutting all the edges going out of  $Z$  will mean there are no paths connecting  $Z$  and  $Y$ , so that  $Y \perp Z$ .  
9 The rule just generalizes this line of reasoning to allow for extra observed and intervened variables.  
10

11  
12 **Rule 3** The third rule allows us to insert or delete actions  $\text{do}(z)$ :

$$\frac{13}{14} \quad p(y|\text{do}(x), \text{do}(z), w) = p(y|\text{do}(x), w) \text{ if } (Y \perp Z|X, W)_{G_{\overline{XZ^*}}} \quad (38.110)$$

15 where  $G_{\overline{XZ^*}}$  cuts edges going into  $X$  and  $Z^*$ , and where  $Z^*$  is the set of  $Z$ -nodes that are not  
16 ancestors of any  $W$ -node in  $G_{\overline{X}}$ . Intuitively, this condition corresponds to intervening on  $X$ , and  
17 checking whether the distribution of  $Y$  is invariant to *any* intervention that we could apply on  $Z$ .  
18

### 19 38.8.2 Revisiting Backdoor Adjustment

21 We begin with a more general form of the adjustment formula we used in Section 38.4.

22 First, suppose we observe all of  $A$ 's parents, call them  $X$ . For notational simplicity, we'll assume  
23 for the moment that  $X$  is discrete. Then,  
24

$$\frac{25}{26} \quad p(Y = y|\text{do}(A = a)) = \sum_x p(Y = y|x, \text{do}(A = a))p(x|\text{do}(A = a)) \quad (38.111)$$

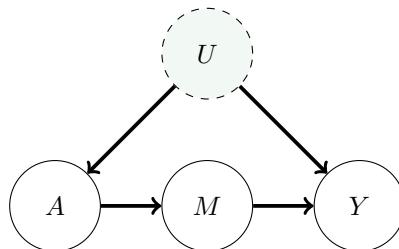
$$\frac{27}{28} \quad = \sum_x p(Y = y|x, A = a)p(x). \quad (38.112)$$

30 The first line is just a standard probability relation (marginalizing over  $z$ ). We are using causal  
31 assumptions in two ways in the second line. First,  $p(x|\text{do}(A = a)) = p(x)$ : the treatment has  
32 no causal effect on  $Z$ , so interventions on  $A$  don't change the distribution of  $Z$ . This is rule 3,  
33 Equation (38.110). Second,  $p(Y = y|z, \text{do}(A = a)) = p(Y = y|z, A = a)$ . This equality holds because  
34 conditioning on the parents blocks all non-directed paths from  $A$  to  $Y$ , reducing the causal effect to  
35 be the same as the observational effect. The equality is an application of rule 2, Equation (38.109).

36 Now, what if we don't observe all the parents of  $A$ ? The key issue is **backdoor paths**: paths  
37 between  $A$  and  $Y$  that contain an arrow into  $A$ . These paths are the general form of the problem  
38 that occurs when  $A$  and  $Y$  share a common cause. Suppose that we can find a set of variables  $S$   
39 such that (1) no node in  $S$  is a descendant of  $A$ ; and (2)  $S$  blocks every backdoor path between  $A$   
40 and  $Y$ . Such a set is said to satisfy the **backdoor criterion**. In this case, we can use  $S$  instead of  
41 the parents of  $X$  in the adjustment formula, Equation (38.112). That is,  
42

$$\frac{43}{44} \quad p(Y = y|\text{do}(A = a)) = \mathbb{E}_S[p(Y = y|S, A = a)]. \quad (38.113)$$

45 The proof follows the invocation of rules 3 and 2, in the same way as for the case where  $S$  is just the  
46 parents of  $A$ . Notice that requiring  $S$  to not contain any descendants of  $A$  means that we don't risk  
47



*Figure 38.9: Causal graph illustrating the frontdoor criterion setup. The effect of the treatment  $A$  on outcome  $Y$  is entirely mediated by mediator  $M$ . This allows us infer the causal effect even if the treatment and outcome are confounded by  $U$ .*

conditioning on any variables that mediate the effect, nor any variables that might be colliders—either would undermine the estimate.

The backdoor adjustment formula generalizes the adjust-for-parents approach and adjust-for-all-common-causes approach of Section 38.4. That's because both the parents of  $A$  and the common causes satisfy the backdoor criterion.

In practice, the full distribution  $p(Y = y|do(A = a))$  is rarely used as the causal target. Instead, we try to estimate a low-dimensional parameter of this distribution, such as the average treatment effect. The adjustment formula immediately translates in the obvious way. If we define

$$\tau = \mathbb{E}_S[\mathbb{E}[Y|A = 1, S] - \mathbb{E}[Y|A = 0, S]],$$

then we have that  $ATE = \tau$  whenever  $S$  satisfies the backdoor criteria. The parameter  $\tau$  can then be estimated from finite data using the methods described in Section 38.4, using  $S$  in place of the common causes  $X$ .

### 38.8.3 Frontdoor Adjustment

Backdoor adjustment is applicable if there's at least one observed variable on every backdoor path between  $A$  and  $Y$ . As we have seen, identification is sometimes still possible even when this condition doesn't hold. Frontdoor adjustment is another strategy of this kind. Figure 38.9 shows the causal structure that allows this kind of adjustment strategy. Suppose we're interested in the effect of smoking  $A$  on developing cancer  $Y$ , but we're concerned about some latent genetic confounder  $U$ .

Suppose that all of the directed paths from  $A$  to  $Y$  pass through some set of variables  $M$ . Such variables are called **mediators**. For example, the effect of smoking on lung cancer might be entirely mediated by the amount of tar in the lungs and measured tissue damage. It turns out that if all such mediators are observed, and the mediators do not have an unobserved common cause with  $A$  or  $Y$ , then causal identification is possible. To understand why this is true, first notice that we can identify the causal effect of  $A$  on  $M$  and the causal effect of  $M$  on  $A$ , both by backdoor adjustment. Further, the mechanism of action of  $A$  on  $Y$  is:  $A$  changes  $M$  which in turn changes  $Y$ . Then, we

1  
2 can combine these as:

3

$$\frac{4}{5} \quad p(Y|\text{do}(A = a)) = \sum_m p(Y|\text{do}(M = m))p(M = m|\text{do}(A = a)) \quad (38.114)$$

6

$$\frac{7}{8} \quad = \sum_m \sum_{a'} p(Y|a', m)p(a')p(m|a) \quad (38.115)$$

9 The second line is just backdoor adjustment applied to identify each of the do expressions (note that  
10  $A$  blocks the  $M$ - $Y$  backdoor path through  $U$ ).

11 Equation (38.115) is called the **front-door formula** [Pea09b, §3.3.2]. To state the result in more  
12 general terms, let us introduce a definition. We say a set of variables  $M$  satisfies the **front-door**  
13 **criterion** relative to an ordered pair of variables  $(A, Y)$  if (1)  $M$  intercepts all directed paths from  
14  $A$  to  $Y$ ; (2) there is no unblocked backdoor path from  $A$  to  $M$ ; and (3) all backdoor paths from  $M$   
15 to  $Y$  are blocked by  $A$ . If  $M$  satisfies this criterion, and if  $p(A, M) > 0$  for all values of  $A$  and  $M$ ,  
16 then the causal effect of  $A$  on  $Y$  is identifiable and is given by Equation (38.115).

17 Let us interpret this theorem in terms of our smoking example. Condition 1 means that smoking  
18  $A$  should have no effect on cancer  $Y$  except via tar and tissue damage  $M$ . Conditions 2 and 3 mean  
19 that the genotype  $U$  cannot have any effect on  $M$  except via smoking  $A$ . Finally, the requirement  
20 that  $p(A, M) > 0$  for all values implies that high levels of tar in the lungs must arise not only due to  
21 smoking, but also other factors (e.g., pollutants). In other words, we require  $p(A = 0, M = 1) > 0$  so  
22 we can assess the impact of the mediator in the untreated setting.

23 We can now use the do-calculus to derive the frontdoor criterion; following [PM18b, p236]. Assuming  
24 the causal graph  $G$  shown in Figure 38.9:

25

$$\frac{26}{27} \quad p(y|\text{do}(a)) = \sum_m p(y|\text{do}(a), m)p(m|\text{do}(a)) \quad (\text{probability axioms})$$

28

$$\frac{29}{30} \quad = \sum_m p(y|\text{do}(a), \text{do}(m))p(m|\text{do}(a)) \quad (\text{rule 2 using } G_{\overline{S}\underline{T}})$$

31

$$\frac{32}{33} \quad = \sum_m p(y|\text{do}(a), \text{do}(m))p(m|a) \quad (\text{rule 2 using } G_{\underline{S}})$$

34

$$\frac{35}{36} \quad = \sum_m p(y|\text{do}(m))p(m|a) \quad (\text{rule 3 using } G_{\overline{S}\overline{T}^*})$$

37

$$\frac{38}{39} \quad = \sum_{a'} \sum_m p(y|\text{do}(m), a')p(a'|\text{do}(m))p(m|a) \quad (\text{probability axioms})$$

40

$$\frac{41}{42} \quad = \sum_{a'} \sum_m p(y|m, a')p(a')p(m|a) \quad (\text{rule 2 using } G_{\underline{T}})$$

43

44 **Estimation** To estimate the causal distribution from data using the frontdoor criterion we need to  
45 estimate each of  $p(y|m, a)$ ,  $p(a)$ , and  $p(m|a)$ . In practice, we can fit models  $\hat{p}(y|m, a)$  by predicting  
46  $Y$  from  $M$  and  $A$ , and  $\hat{p}(m|a)$  by predicting  $M$  from  $A$ . Then, using the empirical distribution to  
47

1 estimate  $p(a)$ , the final estimate is:

$$\frac{1}{|A|} \sum_{a'} \sum_m \hat{p}(y|m, a') \hat{p}(m|a), \quad (38.116)$$

6 where  $|A|$  is the number of treatments.

8 We usually have more modest targets than the full distribution  $p(y|\text{do}(a))$ . For instance, we may  
9 be content with just estimating the average treatment effect. It's straightforward to derive a formula  
10 for this using the frontdoor adjustment. Similarly to backdoor adjustment, more advanced estimators  
11 of the ATE through frontdoor effect are possible in principle. For example, we might combine fitted  
12 models for  $\mathbb{E}[Y|m, a]$  and  $P(M|a)$ . See Fulcher et al. [Ful+20] for an approach to robust estimation  
13 via front door adjustment, as well as a generalization of the front door approach to more general  
14 settings.

## 16 38.9 Further Reading

18 There is an enormous and growing literature on the intersection of causality and machine learning.

19 First, there are many textbooks on theoretical and practical elements of causal inference. These  
20 include Pearl [Pea09c], focused on causal graphs, Angrist and Pischke [AP08], focused on econometrics,  
21 Hernán and Robins [HR20b], with roots in epidemiology, Imbens and Rubin [IR15], with origin in  
22 statistics, and Morgan and Winship [MW15], for a social sciences perspective. The introduction to  
23 causality in Shalizi [Sha22, §7] is also recommended, particularly the treatment of matching.

24 Double machine-learning has featured prominently in this chapter. This is a particular instantiation  
25 of non-parametric estimation. This topic has substantial theoretical and practical importance in  
26 modern causal inference. The double machine learning work includes estimators for many commonly  
27 encountered scenarios [Che+17e; Che+17d]. Good references for a lucid explanation of how and  
28 why non-parametric estimation works include [Ken16; Ken17; FK21]. Usually, the key guarantees of  
29 non-parametric estimator are asymptotic. Generally, there are many estimators that share optimal  
30 asymptotic guarantees (e.g. the AIPTW estimator given in Equation (38.31)). Although these are  
31 asymptotically equivalent, in finite samples their behavior can be very different. There are estimators  
32 that preserve asymptotic guarantees but aim to improve performance in practical finite sample  
33 regimes [e.g., vR11].

34 There is also considerable interest in the estimation of heterogeneous treatment effects. The  
35 question here is: what effect would this treatment have when applied to a unit with such-and-such  
36 specific characteristics? E.g., what is the effect of this drug on women over the age of 50? The causal  
37 identification arguments used here are more-or-less the same as for the estimation of average case  
38 effects. However, the estimation problems can be substantially more involved. Some reading includes  
39 [Kün+19; NW20; Ken20; Yad+21].

40 There are several commonly applicable causal identification and estimation strategies beyond the  
41 ones we've covered in this chapter. **Regression discontinuity designs** rely on the presence of  
42 some sharp, arbitrary non-linearity in treatment assignment. For example, eligibility for some aid  
43 programs is determined by whether an individual has income below or above a fixed amount. The  
44 effect of the treatment can be studied by comparing units just below and just above this threshold.  
45 **Synthetic controls** are a class of methods that try to study the effect of a treatment on a given  
46 unit by constructing a synthetic version of that unit that acts as a control. For example, to study the  
47

1 effect of legislation banning smoking indoors in California, we can construct a synthetic California  
2 as a weighted average of other states, with weights chosen to balance demographic characteristics.  
3 Then, we can compare the observed outcome of California with the outcome of the synthetic control,  
4 constructed as the weighted average of the outcomes of the donor states. See Angrist and Pischke  
5 [AP08] for a textbook treatment of both strategies. Closely related are methods that use time series  
6 modeling to create synthetic outcomes. For example, to study the effect of an advertising campaign  
7 beginning at time  $T$  on product sales  $Y_t$ , we might build a time series model for  $Y_t$  using data in the  
8  $t < T$  period, and then use this model to predict the values of  $(\hat{Y}_t)_{t>T}$  we would have seen had the  
9 campaign not been run. We can estimate the causal effect by comparing the factual, realized  $Y_t$  to  
10 the predicted, counterfactual,  $\hat{Y}_t$ . See Brodersen et al. [Bro+15] for an instantiation of this idea.

11 In this chapter, our focus has been on using machine learning tools to estimate causal effects.  
12 There is also a growing interest in using the ideas of causality to improve machine learning tools.  
13 This is mainly aimed at building predictors that are robust against when deployed in new domains  
14 [SS18c; SCS19; Arj+20; Mei18b; PBM16a; RC+18; Zha+13a; Sch+12b; Vei+21] or that do not rely  
15 on particular ‘spurious’ correlations in the training data [RPH21; Wu+21; Gar+19; Mit+20; WZ19;  
16 KCC20; KHL20; TAH20; Vei+21].

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

# Index

- Q*-function, 1123  
*x*-divergence VI, 443  
*x*-divergence upper bound, 444  
*f*-MAX, 1165  
(fixed interval) smoothing, 320  
#P-hard, 389  
*Global explanation*, 1078  
*Local explanation*, 1077  
*intrinsic*, 1084  
“arms”, 1106  
“do” operator, 179  
1/f, 14  
2d lattice model, 148  
3-SAT, 389  
80-20 rule, 15
- Sylvester determinant lemma, 683
- A\* search, 1043, 1148  
A/B test, 1106  
A2C, 1142  
A3C, 1142  
ABC, 547, 892  
abduction, 183  
ABOBA, 512  
absorbing state, 52, 1122  
abstain, 751  
accept, 468  
acquisition function, 265, 267  
acquisition shift, 742  
action, 1111  
action nodes, 1103  
action-value function, 1123  
Actionability, 1082  
actions, 120, 997  
activation function, 604  
active inference, 1163  
active learning, 264, 269, 682  
actor critic, 1131  
actor-critic, 1141  
acyclic directed mixed graph, 167  
AD, 221  
Adam, 263  
adaptive importance sampling, 831  
adaptive MCMC, 472  
adaptive policies, 1111  
adaptive prediction sets, 567  
adaptive proposal distributions, 535  
adaptive rejection Metropolis sampling, 481  
adaptive rejection sampling, 458  
adaptive resampling, 526  
adaptive tempering, 543  
add-one smoothing, 50, 72  
add-one-in, 569  
additive Gaussian noise, 997  
additive intervention, 180  
additive scalar bijection, 848  
additive unobserved confounding, 1193  
ADF, 360  
adjacency matrix, 618  
adjacency list, 618  
adjoint sensitivity method, 860
- ADMG, 167  
admissible, 1148  
admixture mixture, 958  
admixture model, 24, 956  
advantage actor critic, 1142  
advantage function, 1123  
adversarial stickers, 770  
adversarial attack, 769  
adversarial autoencoder, 806  
adversarial bandit, 1111  
adversarial image, 769  
adversarial inverse RL, 1165  
adversarial risk, 773  
adversarial training, 772  
ADVI, 310, 426  
AEP, 200  
affine autoregressive flows, 852  
affine flow, 847  
affine scalar bijection, 848  
affine transformation, 847  
affinity propagation, 383  
agent, 121, 1111, 1120  
aggregate, 620  
aggregate posterior, 211  
aggregated posterior, 217, 800  
AI, 210  
AIPTW, 1183  
AIRL, 1165  
Akaike information criterion, 117  
aleatoric uncertainty, 67, 562  
ALI, 912  
alpha divergence, 55  
alpha posterior, 629  
AlphaGo, 1148  
AlphaZero, 1148  
alternative hypothesis, 112  
amortization gap, 430  
amortized ELBO, 430  
amortized inference, 385, 429, 535, 791, 792, 832  
analysis by synthesis, 925  
ancestor, 524  
ancestral graph, 157  
ancestral sampling, 136  
annealed importance sampling, 460, 543, 785, 863  
annotation shift, 743  
annulus, 200  
anomaly detection, 731, 749, 951  
anti-causal prediction, 741  
anti-Hebbian term, 867  
antiferromagnetic, 149  
antithetic sampling, 464  
aperiodic, 53  
apprenticeship learning, 1163  
Approximate Bayesian Computation, 547, 892  
approximate dynamic programming, 1125  
approximate inference, 306  
approximate linear programming, 1125  
AR, 777  
architectural methods, 767  
ARD, 585, 663
- ARD kernel, 663  
arithmetic circuits, 172  
arithmetic coding, 210  
ARMA, 725  
arrow of time, 735  
artificial process noise, 531  
ASOS, 1005  
ASR, 331  
associative Markov model, 151  
associative Markov network, 149  
assumed density filtering, 360, 364  
asynchronous advantage actor critic, 1142  
asymmetric numeral systems, 210  
asymptotic consistency, 1044  
asymptotic equipartition property, 200  
asynchronous updates, 380  
asynchronous value iteration, 1126  
ATE, 180, 1168  
atoms, 1036  
ATT, 1189  
attention layer, 609  
attention score, 609  
attention weight, 609  
attribute, 173  
audio-visual speech recognition, 995  
augmented DAG, 180  
Augmented Inverse Probability of Treatment Weighted Estimator, 1183  
augmented reality, 1007  
Austen plots, 1204  
Austen Sensitivity Model, 1207  
auto-regressive HMM, 970  
auto-regressive model, 835  
Auto-Sklearn, 270  
Auto-Weka, 270  
autoconj, 418  
autocorrelation function, 504  
autocorrelation time, 505  
autocovariance function, 505  
autodiff, 221  
automatic differentiation, 221  
automatic differentiation variational inference, 310, 426  
Automatic Relevance Determination, 663  
automatic relevance determination, 951  
automatic relevancy determination, 585, 628  
automatic speech recognition, 331, 780, 992  
autoregressive, 777  
autoregressive bijection, 852  
autoregressive flows, 852  
autoregressive models, 920  
auxiliary latent variables, 438  
auxiliary variable deep generative model, 438  
auxiliary variables, 484  
average causal effect, 1177  
Average Treatment Effect, 1168  
average treatment effect, 180, 1177

- 1
- 2 average treatment effect on the treated, 1189
- 3 axis aligned, 16
- 4 BA lower bound, 208
- 5 back translation, 747
- 6 backcasting, 736
- 7 backdoor criterion, 1212
- 8 backdoor paths, 1212
- 9 backoff smoothing, 109
- 10 backpropagation, 230
- 11 backup diagram, 1127
- 12 backward Euler method, 999
- 13 backwards kernel, 541
- 14 backwards transfer, 767
- 15 bagging, 636
- 16 balance the dataset, 744
- 17 Bambi, 599
- 18 BAMDP, 1133
- 19 bandit problem, 1111
- 20 bandwidth, 662, 781
- 21 BAOAB, 512
- 22 base distribution, 843
- 23 base measure, 29, 1034
- 24 base-rate, 1059
- 25 baseline, 463
- 26 baseline function, 242
- 27 basic random variables, 173
- 28 basis functions, 964
- 29 batch ensemble, 637
- 30 batch normalization, 607
- 31 batch optimization, 241
- 32 batch reinforcement learning, 1154
- 33 batched Bayesian optimization, 270
- 34 Baum-Welch, 977, 977
- 35 Baum-Welch algorithm, 142
- 36 Bayes ball algorithm, 131
- 37 Bayes by backprop, 631
- 38 Bayes estimator, 121
- 39 Bayes factor, 112
- 40 Bayes filter, 322
- 41 Bayes nets, 125
- 42 Bayes' rule, 64
- 43 Bayes' rule for Gaussians, 18
- 44 Bayes-adaptive MDP, 1133
- 45 BayesBiNN, 264
- 46 Bayesian approach, 67
- 47 Bayesian dark knowledge, 638
- 48 Bayesian decision theory, 120, 1103
- 49 Bayesian deep learning, 625
- 50 Bayesian factor regression, 943
- 51 Bayesian gradient descent, 436
- 52 Bayesian hypothesis testing, 112
- 53 Bayesian inference, 64
- 54 Bayesian information criterion, 116
- 55 Bayesian lasso, 583
- 56 Bayesian learning rule, 257
- 57 Bayesian model selection, 111
- 58 Bayesian multi net, 146
- 59 Bayesian networks, 125
- 60 Bayesian neural network, 625
- 61 Bayesian nonparametric models, 516
- 62 Bayesian nonparametrics, 1033
- 63 Bayesian Occam's razor, 111
- 64 Bayesian online changepoint detection, 989
- 65 Bayesian optimization, 264, 661
- 66 Bayesian p-value, 118
- 67 Bayesian statistics, 63
- 68 Bayesian structural time series, 725
- 69 BayesOpt, 264
- 70 BBB, 631
- 71 BBMM, 691
- 72 BBVI, 418
- 73 beam search, 1043
- 74 Bean Machine, 176
- 75 behavior cloning, 1164
- 76 behavior policy, 1154
- 77 behavior-agnostic off-policy, 1154
- 78 belief networks, 125
- 79 belief propagation, 370
- 80 belief state, 319, 323, 562, 1113, 1133
- 81 belief states, 370
- 82 belief-state MDP, 1115
- 83 Bellman backup, 1125
- 84 Bellman error, 1123
- 85 Bellman residual, 1123
- 86 Bellman's optimality equations, 1123
- 87
- Berkson's paradox, 128, 133
- Bernoulli bandit, 1114
- Bernoulli distribution, 5
- Bernoulli mixture model, 930
- BERT, 615, 812
- Bessel function, 24, 663
- best arm identification, 264, 1109
- best-arm identification, 1113
- best-first search, 1147
- beta distribution, 12, 71
- beta function, 12
- Beta process, 1050
- beta-binomial, 75
- beta-VAE, 803
- Bethe free energy, 381
- Bhattacharya distance, 529
- bi-directed graph, 168
- BIC, 116, 699
- BIC loss, 116
- BIC score, 116
- big data, 69
- BiGAN, 912
- bigram model, 48, 202
- bigram statistics, 50
- bijection, 845
- bilinear form, 611
- bilinear method, 999
- binary entropy function, 198
- binary logistic regression, 587
- binary neural network, 264
- binary partition transformer, 622
- binomial coefficient, 5
- binomial distribution, 5
- binomial regression, 572
- BIO, 719
- bit error, 376
- bits, 190
- bits back coding, 213
- bits per dimension, 784
- BIVA, 819
- bivariate Gaussian, 16
- black box attack, 771
- black box shift estimation, 747
- black box variational inference, 418
- blackbox EP, 448
- blackbox matrix-matrix multiplication, 691
- Blackwell-MacQueen, 1038
- blind inverse problem, 935
- blind source separation, 959
- block length, 213
- block stacking, 1149
- blocked Gibbs sampling, 481
- BLOG, 176
- BN, 607
- NN, 625
- Bochner's theorem, 669
- BOCPD, 989
- Boltzmann machine, 151
- Boltzmann policy, 1132
- bond variables, 487
- Bonnet's theorem, 237, 244
- bootstrap filter, 523
- bootstrapping, 1130, 1134
- borrow statistical strength, 101
- bottom-up inference model, 819
- bouncy particle sampler, 468
- bound optimization, 247, 250
- Box-Muller, 455
- BP, 370
- BPT, 622
- brain, 340
- branching factor, 202
- Bregman divergence, 197, 239, 240, 432
- Brier score, 558
- BRMS, 599
- Brownian motion, 494, 495, 664, 1056
- Brownian noise, 509, 512
- BSS, 959
- BSTS, 725
- bucket elimination, 385
- building blocks, 603
- burn-in phase, 497
- burn-in time, 467
- burstiness, 1059
- calculus of intervention, 1211
- calculus of variations, 38
- calibrated, 558
- calibration set, 566
- canonical correlation analysis, 945
- canonical form, 17, 30, 33
- canonical link function, 574
- canonical parameters, 17, 29, 33, 155
- CAQL, 1124
- cart-pole swing-up, 1149
- catastrophic forgetting, 766
- categorical, 6
- categorical PCA, 946
- CatPCA, 946
- Cauchy, 10
- Cauchy sequence, 676
- causal convolution, 837
- causal DAGs, 741
- causal discovery, 1030
- Causal graphs, 1171
- causal hierarchy, 182
- causal impact, 183, 735
- Causal inference, 1167
- causal Markov assumption, 177
- causal models, 177
- causal prediction, 741
- causally sufficient, 177
- causes of effects, 181
- CAVI, 403
- cavity distribution, 447
- CCA, 945
- cdf, 8
- CEB, 218
- CelebA, 782, 798
- centering matrix, 87
- central composite design, 696
- central limit theorem, 453
- central moment, 11
- certify, 773
- ceteris paribus, 183
- chain components, 166
- chain compositions, 226
- chain graph, 158, 166, 167
- chain rule, 225
- chance nodes, 1103
- change of variables, 44
- change-point detection, 748, 987, 989
- channel coding, 185, 213
- channel coding theorem, 383
- Chapman-Kolmogorov, 47
- Chapman-Kolmogorov equation, 322
- characteristic length scale, 663
- Chernoff-Hoeffding inequality, 1116
- chi-squared distance, 56
- Chi-squared distribution, 13
- children, 125
- Chinese restaurant process, 1038
- CHIVI, 443
- choice theory, 594
- Cholesky decomposition, 455, 508
- Chomsky normal form, 720, 721
- chordal, 164, 392
- CI, 125
- circuits, 228
- circular flow, 858
- circular normal, 24
- citation matching, 176
- clamped phase, 159, 867
- class incremental learning, 764
- classical statistics, 63
- Claude Shannon, 210
- click through rate, 1110, 1113
- clinical trials, 1113
- CLIP, 841
- clique, 146
- cliques, 387
- closed world assumption, 175, 1012
- closing the loop, 1008
- closure, 156
- cluster variational method, 381
- clutter problem, 312
- CMA-ES, 1151
- CNN, 613
- co-information, 205
- co-parents, 135
- coagulation, 1047
- coalescence, 524
- cocktail party problem, 959
- code words, 210
- codebook, 826

- 1  
 2 codebook loss, 826  
 2 codewords, 185  
 3 coffee, lemon, milk, and tea, 280  
 4 cold start problem, 69  
 4 collapsed, 142  
 5 collapsed Gibbs sampler, 482, 1041  
 6 collider, 131, 1171  
 7 collocation, 1148  
 7 coloring, 476  
 8 commitment loss, 827  
 8 common random number, 1150  
 9 commutative semi-ring, 396  
 9 commutative semiring, 397  
 10 compact support, 691  
 10 Compactness, Sparsity, 1082  
 11 compatible, 1146  
 11 complementary log-log, 574  
 12 complete, 676  
 12 complete data, 159  
 13 completely random measures, 1054  
 13 Completeness, 1082, 1084  
 14 completing the square, 89  
 14 complexity penalty, 115  
 15 compiler average treatment effect, 1194  
 15 components, 725  
 16 composite likelihood, 161  
 16 compositional kernel, 706  
 17 compositional pattern-producing network, 772  
 17 Compression Lemma, 192  
 18 computation graph, 603  
 18 computation tree, 378  
 19 concave, 37  
 20 concentration inequality, 1116  
 20 concentration of measure, 774  
 21 concept drift, 763  
 21 concept shift, 743  
 22 concrete distribution, 245  
 22 condensation, 523  
 23 conditional entropy bottleneck, 218  
 24 conditional expected outcome, 1181  
 24 conditional GAN, 911  
 25 conditional generative model, 777  
 25 Conditional generative models, 911  
 26 conditional independence, 125  
 26 conditional KL divergence, 189  
 27 conditional maxen model, 717  
 27 conditional parallel trends, 1200  
 28 conditional probability distribution, 126  
 28 conditional random field, 717  
 29 conditional random fields, 146, 148  
 30 Conditional shift, 743  
 30 conditional SMC, 550  
 31 conditional value at risk, 1129  
 31 conditionally conjugate, 87  
 32 conditioner, 851, 852  
 32 conditioning case, 127  
 33 conditioning matrix, 231  
 34 conductance, 498  
 34 confidence score, 749  
 35 conformal prediction, 565, 748, 750  
 35 conformal score, 565  
 35 conformalized quantile regression, 568  
 36 confounder, 169  
 36 confounders, 1178  
 37 conical combination, 849  
 37 conjugate, 70, 305  
 38 conjugate gradients, 691  
 38 conjugate prior, 19, 29, 70, 71  
 39 conjugate-computation variational inference, 430  
 40 conjunction of features, 864  
 40 consensus sequence, 976  
 41 conservative policy iteration, 1144  
 41 consistent, 1034  
 42 constraint satisfaction problems, 396, 397  
 43 content constrained, 772  
 43 context free grammar, 720  
 44 context variables, 758  
 44 Context., 1069  
 45 contextual bandit, 1112  
 45 continual learning, 650, 753, 762  
 46 continuation method, 474  
 46 continuing task, 1122  
 47 continuous task-agnostic learning, 764  
 47 continuous time, 998  
 47 continuous-ranked probability score, 734  
 47 continuous-time flows, 859  
 47 contraction, 1125  
 47 contrastive divergence, 160, 866  
 47 Contrastiveness, 1084  
 47 control, 1106  
 47 control theory, 1121  
 47 control variate, 463, 510  
 47 control variates, 242, 419  
 47 controller, 1121  
 47 controls, 997  
 47 converge, 498  
 47 conversions, 1110  
 47 convex combination, 73  
 47 ConvNeXt, 613  
 47 convolution, 605  
 47 convolutional layer, 606  
 47 convolutional Markov model, 837  
 47 convolutional neural network, 613, 836  
 47 convolutional neural networks, 722  
 47 cooling schedule, 475  
 47 cooperative cut, 286  
 47 coordinate ascent variational inference, 403  
 47 core sets, 546  
 47 coresets, 682  
 47 correlation coefficient, 17  
 47 correspondence, 1011  
 47 cosine distance, 24  
 47 cosine kernel, 665  
 47 count-based exploration, 1132  
 47 Counterfactual queries, 1175  
 47 counterfactual question, 181  
 47 counterfactual reasoning, 734  
 47 coupled HMM, 995  
 47 coupling flows, 851  
 47 coupling layer, 851  
 47 covariance function, 659  
 47 covariance graph, 168, 1031  
 47 covariance matrix, 16  
 47 covariate shift, 742  
 47 coverage, 566  
 47 Cox process, 680  
 47 CPD, 126  
 47 CPPN, 772  
 47 CPT, 127  
 47 CQR, 568  
 47 credible interval, 66, 85  
 47 CRF, 146, 717  
 47 CRFs, 148  
 47 critic, 894  
 47 critical temperature, 149, 488  
 47 cross correlation, 605  
 47 cross entropy, 195, 201  
 47 cross entropy method, 1148  
 47 cross fitting, 1185  
 47 cross validation, 113  
 47 cross-entropy method, 547  
 47 CRP, 1038  
 47 CRPS, 734  
 47 CTR, 1113  
 47 cubature, 451  
 47 cubature Kalman filter, 359  
 47 CUBO, 444  
 47 cumulants, 34  
 47 cumulative distribution function, 8  
 47 cumulative regret, 1119  
 47 cumulative reward, 1112  
 47 curse of dimensionality, 781, 1124  
 47 curse of horizon, 1157  
 47 curved exponential family, 30  
 47 CVI, 260, 430  
 47 cycle consistency, 919  
 47 cyclical annealing, 816  
 47 d-separated, 131  
 47 D4PG, 1146  
 47 DAGs, 125  
 47 DALL-E, 840  
 47 damped updates, 406  
 47 damping, 378, 449  
 47 dark knowledge, 638  
 47 DARN, 131  
 47 data assimilation, 356  
 47 data association, 1011  
 47 data augmentation, 596, 747  
 47 data cleaning, 749  
 47 Data compression, 210  
 47 data compression, 185, 780  
 47 data generating process, 741  
 47 data processing inequality, 193, 203  
 47 data stream classification, 764  
 47 data tempering, 519  
 47 data-driven MCMC, 472  
 47 datatset shift, 739  
 47 daydream estimator, 445  
 47 daydream phase, 832  
 47 DBN, 153, 996  
 47 DCGAN, 913  
 47 DDP, 1148  
 47 DDPG, 1146  
 47 DDPM, 883  
 47 de Finetti's theorem, 65  
 47 dead leaves, 934  
 47 decision diagram, 1103  
 47 decision nodes, 1103  
 47 decision tree, 1105  
 47 declarative approach, 177  
 47 decoder, 791, 798, 927  
 47 decomposable, 161, 164, 393  
 47 decompose, 139  
 47 decoupled EKF, 652  
 47 deep autoregressive network, 131  
 47 deep belief network, 153  
 47 deep Boltzmann machine, 153  
 47 deep Boltzmann network, 153  
 47 deep CCA, 945  
 47 deep deterministic policy gradient, 1146  
 47 deep ensembles, 635  
 47 Deep Factors, 734  
 47 deep fakes, 779, 917  
 47 deep Gaussian process, 711  
 47 deep generative model, 130  
 47 deep generative models, 777  
 47 deep image prior, 628  
 47 Deep kernel learning, 703  
 47 deep latent Gaussian model, 791  
 47 deep latent variable model, 791  
 47 deep learning, 603  
 47 deep Markov model, 1017  
 47 deep neural network, 603  
 47 deep PILCO, 1150  
 47 deep Q-network, 1139  
 47 deep state space models, 1017  
 47 deep submodular function, 286  
 47 deep unfolding, 385  
 47 DeepAR, 734  
 47 DeepGLO, 734  
 47 DeepSSM, 734  
 47 default prior, 95  
 47 deformable parts model, 723  
 47 degenerate kernel, 668  
 47 degree of normality, 10  
 47 degrees of freedom, 10, 82, 116  
 47 deleted interpolation, 109  
 47 delta function, 66  
 47 delta method, 245  
 47 delta VAE, 816  
 47 demand forecasting, 1013  
 47 denoising diffusion probabilistic models, 874, 883  
 47 Denoising Score Matching, 871  
 47 density estimation, 780  
 47 density model, 750  
 47 Derivative free optimization, 299  
 47 derivative function, 223  
 47 derivative operator, 223  
 47 detailed balance, 469  
 47 detailed balance equations, 54  
 47 determinantal point process, 332  
 47 Determinantal point processes, 1062  
 47 determinantal projection point processes, 1064  
 47 deterministic annealing, 980  
 47 deterministic inducing conditional, 685  
 47 deterministic policy gradient theorem, 1145  
 47 deterministic training conditional, 685  
 47 deviance, 115  
 47 DFO, 299  
 47 DGM, 125  
 47 DGP, 711  
 47 diagonal covariance matrix, 17  
 47 diameter, 374  
 47 DIC, 685

- 1
- 2 dictionary learning, 964  
diffeomorphism, 845  
3 difference in differences, 1199  
differential dynamic programming, 1148  
4 differential entropy, 198  
diffuse prior, 95  
5 diffusion matrix, 495  
diffusion models, 777  
6 diffusion process, 883  
diffusion term, 495  
7 digamma function, 413  
dilated convolution, 837  
8 diminishing returns, 282  
direct cause, 181  
9 direct method, 1154  
directed acyclic graphs, 125  
10 directed Gaussian graphical model, 129  
directed graphical models, 125  
11 Dirichlet, 27  
Dirichlet distribution, 78  
12 Dirichlet process, 484, 1025, 1027, 1034  
Dirichlet process mixture models, 415  
13 discount factor, 1112, 1122, 1129  
discount parameter, 1045  
14 discrete task-agnostic learning, 765  
discrete with probability one, 1036  
15 discriminative model, 555  
discriminative reranking, 331  
16 discriminator, 894  
disease mapping, 680  
17 disease transmission, 1023  
disentangled, 804, 963  
18 dispersion parameter, 38  
distillation, 638  
19 distortion, 210  
distributed representation, 152, 992  
20 distribution free, 565  
distribution shift, 739, 773, 1153  
21 distributional particles, 537  
distributional RL, 1140, 1146  
22 distributionally robust optimization, 748  
23 distributive law, 396  
divergence metric, 54  
24 DLGM, 791  
DLM, 725  
25 DLVM, 791  
DNN, 603  
26 DNN factor analysis, 947  
do-calculus, 1211  
27 do-notation, 1175  
domain adaptation, 746  
28 domain adversarial learning, 746  
domain drift, 763  
29 domain generalization, 655, 758, 758  
domain randomization, 747, 1152  
30 domain shift, 742  
domains, 758  
31 donor, 736  
Donsker Varadhan lower bound, 209  
32 Donsker-Varadhan, 192  
dot product attention, 609  
33 double descent risk curve, 643, 645  
double DQN, 1139  
34 double loop algorithms, 379  
double machine-learning, 1183  
35 double Q-learning, 1139  
double robust, 1185  
36 double sided exponential, 10  
doubly intractable, 159  
37 doubly reparameterized gradient estimator, 442  
38 doubly robust, 1156  
doubly stochastic, 421  
39 downstream, 783  
DQN, 1139  
40 Dreamer, 1153  
drift, 512  
41 dropout, 607  
DTC, 685  
42 DualDICE, 1158  
dueling DQN, 1139  
43 Dutch book, 65  
dyna, 1148  
44 dynamic Bayesian network, 996  
dynamic linear model, 339, 725, 1003  
45 dynamic programming, 306, 369, 392, 1124  
46 dynamic programming., 390  
47
- dynamic topic model, 782  
dynamic VAE, 812  
dynamical variational autoencoders, 1017  
E step, 250, 251  
earning while learning, 1112  
EB, 106  
EBM, 777, 863  
ECE, 559  
ECM, 256  
ECME, 256  
edge potentials, 155  
edit distance, 974  
EEKF, 349  
effective dimensionality, 641  
effective sample size, 505, 526  
effects of causes, 181  
EI, 268  
eigenfunction, 668  
eigengap, 498  
eight schools, 103  
Einstein summation, 398  
einsum, 398  
einsum networks, 172  
EKF, 346  
elastic weight consolidation, 654, 767  
ELBO, 250, 310, 402, 793  
elementwise flow, 848  
eligibility traces, 1135  
elimination, 392  
elimination order, 387  
ELPD, 114  
EM, 141, 250  
empirical Bayes, 106, 628  
empirical distribution, 194  
empirical Fisher, 236  
empirical risk, 556  
empirical risk minimization, 556  
empirical risk minimization, 257  
emulate, 548  
encoder, 798  
End-task., 1069  
endogenous, 177  
energy, 148  
energy based model, 148  
energy based models, 777  
energy disaggregation, 992  
energy function, 308, 473, 863, 1061  
energy score, 749  
Energy-based models, 863  
EnKF, 356  
ensemble, 607  
ensemble Kalman filter, 356  
entity resolution, 175  
entropy, 197, 250  
entropy search, 269  
environment, 1111, 1120  
environments, 758  
EP, 447  
epidemiology, 523  
episodic task, 1122  
epistemic uncertainty, 67, 562  
epistemological uncertainty, 1025  
EPLL, 933  
epsilon-greedy, 1131  
episode, 1122  
equilibrium distribution, 51  
equivalent sample size, 71  
ergodic, 54  
Erlang distribution, 13  
ERM, 257, 556  
error correcting codes, 213, 382  
error correction, 185  
error-correcting codes, 171  
ES-RNN, 733  
ESS, 526  
estimated potential scale reduction, 502  
estimator, 63  
Etsy, 974  
EURO, 444  
Euler approximation, 509  
Euler's method, 490, 859, 999  
evidence, 64, 75, 111, 309, 793  
evidence lower bound, 250, 310, 402, 793  
evidence maximization, 628  
evidence upper bound, 444  
evolutionary search, 299  
evolutionary strategies, 264  
EWC, 654  
excess kurtosis, 11  
exchangeable, 144  
exchangeable process, 1038  
exchangeable with, 101  
Exclusion Restriction, 1192  
execution traces, 177  
exogenous, 177  
exp-sine-squared kernel, 665  
expanded parameterization, 931  
expectation backpropagation, 653  
expectation maximization, 250  
expectation propagation, 447  
expected calibration error, 559  
expected complete data log likelihood, 251  
expected free energy, 1163  
expected improvement, 268  
expected LPPD, 114  
expected patch log likelihood, 933  
expected sufficient statistics, 142, 251  
experience replace, 1139  
experience replay, 767  
explainability, 182  
explaining away, 87, 128, 133, 137, 206  
explicit duration HMM, 986  
explicit layers, 612  
explicit probabilistic models, 891  
exploration bonus, 1115, 1132  
exploration-exploitation tradeoff, 1106, 1113, 1131  
exponential cooling schedule, 475  
exponential dispersion family, 38  
Exponential distribution, 13  
exponential family, 28, 29, 39, 93  
Exponential family EKF, 349  
exponential family factor analysis, 945  
exponential family harmonium, 152  
exponential family PCA, 945  
exponential family state space model, 1012  
Exponential linear unit, 605  
exponentiated quadratic, 662  
extended Kalman filter, 346, 1006  
extended particle filter, 534  
external field, 150  
external validity, 739, 741  
extrapolation, 693  
extrinsic variables, 410  
f-divergence, 55  
f-Divergence Max-Ent IRL, 1165  
Facebook, 1026  
factor, 146  
factor analysis, 144  
factor graph, 169, 374, 383  
factor loading matrix, 144, 935  
factor of variation, 804  
factor rotations problem, 938  
factorial HMM, 438, 992  
factorization property, 135  
FAIRL, 1165  
fairness, 182  
Faithfulness, Fidelity, 1082  
family marginal, 142  
fan-in, 228  
fan-out, 225, 228  
fantasy data, 868  
fast geometric ensembles, 633  
fast gradient sign, 771  
fast ICA, 962  
fast weights, 637  
FastSLAM, 540  
feature-based, 284  
feedback loop, 1113  
feedforward neural network, 613  
ferromagnetic, 149  
few-shot learning, 755, 839  
Feynman-Kac, 517  
FFG, 171  
FFJORD, 860  
FFNN, 613  
FGS, 771  
FIC, 686  
FID, 786  
fill-in, 392

- 1  
 2 fill-in edges, 387  
 FILM, 612  
 filter, 605  
 filter response normalization, 607  
 filtering, 319  
 FIM, 39  
 finite horizon, 1112  
 finite horizon problem, 1122  
 finite state machine, 1121  
 finite sum objective, 241  
 finite-state Markov chain, 47  
 first order stationary, 505  
 first-order delta method, 245  
 Fisher divergence, 870  
 Fisher information, 96  
 Fisher information matrix, 36, 39, 97  
 FITC, 686  
 fitted value iteration, 1139  
 fixed effects, 599  
 Fixed lag smoothing, 320  
 flat minima, 495, 639  
 fluctuation, 512  
 folded over, 9  
 folds, 113  
 FOO-VB, 435  
 fooling images, 772  
 force, 512  
 forest plot, 103  
 fork, 131  
 Forney factor graph, 171  
 Forney factor graphs, 169  
 forward adversarial inverse RL, 1165  
 forward transfer, 767  
 forward-mode automatic differentiation, 226  
 forwards algorithm, 323, 389, 977  
 forwards filtering backwards sampling, 332  
 forwards filtering, backwards smoothing, 325  
 forwards kernel, 542  
 forwards mapping, 925  
 forwards process, 883  
 forwards-backwards, 390, 719  
 forwards-backwards algorithm, 325, 327  
 founder variables, 938  
 Fourier basis, 965  
 Fourier transform, 669  
 Fréchet Inception Distance, 786  
 fragmentation, 1047  
 Frechet inception distance, 60  
 free bits, 816  
 free energy, 402  
 free energy principle, 1163  
 freeze-thaw algorithm, 271  
 frequentist sampling distribution, 579  
 frequentist statistics, 63  
 friction, 512  
 front-door criterion, 1214  
 front-door formula, 1214  
 frustrated, 488  
 frustrated system, 149  
 full conditional, 135, 476  
 full conditionals, 311  
 full conformal prediction, 569  
 full covariance matrix, 17  
 fully connected CRF, 722  
 fully connected layer, 604  
 fully independent conditional, 686  
 fully independent training conditional, 686  
 function space, 673  
 functional, 221  
 functional causal model, 177  
 functional kernel learning, 705  
 fundamental problem of causal inference, 183  
 funnel shape, 105  
 funnel transformer, 812  
 fuzzy clustering, 1025  
 g-prior, 580  
 GAE, 1143  
 GAIL, 1165  
 GAM, 698  
 Gamma, 986  
 gamma distribution, 12  
 GAN, 777  
 GANs, 891  
 GaP, 955  
 gap, 1119  
 GAT, 622  
 gated recurrent unit, 615  
 gather, 620  
 Gauss-Hermite integration, 359  
 Gaussian bandit, 1115, 1116  
 Gaussian Bayes net, 129  
 Gaussian copula, 734  
 Gaussian distribution, 8  
 Gaussian filter, 357  
 Gaussian filtering, 361  
 Gaussian graphical model, 154  
 Gaussian kernel, 662  
 Gaussian mixture model, 928  
 Gaussian MRF, 154  
 Gaussian process, 266, 659, 1033, 1034  
 Gaussian processes, 432, 732, 1149  
 Gaussian scale mixture, 254, 584, 980  
 Gaussian soap bubble, 200  
 Gaussian SSMs, 998  
 Gaussian sum filter, 361  
 Gaussian VI, 422  
 Gaussianizing the likelihood, 433  
 GELU, 605  
 GEM, 256  
 Gen, 177  
 GenDICE, 1158  
 Generality, 1085  
 generalization, 784  
 generalized additive model, 698, 731  
 generalized advantage estimation, 1143  
 generalized Bayesian inference, 259, 557  
 generalized belief propagation, 381  
 generalized bilinear transform, 999  
 generalized CCA, 945  
 generalized distributive law, 396  
 generalized EM, 256  
 generalized Gauss-Newton, 630  
 generalized linear mixed model, 599  
 generalized linear model, 571  
 generalized low rank models, 946  
 generalized online learning, 768  
 generalized policy improvement, 1127  
 generalized pseudo Bayes filter, 363  
 generalized variational continual learning, 436  
 generate and test, 472  
 generative adversarial imitation learning, 1165  
 Generative Adversarial Networks, 891  
 generative adversarial networks, 777  
 generative model, 777  
 generative weights, 960  
 geometric distribution, 7, 985  
 geometric path, 543  
 GGM, 154  
 GGN, 630  
 Gibbs distribution, 148, 863  
 Gibbs point processes, 1061  
 Gibbs posterior, 557  
 Gibbs sampling, 149, 311, 476  
 Gittins index, 1115  
 Glauber dynamics, 476  
 GLIDE, 842  
 GLIE, 1136  
 GLM, 571  
 GLM predictive distribution, 638  
 GLMM, 599  
 global balance equations, 52  
 global latent variables, 304  
 global localization, 530  
 global Markov property, 131, 155  
 global variables, 138  
 globally normalized, 718  
 Glorot initialization, 626  
 GMM, 928  
 GNNS, 618  
 goodness-of-fit, 54  
 GP, 266  
 GP-LVM, 947  
 GP-MPC, 1150  
 GP-SSM, 1016  
 GP-UCB, 268  
 GPT, 839  
 GPT-2, 839  
 GPT-3, 839  
 gradient descent, 231  
 gradient sign reversal, 746  
 gradient-based meta-learning, 761  
 Gram matrix, 661  
 grammar VAE, 815  
 grammars, 720  
 graph attention network, 622  
 Graph Nets, 618  
 graph neural networks, 618  
 graph surgery, 179, 1172  
 graphical lasso, 155, 1029  
 greatest common divisor, 53  
 greedy action, 1124  
 grid approximation, 306  
 grid world, 1124  
 ground network, 174  
 ground set, 1062  
 ground states, 149  
 ground terms, 173  
 group lasso, 584  
 group normalization, 607  
 GRU, 615  
 GSM, 930  
 guided cost learning, 1165  
 guided particle filter, 532  
 Gumbel distribution, 245  
 Gumbel-Max trick, 245  
 Gumbel-Softmax, 828  
 Gumbel-Softmax distribution, 245  
 half Cauchy, 10  
 half-Cauchy distribution, 585  
 half-edge, 171  
 half-normal distribution, 9  
 Halton sequence, 465  
 Hamilton's equations, 489  
 Hamiltonian, 488  
 Hamiltonian mechanics, 488  
 Hamiltonian Monte Carlo, 311, 488, 511, 593  
 Hamiltonian Variational Inference, 441  
 Hammersley-Clifford theorem, 147  
 Hamming distance, 68  
 Hamming loss, 69  
 hard clustering, 929  
 hard EM, 256  
 hard tanh, 246  
 hardcore repulsive process, 1062  
 harmonic mean estimator, 112  
 harmonium, 152  
 Hastings correction, 468  
 Hawkes processes, 1059  
 hazard function, 988  
 heat bath, 476  
 heavy tailed, 15  
 heavy tails, 10, 15  
 Hebb's rule, 867  
 Hebbian learning, 867  
 Hebbian term, 867  
 Hebbian update rule, 342  
 Hellinger distance, 56  
 Helmholtz machine, 792  
 Hessian free optimization, 236  
 heuristic function, 1147  
 heuristic search, 1148  
 hidden Gibbs random field, 161  
 hidden Markov model, 316, 835, 967  
 hidden semi-Markov model, 986  
 hidden state, 835  
 hidden variable, 967  
 hidden variables, 136, 250  
 hierarchical Bayesian model, 100, 598, 655  
 hierarchical Bayesian models, 506  
 hierarchical Dirichlet process, 1046  
 hierarchical generalized linear model, 598  
 hierarchical HMM, 992  
 hierarchical kernel learning, 698  
 hierarchical VAE, 819  
 hierarchical variational model, 438  
 Hilbert space, 676  
 hindsight, 320  
 Hinton diagram, 49  
 Hinton diagrams, 951  
 histogram binning, 560  
 histogram filter, 530  
 HMC, 311, 488  
 HMM, 967

- 1
- 2 HMM filter, 333  
homogeneous, 46  
3 homogeneous Poisson process, 1057  
horizon, 321  
4 horseshoe distribution, 931  
horseshoe prior, 584  
5 HSMM, 986  
Huffman coding, 210  
6 Hungarian algorithm, 981, 1011  
Hutchinson trace estimator, 858  
7 Hutter prize, 210  
hybrid MC, 488  
8 hybrid system, 361  
hyper-parameters, 71  
9 hypergeometric distribution, 8  
hypergraphs, 621  
10 hypernetwork, 611  
hyperparameters, 100  
11 hypothesis test, 1109
- 12 I-map, 136  
I-projection, 446  
13 IAF, 855  
14 IBIS, 545  
ICA, 960  
15 ID, 748  
identifiable, 1173  
16 identification strategy, 1173  
identified, 1173  
17 identity uncertainty, 175  
iid, 70  
18 image deblurring, 933  
image denoising, 933  
19 image imputation, 955  
image inpainting, 933  
20 image super-resolution, 933  
image to image translation, 917  
21 imagination-augmented agents, 1149  
Imbens' Sensitivity Model, 1204  
22 imitation learning, 1163  
IMM, 364  
23 implicit generative models, 785, 891  
implicit layers, 612  
24 implicit models, 547, 777  
implicit probabilistic model, 891  
25 implicit probabilistic models, 891  
implicit probability distributions, 440  
26 implicit probability model, 805  
importance ratio, 1155  
27 importance sampling, 310, 458  
importance weighted autoencoder, 441  
28 importance weighted autoencoders, 460  
importance weights, 459  
29 imputation, 781  
in-context learning, 839  
30 in-distribution, 748  
in-domain, 739  
31 in-painting, 781  
Inception, 60  
32 Inception Score, 786  
income inequality, 15  
33 incomplete data, 161  
incremental EM, 257  
34 incremental importance weights, 520  
incumbent, 267  
35 independence sampler, 470  
independent and identically distributed, 70  
36 independent components analysis, 960  
37 indirect cause, 181  
induced width, 388  
38 inducing points, 684  
inference, 64, 136, 303  
39 inference compilation, 832  
inference network, 429, 603, 791  
40 infinite hidden relational model, 1027  
infinite horizon, 1112  
41 infinite mixture model, 1040  
42 infinitely divisible distribution, 1056  
infinitely exchangeable, 64  
43 infinitely wide neural networks, 627  
influence curve, 1183  
44 influence diagram, 180, 1103  
influence model, 995  
45 infomax, 964  
InfoNCE, 209  
46 information arc, 1104
- information bottleneck principle, 215  
information criterion, 115  
information diagram, 203, 206  
information diagrams, 203  
information filter, 337  
information form, 17, 33, 155  
information gain, 187, 269  
information processing, 185  
information projection, 361, 446  
information state, 1114  
informative vector machine, 682  
InfoVAE, 805, 817  
inhomogeneous Poisson process, 1057  
inner product, 675  
innovation, 384  
input nodes, 228  
inside outside, 992  
inside-outside algorithm, 720  
instance normalization, 607  
instantaneous ELBO, 430  
instrument monotonicity, 1195  
Instrument Relevance, 1192  
Instrument Unconfoundedness, 1192  
instrumental variables, 1191  
integer-valued random measure, 1055  
integral probability metric, 56  
integrating out, 66  
inter-causal reasoning, 133  
interaction information, 205  
interactive multiple models, 364  
Interactivity, 1083  
interpolated Kneser-Ney, 111  
interpolator, 670  
intervention, 183  
interventions, 179  
Interventional queries, 1175  
intrinsic uncertainty, 67  
intrinsic variables, 410  
invariant, 96, 469  
invariant causal prediction, 758  
invariant distribution, 51  
invariant risk minimization, 759  
inventory data, 1015  
inverse autoregressive, 855  
inverse autoregressive flow, 439  
inverse chi-squared distribution, 82  
inverse Gamma, 81, 577  
inverse Gamma distribution, 13  
inverse mass matrix, 492  
inverse optimal control, 1164  
inverse probability of treatment  
weighted estimator, 1182  
inverse probability theory, 63  
inverse probability transform, 454  
inverse reinforcement learning, 1164  
inverse temperature, 543  
inverse Wishart, 26, 86, 87  
IPF, 160  
IPM, 56  
iResNet, 857  
IRLS, 249  
IRM, 759, 1027  
irreducible, 52  
Ising model, 148, 150  
Ising models, 477  
isotonic regression, 560  
isotropic covariance matrix, 17  
iterated batch importance sampling, 545  
iterated EKF, 347  
iterative amortized inference, 430  
iterative proportional fitting, 160  
IWAE, 441  
IWAE bound, 442
- jackknife+, 569  
Jacobi, 380  
Jacobian, 44  
Jacobian determinant, 844  
Jacobian vector product, 873  
Jacobian-vector product (JVP), 223  
JamBayes, 1030  
Jeffrey's conditionalization rule, 196  
Jeffreys prior, 84, 96  
Jensen's inequality, 188, 250, 442  
JMLS, 362  
JPDA, 1011  
JTA, 390  
jtree, 391
- judge fixed effects, 1192  
jump Markov linear system, 362  
junction graph, 392  
junction tree, 390, 391  
junction tree algorithm, 390  
junction trees, 1009
- K-means clustering, 930  
Kalahari, 581  
Kalman filter, 20, 333, 339, 342, 1001  
Kalman gain matrix, 334, 340  
Kalman smoother, 20, 434, 1001  
Kalman smoothing, 342  
KDE, 781  
kernel, 468, 605  
kernel basis function expansion, 587  
kernel density estimation, 25, 781  
kernel function, 661, 676  
Kernel Inception Distance, 787  
kernel inception distance, 60  
kernel mean embedding, 58  
kernel PCA, 947  
kernel ridge regression, 675, 677  
kernel trick, 58, 668  
keys, 608  
KFAC, 235, 631  
kick, 512  
kinetic energy, 489  
KISS, 698  
KISS-GP, 692  
KL annealing, 816  
KL divergence, 187, 251  
knots, 850  
knowledge gradient, 269  
knowledge graph, 621  
known knowns, 749  
known unknowns, 749  
Kolmogorov-Smirnov test, 45  
kriging, 660  
Kronecker FActored Curvature, 631  
Krylov subspace methods, 690  
Kullback Leibler divergence, 55  
Kullback-Leibler divergence, 187, 251  
kurtosis, 11, 963
- L-ensembles, 1063  
L<sub>0</sub> norm, 582  
L<sub>0</sub> regularization, 582  
L<sub>2</sub>, 676  
L<sub>2</sub> loss, 123  
Lévy processes, 1055  
Lévy subordinators, 1055  
Lévy-Itô decomposition, 1057  
Lévy-Kintchine, 1056  
label bias, 718  
label shift, 743  
label smoothing, 561  
ladder network, 820  
lag, 320, 504  
Lagrange multipliers, 38  
Lagrangian, 38  
lambda-return, 1135  
Lanczos algorithm, 942  
Langevin diffusion, 494, 508  
Langevin MCMC, 866  
Langevin Monte Carlo, 493  
language models, 48  
LapGAN, 914  
Laplace approximation, 307, 590  
Laplace distribution, 10  
Laplace Gaussian filter, 533  
Laplace propagation, 448  
Laplace's rule of succession, 74  
Laplace-EM, 1012, 1013  
lasso, 582, 583  
latent Dirichlet allocation, 484, 959  
latent factors, 925  
latent overshooting, 1020  
latent space interpolation, 782, 813  
latent variable, 927  
latent variable collapse, 823  
latent variable model, 925, 927  
Lauritzen-Spiegelhalter, 395  
layer, 604  
layer normalization, 607  
layers, 603  
lazy training, 710  
LBP, 377
- 47

- 1  
 2 LDA, 484, 959  
 LDPC code, 382  
 3 LDS, 998  
 Leaky ReLU, 605  
 4 Leapfrog integrator, 490  
 learned loss function, 913  
 5 learning, 1129  
 learning from demonstration, 1163  
 least mean squares, 340  
 least squares classification, 643  
 7 leave-one-out cross validation, 113  
 LeCun initialization, 626  
 8 left-to-right, 328  
 left-to-right transition matrix, 47  
 9 legal reasoning, 182  
 leptokurtic, 11  
 10 level sets, 16  
 LG-SSM, 998  
 11 life-long learning, 762  
 likelihood function, 64  
 12 likelihood ratio, 112, 750  
 likelihood ratio gradient estimator, 242  
 13 likelihood tempering, 519  
 likelihood-free inference, 547, 892  
 14 lily pads, 969  
 limiting distribution, 53  
 15 linear assignment problem, 981  
 linear dynamical system, 318, 998  
 16 linear flow, 847  
 linear Gaussian CPD, 129  
 17 linear Gaussian system, 18  
 linear layer, 604  
 18 linear programming, 1124  
 Linear regression, 576  
 19 linear regression bandit, 1115  
 linear-Gaussian CPD, 167  
 20 linear-Gaussian state space model, 318, 998  
 21 linear-Gaussian state space models, 333  
 linear-quadratic-Gaussian, 1148  
 22 linearization point, 223  
 link function, 571, 574  
 Lipschitz constant, 57  
 LKJ distribution, 94  
 24 LM, 48  
 LMC, 493  
 25 LMS, 340  
 local and global latent variables, 305  
 26 local average treatment effect, 1194  
 local evidence, 323, 404  
 27 local factor, 447  
 local latent variables, 304  
 28 local level model, 726  
 local linear model, 1004  
 29 local linear trend, 726  
 local Markov property, 135  
 30 local variables, 138  
 local+global, 734  
 31 localist representation, 152  
 locally normalized, 165, 718  
 32 locally optimal proposal distribution, 532  
 33 location-scale family, 99  
 log derivative trick, 242, 418  
 34 log loss, 556, 558  
 log partition function, 29  
 35 log-derivative trick, 160  
 log-linear model, 154  
 36 log-odds score, 977  
 log-pointwise predictive-density, 114  
 37 log-returns, 1009  
 logical reasoning problems, 397  
 38 logistic, 588  
 logistic distribution, 595, 961  
 39 Logistic regression, 587  
 logistic regression bandit, 1115  
 40 logits, 588  
 long short term memory, 615  
 41 long tail, 69  
 long tails, 15  
 42 LOO-CV, 113  
 loopy belief propagation, 377, 383  
 43 Lorentz, 10  
 loss function, 121  
 44 lossless compression, 210  
 lossy compression, 210  
 45 low discrepancy sequences, 465  
 low variance resampling, 525  
 46 low-density parity check code, 382  
 47  
 low-resource languages, 975  
 LPPD, 114  
 LQG, 1148  
 LSTM, 615  
 LVM, 927  
 M step, 250  
 M-projection, 446  
 m-separation, 167  
 M2, 811  
 M4 forecasting competition, 733  
 machine translation, 780  
 MADE, 836  
 MAF, 854  
 Mahalanobis distance, 16  
 majorize-minimize, 247  
 MALA, 493  
 MAMI, 761  
 manifestation shift, 743  
 MAP estimate, 65, 123, 321, 329, 556  
 MAPIE, 565  
 MAR, 810  
 marginal calibration error, 559  
 marginal likelihood, 64, 75, 77, 106, 111, 191  
 marginalizing out, 66  
 Mariner 10, 337  
 marked, 1060  
 Markov, 136  
 Markov assumption, 46, 315, 835  
 Markov blanket, 135, 156, 476  
 Markov chain, 46  
 Markov chain Monte Carlo, 311, 467  
 Markov decision process, 1120  
 Markov kernel, 46  
 Markov logic networks, 173  
 Markov mesh, 146  
 Markov model, 46, 835  
 Markov model of order n, 48  
 Markov network, 146  
 Markov process, 1033  
 Markov random field, 146  
 masked attention, 609  
 masked autoregressive flow, 854  
 masked convolution, 837  
 matching, 1186  
 Matern kernel, 663  
 matrix determinant lemma, 858  
 matrix inversion lemma, 334  
 matrix variate Gaussian, 24  
 matrix vector multiplication, 690  
 max marginals, 69, 375  
 max-product belief propagation, 375  
 maxent, 154  
 maxent prior, 95  
 maximal clique, 146  
 maximal weight bipartite matching, 1011  
 maximization bias, 1137  
 maximizer of posterior marginals, 69  
 maximizer of the max marginal, 375  
 maximizer of the posterior marginal, 375  
 maximum a posteriori, 123  
 maximum entropy, 95, 154  
 maximum entropy Markov models, 718  
 maximum entropy model, 38  
 maximum entropy RL, 1161  
 maximum expected utility principle, 121  
 maximum likelihood estimation, 556  
 maximum mean discrepancy, 57, 57, 805, 899  
 MBIE, 1132  
 MBIE-EB, 1132  
 MBRL, 1146  
 MCAR, 809  
 MCEM, 256  
 MCMC, 311, 467  
 MCTS, 1148  
 MDL, 116  
 MDP, 1120  
 mean field, 310, 403  
 mean function, 571, 659  
 mean value imputation, 781  
 measure, 676  
 measurement step, 334  
 mediated by, 181  
 mediators, 1213  
 MEMMs, 718  
 MEMO, 753  
 memory methods, 767  
 memorylessness, 1058  
 Mental Model, 1083  
 Mercer kernel, 659, 661  
 Mercer's theorem, 668  
 merit function, 267  
 MERL, 1161  
 message passing, 370, 620  
 message passing algorithms, 369  
 message passing neural network, 618  
 message passing schedule, 370  
 messages, 328, 369  
 meta-data, 758  
 meta-learning, 759  
 Method., 1070  
 Metropolis Adjusted Langevin Algorithm, 493  
 Metropolis Hastings, 467, 473  
 Metropolis Hastings algorithm, 311  
 Metropolis within Gibbs, 481  
 MH, 467  
 midi format, 839  
 min-fill heuristic, 394  
 min-max, 903  
 min-max optimization problem, 748  
 min-weight heuristic, 394  
 minibatch, 241  
 minimal, 29  
 minimal I-map, 136  
 minimal representation, 30  
 minimal sufficient statistic, 204, 215, 215  
 minimally informative prior, 95  
 minimum description length, 116  
 minimum mean squared error, 123  
 minimum weight matching, 981  
 minorize-maximize, 247  
 mirror descent, 238, 240, 432  
 missing at random, 810  
 missing completely at random, 809  
 missing data, 250, 252, 809, 1027  
 missing data mechanism, 810  
 mixed effects, 655  
 mixed effects model, 599  
 mixed graph, 167  
 mixed membership model, 956, 958  
 mixed membership stochastic block model, 1025  
 mixing matrix, 960  
 mixing time, 467, 497, 498  
 mixing weights, 77  
 mixture model, 927  
 mixture of Bernoullis, 930  
 mixture of beta distributions, 76  
 mixture of experts, 636, 864, 1025  
 mixture of factor analysers, 949  
 mixture of Gaussians, 928  
 mixture of Kalman filters, 537  
 mixture of truncated basis functions, 397  
 mixture proposal, 472  
 ML-PIP, 759  
 MLE, 556  
 MLP, 613  
 MM, 247  
 MMD, 57, 57, 805, 899  
 MMD VAE, 805  
 MMI, 205  
 MMM, 375  
 MMSE, 123  
 Möbius inversion formula, 207  
 mode, 123  
 mode collapse, 905  
 mode connectivity, 640  
 mode hopping, 906  
 mode-covering, 191  
 mode-seeking, 191  
 model checking, 117  
 model misspecification, 630  
 model predictive control, 1147  
 model-agnostic meta-learning, 761  
 model-based RL, 1129, 1131, 1146  
 model-free RL, 1129  
 Modified Euler's method, 490  
 Modularity, 1082  
 MoG, 928  
 molecular graph structure, 815  
 moment matching, 37, 108, 159, 358, 364, 446, 900

- 1
- 2 moment parameters, 17, 30  
moment projection, 361, 446
- 3 monference, 792  
monks, 1026
- 4 Monte Carlo, 69  
Monte Carlo approximation, 310  
Monte Carlo control, 1134  
Monte Carlo dropout, 608, 632  
Monte Carlo EM, 256  
Monte Carlo estimation, 1134  
Monte Carlo integration, 451  
Monte Carlo intergration, 454  
Monte Carlo localization, 530, 539  
Monte Carlo methods, 451  
Monte Carlo rollouts, 1130  
Monte-Carlo tree search, 1148  
moralization, 157, 164  
Mormons, 120  
most probable explanation, 377  
motion capture, 948
- 12 MPC, 1147  
mPCA, 956  
13 MPE, 377  
MPM, 69, 375  
14 MQ-CNN, 734  
MRF, 146  
15 multi-armed bandit, 1111  
multi-entity Bayesian networks, 176  
16 multi-head attention, 610  
multi-headed DNN, 765  
17 multi-information, 205  
multi-layer perceptron, 613  
18 multi-level model, 100  
multi-moons, 655  
19 multi-sample ELBO, 442  
multi-scale, 823  
20 multi-stage likelihood, 1015  
multi-target tracking, 175, 1010  
21 multi-task learning, 655, 756  
multiclass logistic regression, 587  
22 multigraphs, 621  
multimodal VAE, 806  
23 multinomial distribution, 6  
Multinomial logistic regression, 588  
24 multinomial logistic regression, 587  
multinomial PCA, 956  
25 multinomial probit, 598  
multinomial resampling, 524  
26 multinoulli, 6  
multiple hypothesis tracking, 175, 363  
27 multiple imputation, 781  
multiple kernel learning, 697  
28 multiple mutual information, 205  
multiple plays, 1112  
29 multiple restarts, 980  
multiple sequence alignment, 977  
30 multiplexer, 174  
multiplicative interactions, 611  
31 multiplicative layers, 611  
multiplicative noise, 931  
32 MultiSWAG, 635  
multivariate Gaussian, 16  
33 multivariate mutual information, 205  
multivariate normal, 16  
34 multivariate probit, 598  
multivariate Student distribution, 23  
35 music transformer, 839  
multilevel model, 598  
36 mutual information, 202  
    MVAE, 806  
37 MVG, 24  
    MVM, 690  
38 MVN, 16  
39 myopic, 1122, 1124
- 40 N-BEATS, 734  
    N-best list, 331  
41 n-gram model, 48  
    NADE, 836  
42 NAGVAC, 632  
    NAGVAC-1, 424  
43 naive Bayes classifier, 145  
named entity extraction, 720  
44 named variable, 221  
nats, 190  
45 natural evolutionary strategies, 236  
natural exponential family, 30  
46 natural gradient, 42, 233, 962
- 47
- natural gradient descent, 39, 232, 240, 259, 1145  
Natural Gradient Gaussian variational approximation, 424  
natural gradient VI, 432  
natural language processing, 719  
natural parameters, 17, 29, 33  
Neal's funnel, 507, 601  
nearest neighbor data association, 1011  
NEF, 30  
negative binomial, 985  
negative binomial distribution, 7  
negative ELBO, 794  
negative log likelihood, 556, 784  
negative phase, 159  
negative transfer, 656  
negentropy, 963  
nested dissection order, 394  
nested plates, 145  
nested SMC, 534  
neural architecture search, 629  
neural auto-regressive density estimator, 836  
neural bandit, 1115  
neural CRF, 719  
neural CRF parser, 721  
neural enhanced BP, 385  
neural net kernel, 707  
neural network Gaussian process, 627  
Neural ODE, 860  
neural processes, 762  
neural spike trains, 533  
neural tangent kernel, 709  
neural-linear, 632  
neuron, 604  
NeuTra, 497  
NeuTra HMC, 492  
neutral process, 1054  
Newton's laws of motion, 998  
Newton's method, 231  
NGD, 232  
NGVI, 432  
NICE, 860  
NIW, 88  
NIX, 83  
NLDS, 1006  
NLP, 719  
NMAR, 810  
NMF, 955  
NNGP, 706  
No-U-Turn Sampler, 492  
node potentials, 155  
Noise Conditional Score Network, 876  
Noise Contrastive Estimation, 877  
noisy channel, 213  
noisy channel model, 973  
noisy nets, 1140  
non-centered parameterization, 105, 507, 602  
non-descendants, 135  
non-factorial prior, 587  
non-linear squared flow, 848  
non-Markovian models, 518  
non-negative matrix factorization, 955  
non-null recurrent, 54  
nonparametric Bayesian models, 926  
non-parametric BP, 377  
non-parametric models, 555  
non-parametrically efficient, 1185  
non-terminals, 720  
nondecreasing, 1056  
noninformative, 95  
nonlinear dynamical system, 1006  
nonparametric copula, 734  
nonparametric models, 659  
nonstationary kernel, 707  
normal distribution, 8  
normal inverse chi-squared, 83  
normal inverse Gamma, 82, 577  
Normal-inverse-Wishart, 88  
normalization layers, 607  
normalized completely random measures, 1055  
normalized occupancy distribution, 1127  
normalized random measures (NRMs), 1055  
normalized stable process, 1055  
normalized weights, 460, 520
- normalizes, 843  
Normalizing flows, 438  
normalizing flows, 777, 843  
not missing at random, 810  
noun phrase chunking, 719  
Nouveau VAE, 824  
novelty detection, 749  
NP-hard, 389  
NTK, 709  
nuisance functions, 1183  
nuisance variables, 136, 377  
null hypothesis, 112  
numerical integration, 451  
NUTS, 492  
NWJ lower bound, 209  
Nyström approximation, 684
- object detection, 722  
objective, 95  
observation model, 968  
observation noise, 1001  
observation overshooting, 1019  
Occam factor, 116  
Occam's razor, 951  
occasionally dishonest casino, 317  
occlusion, 723  
occupancy grid, 529  
off-policy, 1136  
off-policy policy-gradient, 1156  
offline reinforcement learning, 1154  
OGN, 262  
Olivetti faces dataset, 704  
on-policy, 1136  
one-armed bandit, 1111  
one-shot decision problems, 122  
one-shot learning, 755  
one-step ahead prediction distribution, 323  
one-step-ahead predictive distribution, 360  
online adaptation, 763  
online advertising system, 1113  
online Bayesian inference, 338, 1003  
Online elastic weight consolidation, 437  
online EM, 257  
Online Gauss-Newton, 262  
online learning, 650, 767  
Online structured Laplace, 437  
online variational inference, 434  
ontological uncertainty, 1025  
ontology, 1027  
OOD, 748  
open class, 720  
open set recognition, 753  
open universe probability models, 176  
open world, 1012  
open world classification, 753  
open world recognition, 743  
OpenGAN, 751  
opportunity cost, 1106  
optimal action-value function, 1123  
optimal partial policy, 1126  
optimal policy, 121, 1123  
optimal resampling, 538  
optimal state-value function, 1123  
optimal transport, 271  
optimism in the face of uncertainty, 1115  
optimization problems, 221  
optimizer's curse, 1137  
Optimus, 812  
oracle, 264  
ordered Markov property, 125, 135  
ordinal regression, 597  
ordinary differential equation, 998  
Ornstein-Uhlenbeck process, 664  
orthogonal Monte Carlo, 466  
orthogonal random features, 670  
OUPM, 176  
out-of-distribution detection, 749  
out-of-distribution generalization, 739  
out-of-domain, 739  
outlier detection, 749, 780  
outlier exposure, 749  
over-complete representation, 30  
over-parameterized models, 643  
overcomplete representation, 964  
overcounting, 370  
overdispersed, 499

- 1  
 2 overfitting, 74, 556  
 3 overlap, 1180  
 4 Polya urn, 1038  
 5 PAC-Bayes, 557, 646  
 6 padding, 606  
 7 PageRank, 51  
 8 paired data, 910  
 9 pairwise Markov property, 156  
 10 pairwise potentials, 1061  
 11 palindromic, 512  
 12 panel data, 1016, 1198  
 13 parallel prefix scan, 328, 343, 693, 732  
 14 parallel tempering, 499, 516  
 15 parallel trends, 1199  
 16 parallel wavenet, 856  
 17 parameter learning, 138  
 18 parameter tying, 46, 101, 172  
 19 parameterized ReLU, 845  
 20 parametric models, 555  
 21 parametric prior, 1034  
 22 parents, 125  
 23 Pareto distribution, 14  
 24 pareto index, 15  
 25 Pareto smoothed importance sampling, 115  
 26 parity check bits, 214  
 27 part of speech tagging, 718  
 28 partial least squares, 944  
 29 partially directed acyclic graph, 166  
 30 partially observable Markov decision process, 1121  
 31 partially observed, 121  
 32 partially observed data, 252  
 33 partially observed Markov process, 967  
 34 partially pooled model, 598  
 35 particle BP, 377  
 36 particle filtering, 312, 517, 1006, 1043  
 37 particle Gibbs sampling, 550  
 38 particle Gibbs with ancestor sampling, 551  
 39 particle impoverishment, 521  
 40 particle independent MH, 550  
 41 particle marginal Metropolis Hastings, 549  
 42 particle MCMC, 1016  
 43 partition function, 29, 147, 369, 863  
 44 partition of the integers, 1038  
 45 parts, 955  
 46 patchGAN, 918  
 47 path consistency learning, 1159  
 48 path degeneracy, 524  
 49 path diagram, 179  
 50 path sampling, 442  
 51 pathwise derivative, 243  
 52 patience, 420  
 53 PBP, 632, 653  
 54 PCFG, 720  
 55 PCL, 1159  
 56 PDAG, 166  
 57 peaks function, 474  
 58 peeling algorithm, 385  
 59 PEGASUS, 1151  
 60 per-decision importance sampling, 1155  
 61 per-sample ELBO, 430  
 62 per-step importance ratio, 1155  
 63 per-step regret, 1118  
 64 perceptual aliasing, 530, 925  
 65 perceptual distance metrics, 786  
 66 perfect elimination ordering, 393  
 67 perfect information, 1109  
 68 perfect intervention, 179  
 69 perfect map, 163  
 70 period, 53  
 71 periodic kernel, 665, 678  
 72 permuted MNIST, 765  
 73 perplexity, 201, 784  
 74 persistent contrastive divergence, 869  
 75 persistent variational inference, 159  
 76 personalized recommendations, 69  
 77 perturbation, 223  
 78 PETs, 1150  
 79 PGD, 771  
 80 PGMs, 125  
 81 phase space, 488  
 82 phase transition, 150  
 83 phi-exponential family, 34  
 84 phone, 991  
 85 Picard–Lindelöf theorem, 859  
 86 pictorial structure, 723  
 87 PILCO, 1149  
 88 Pilot Studies, 1093  
 89 pinball loss, 567  
 90 pipe, 131  
 91 pipeline, 719  
 92 Fitman-Koopman-Darmois theorem, 37, 204  
 93 pix2pix, 918  
 94 pixelCNN, 837  
 95 pixelCNN++, 838  
 96 pixelRNN, 838  
 97 PixelSNAIL, 827  
 98 placebo, 1113  
 99 planar flow, 858  
 100 PlaNet, 1152  
 101 planning, 1124, 1129  
 102 planning horizon, 1147  
 103 plant, 1121  
 104 plates, 144  
 105 Platt scaling, 560  
 106 platykurtic, 11  
 107 PLS, 944  
 108 plug-in approximation, 74  
 109 plug-in estimator, 1154  
 110 plugin approximation, 66  
 111 plutocratic, 15  
 112 PMMH, 549  
 113 PoE, 864  
 114 point estimate, 63, 66  
 115 point process, 1055  
 116 Poisson, 6  
 117 Poisson process, 1055  
 118 Poisson regression, 573  
 119 policy, 1111, 1120  
 120 policy evaluation, 1123, 1126  
 121 policy gradient, 1130  
 122 policy gradient theorem, 1141  
 123 policy improvement, 1126  
 124 policy iteration, 1126  
 125 policy optimization, 1123  
 126 policy search, 1130, 1140  
 127 Polya's urn, 8  
 128 Polyak-Ruppert averaging, 633  
 129 polymatroid function, 283  
 130 polynomial kernel, 665  
 131 polynomial regression, 581  
 132 polytree, 370  
 133 POMDP, 1121  
 134 pooled, 101  
 135 pooling, 619  
 136 pooling layer, 606  
 137 population shift, 742  
 138 POS, 719  
 139 position-specific scoring matrix, 977  
 140 positional encoding, 618  
 141 positive definite, 24  
 142 positive definite kernel, 661  
 143 positive phase, 159  
 144 possible worlds, 174  
 145 posterior collapse, 416, 815  
 146 posterior distribution, 64  
 147 posterior expected loss, 121  
 148 posterior inference, 64, 317  
 149 posterior marginal, 136  
 150 posterior mean, 123  
 151 posterior predictive check, 118  
 152 posterior predictive distribution, 66, 74  
 153 posterior-predictive p-value, 118  
 154 potential, 21  
 155 potential energy, 489  
 156 potential function, 146  
 157 potential outcome, 182  
 158 Potts model, 151  
 159 Potts models, 477  
 160 power EP, 449  
 161 power law, 14  
 162 power posterior, 629  
 163 PPCA, 939  
 164 PPL, 177  
 165 PPO, 1145  
 166 pre-train and fine-tune, 755  
 167 precision, 8, 80  
 168 precision matrix, 17, 33  
 169 preconditioned SGLD, 509  
 170 predict-update cycle, 324  
 171 prediction, 183  
 172 prediction step, 322  
 173 predictive coding, 340, 1163  
 174 predictive distribution, 321  
 175 predictive entropy search, 269  
 176 predictive model, 555  
 177 predictive sparse decomposition, 964  
 178 predictive state representation, 984  
 179 predictive uncertainty, 66  
 180 preferences, 122  
 181 prequential analysis, 113  
 182 prequential inference, 767  
 183 prescribed probabilistic models, 891  
 184 prevalence shift, 743  
 185 Price's theorem, 237, 244  
 186 primitive nodes, 228  
 187 primitive operations, 226  
 188 principle of insufficient reason, 95  
 189 prior distribution, 63  
 190 prior predictive distribution, 581  
 191 prior shift, 743  
 192 prioritized experience replay, 1139  
 193 Probabilistic backpropagation, 653  
 194 probabilistic backpropagation, 450, 632  
 195 probabilistic circuit, 172  
 196 probabilistic ensembles with trajectory sampling, 1150  
 197 Probabilistic graphical models, 125  
 198 probabilistic graphical models, 777  
 199 probabilistic principal components analysis, 939  
 200 probabilistic programming language, 177  
 201 probabilistic programming systems, 549  
 202 probability integral transform, 44  
 203 probability matching, 1117  
 204 probability of improvement, 267  
 205 probability simplex, 27  
 206 probit approximation, 591  
 207 probit function, 594  
 208 probit link function, 574  
 209 probit regression, 594  
 210 procedural approach, 177  
 211 process noise, 650, 1000, 1003  
 212 product density function, 1063  
 213 product of experts, 152, 808, 864  
 214 product partition model, 989  
 215 production rules, 720  
 216 profile HMM, 977  
 217 projected gradient descent, 771  
 218 projecting, 361  
 219 projection, 166  
 220 projection pursuit, 963  
 221 prompt, 755  
 222 propensity score, 1182  
 223 propensity score matching, 1187  
 224 proper scoring rule, 558, 895  
 225 Properties., 1070  
 226 Prophet, 731  
 227 proposal distribution, 311, 456, 458, 468, 475  
 228 propose, 468  
 229 protein sequence alignment, 976  
 230 protein-protein interaction networks, 1023  
 231 prototypical networks, 762  
 232 proximal policy optimization, 1145  
 233 proximal update, 240  
 234 pseudo counts, 66, 71  
 235 pseudo inputs, 684  
 236 pseudo likelihood, 160, 161  
 237 pseudo marginal, 1016  
 238 pseudo random number generator, 454  
 239 pseudo-marginal methods, 548  
 240 pseudo-observations, 433  
 241 PSIS, 115  
 242 pure exploration, 1119  
 243 pushforward, 845  
 244 pushing sums inside products, 385  
 245 Pyro, 177  
 246 Q-learning, 1130, 1137  
 247 QALY, 122  
 248 QKF, 359  
 249 QT-Opt, 1124  
 250 quadratic approximation, 307  
 251 quadratic kernel, 668  
 252 quadratic loss, 123  
 253 quadrature, 451  
 254 quadrature EP, 366

- 1 quadrature Kalman filter, 359  
 2 Qualitative Studies, 1092  
 3 quality-adjusted life years, 122  
 4 quantile loss, 567  
 5 quantile regression, 567, 734  
 6 quantization, 198  
 7 Quasi Monte Carlo, 465  
 8 queries, 264  
 9 query, 608  
 10 query nodes, 136  
 11 R-hat, 502  
 12 radar, 1011  
 13 radial basis function kernel, 662  
 14 radon, 599  
 15 rainbow, 1140, 1146  
 16 random accelerations model, 1000  
 17 random assignment with non-compliance, 1192  
 18 random effects, 599  
 19 random finite sets, 1012  
 20 random Fourier features, 643, 670  
 21 random measure, 1050  
 22 random walk kernel, 667  
 23 random walk Metropolis, 311, 471  
 24 random walk on the integers, 53  
 25 random walk proposal, 475  
 26 randomized control trials, 1176  
 27 Randomized QMC, 465  
 28 Rao-Blackwellised particle filtering, 537  
 29 rare event, 547  
 30 raster scan, 837  
 31 rate, 211  
 32 rate distortion curve, 212, 803  
 33 rational quadratic, 665, 733  
 34 rats, 103  
 35 RBM, 152  
 36 RBPF, 537  
 37 Real NVP, 860  
 38 real-time dynamic programming, 1126  
 39 receding horizon control, 1147  
 40 recognition network, 429, 791  
 41 recommendation weights, 960  
 42 recommender system, 174  
 43 reconstruction error, 211, 751  
 44 record linkage, 175  
 45 Rectified linear unit, 605  
 46 recurrent, 53  
 47 recurrent layer, 611  
 48 recurrent neural network, 613, 835  
 49 recurrent neural networks, 611  
 50 recurrent SSM, 1018  
 51 recursive, 177  
 52 recursive least squares, 338, 576, 650,  
 53 1003  
 54 recursive nested particle filter, 548  
 55 redundancy, 206, 214  
 56 reference prior, 100  
 57 refractoriness, 1059  
 58 regime switching Markov model, 970  
 59 Regression discontinuity designs, 1215  
 60 regression estimator, 1154  
 61 regression model, 555  
 62 regressive tax, 415  
 63 regret, 768, 1109, 1118, 1119  
 64 regular, 36, 53  
 65 regularization methods, 767  
 66 rehearsal, 767  
 67 REINFORCE, 418, 1130, 1141  
 68 Reinforcement learning, 1129  
 69 reject action, 123  
 70 rejection sampling, 456  
 71 relational data, 1027  
 72 relational probability models, 173  
 73 relational uncertainty, 174  
 74 relative entropy, 187  
 75 relative risk, 680  
 76 relevance network, 1031  
 77 relevance vector machine, 587  
 78 reliability diagram, 559  
 79 Renewal processes, 1058  
 80 reparameterization gradient, 243  
 81 reparameterization trick, 421, 631, 795  
 82 reparameterized VI, 421  
 83 repeated trials, 63  
 84 representation, 215  
 85 Representation learning, 783  
 86 representation learning, 693  
 87 representer theorem, 677  
 88 Reproducing Kernel Hilbert Space, 676  
 89 reproducing property, 676  
 90 resample, 522  
 91 residual, 334  
 92 residual belief propagation, 380  
 93 residual block, 857  
 94 residual connections, 606, 857  
 95 residual flows, 857  
 96 ResNet, 613  
 97 resource allocation, 1113  
 98 response surface model, 264  
 99 responsibility, 929  
 100 restless bandit, 1113  
 101 restricted Boltzmann machine, 152  
 102 return, 1122  
 103 reverse process, 883  
 104 reverse-mode automatic differentiation, 227  
 105 reversible jump MCMC, 513, 951  
 106 reward, 1106, 1111  
 107 reward function, 1120  
 108 reward model, 1120  
 109 reward-to-go, 1122  
 110 reweighted wake sleep, 831, 832  
 111 RFF, 670  
 112 Riccati equations, 335  
 113 rich get richer, 8, 415, 1038  
 114 ridge regression, 577, 579, 675  
 115 ridgeless regression, 710  
 116 Riemann Manifold HMC, 493  
 117 Riemannian manifold, 233  
 118 risk, 121  
 119 RJMCMC, 513  
 120 RKHS, 676  
 121 RL, 1129  
 122 RLS, 338, 1003  
 123 RMSprop, 263, 263  
 124 RNADE, 836  
 125 RNN, 613  
 126 RNN-HSMM, 987  
 127 robust optimization, 772  
 128 robust priors, 93  
 129 robustness analysis, 93  
 130 roll call, 956  
 131 row stochastic matrix, 127, 967  
 132 RPMs, 173  
 133 RStanARM, 599  
 134 RTS smoother, 342  
 135 Rubin Causal Model, 183  
 136 run length, 988  
 137 running intersection property, 391  
 138 Russian roulette estimator, 858  
 139 RVI, 421  
 140 SAC, 1162  
 141 safe policy iteration, 1144  
 142 SAGA-LD, 511  
 143 SAGAN, 913  
 144 sample diversity, 783  
 145 sample inefficient, 1131  
 146 sample quality, 783  
 147 sample size, 70  
 148 sample standard deviation, 84  
 149 sampling distribution, 63, 579  
 150 sampling with replacement, 7  
 151 sampling without replacement, 8  
 152 sandwich estimate, 443  
 153 SARS-CoV2, 356  
 154 SARSA, 1130, 1136  
 155 satisfying assignment, 389  
 156 SBED, 1160  
 157 scale-invariant prior, 99  
 158 scaled inverse chi-squared, 14  
 159 scaling-binning calibrator, 560  
 160 SCFGs, 992  
 161 SCM, 177  
 162 SCMC, 546  
 163 scope, 135  
 164 score, 870  
 165 score function, 39, 41, 866  
 166 score function estimator, 242, 418, 921  
 167 score matching, 870, 888  
 168 score-based generative models, 874  
 169 seasonality, 727  
 170 second order stationary, 505  
 171 second-order delta method, 245  
 172 segment, 317  
 173 segmental HMM, 987  
 174 selection bias, 134, 744  
 175 selective prediction, 751  
 176 self attention, 838  
 177 Self Attention GAN, 913  
 178 self-attention, 610  
 179 self-normalized importance sampling, 459  
 180 semantic network, 1027  
 181 semantic segmentation, 722  
 182 semi-amortized VI, 430  
 183 semi-Markov model, 986  
 184 semi-parametric model, 673  
 185 semi-parametrically efficient, 1183  
 186 semi-supervised learning, 811  
 187 semipartial linear trend, 726  
 188 semivariogram, 505  
 189 sensible PCA, 939  
 190 sensitivity analysis, 93  
 191 sensor fusion, 19  
 192 separator, 392  
 193 sequence memoizer, 1047  
 194 sequential Bayesian inference, 311, 650  
 195 sequential Bayesian updating, 323, 338,  
 196 1003  
 197 sequential decision problem, 1111  
 198 sequential decision problems, 123  
 199 sequential importance sampling, 520  
 200 sequential importance sampling with re-  
 201 sampling, 521  
 202 sequential model-based optimization,  
 203 265  
 204 sequential Monte Carlo, 311, 517  
 205 sequential VAE, 812  
 206 sFA, 956  
 207 SFE, 242  
 208 SG-HMC, 511  
 209 SGD, 232  
 210 SGLD, 494, 509  
 211 SGLD-Adam, 509  
 212 SGLD-CV, 510  
 213 SGPR, 689  
 214 SGRLD, 509  
 215 shaded nodes, 144  
 216 Shafer-Shenoy, 395  
 217 sharp minima, 639  
 218 shift equivariance, 606  
 219 shift invariance, 606  
 220 shift-invariant kernel, 668  
 221 shooting, 1148  
 222 shortcut features, 740  
 223 shortest path problems, 1125  
 224 shrinkage, 81, 103  
 225 sigma point BP, 377  
 226 sigma points, 351  
 227 sigma-VAE, 802  
 228 sigmoid, 588  
 229 sigmoid belief net, 130  
 230 silent state, 975  
 231 sim2real gap, 747  
 232 simple regret, 1118  
 233 simplex factor analysis, 956  
 234 Simplicity., 1085  
 235 Simulability, 1083  
 236 Simulated annealing, 473  
 237 simulated annealing, 264  
 238 simulation-based inference, 547, 892  
 239 simultaneous localization and mapping,  
 240 1007  
 241 single site updating, 481  
 242 single world intervention graph, 183  
 243 singular statistical model, 117  
 244 SIS, 520  
 245 SISR, 521  
 246 site parameters, 432  
 247 site potential, 447  
 248 sketch-rnn, 814  
 249 SKI, 691, 692  
 250 SKIP, 692  
 251 skip connections, 606, 817, 819  
 252 skip-chain CRF, 720  
 253 skip-VAE, 817  
 254 SLAM, 539, 1007  
 255 SLDS, 362  
 256 SLDS), 537  
 257 sleep phase, 832  
 258 slice sampling, 486  
 259 sliced Fisher divergence, 872

- 1
- Sliced Score Matching, 872  
sliding window detector, 722  
slippage, 1008  
slot machines, 1111  
slow weights, 637  
SMBO, 265  
SMC, 311, 517  
SMC sampler, 517  
SMC samplers, 541  
 $\text{SMC}^2$ , 548  
SMC-ABC, 548  
SMILES, 815  
smoothed Bellman error embedding, 1160  
snapshot ensembles, 633  
SNGP, 632  
Sobol sequence, 465  
social networks, 1023  
soft actor-critic, 1162  
soft clustering, 929, 1025  
soft constraint, 864  
soft Q-learning, 1162  
soft-thresholding, 415  
softmax, 31  
softmax function, 588  
Softplus, 605  
SOLA, 437  
SOR, 685  
Soundness, 1084  
source coding, 185, 210  
source coding theorem, 210  
source distributions, 754  
source domain, 918  
space filling, 465  
SPADA, 1011  
sparse, 27, 582  
sparse Bayesian learning, 585  
sparse coding, 964  
sparse factor analysis, 939  
sparse GP, 684  
sparse GP regression, 689  
sparse variational GP, 687  
sparsity, 632  
sparsity promoting priors, 627  
spatial statistics community, 505  
spectral density, 528, 669, 701, 1001  
spectral estimation, 984  
spectral estimation method, 1006  
spectral mixture kernel, 701  
spectral mixture kernels, 669  
spelling correction, 973  
sphere the data, 962  
spherical covariance matrix, 17  
spherical cubature integration, 359  
spherical K-means algorithm, 24  
spherical topic model, 24  
spherling, 960  
spike and slab, 931  
spike-and-slab, 582  
spin, 148  
splines, 850  
split conformal prediction, 569  
split MNIST, 765  
split-Rhat, 502  
splitting, 512  
spring mass system, 998  
spurious correlation, 740  
SQF-RNN, 734  
square root filter, 337  
square root information filter, 337  
square-integrable functions, 676  
squared error, 123  
squared exponential, 662  
squared exponential (SE) kernel, 662  
SS, 582  
SSID, 1005  
SSM, 316, 725, 997  
Stability, 1082  
stacking, 636  
standard Brownian motion, 1057  
standard error, 66, 73, 453  
standard error of the mean, 85  
standard Normal, 8  
state of nature, 121  
state space model, 316, 997  
state transition diagram, 47  
state-space models, 967  
state-value function, 1122  
stateful, 611
- static calibration error, 560  
stationary, 46  
stationary distribution, 51  
stationary kernels, 662  
statistical estimand, 1173  
Statistics, 63  
steepest descent, 231  
stepping out, 486  
stepwise EM, 257  
stick-breaking construction, 1036  
stick-breaking process, 1037  
sticking the landing, 422  
sticky, 471  
stochastic approximation, 256, 264  
stochastic approximation EM, 256  
stochastic automaton, 47  
stochastic averaged gradient acceleration, 511  
stochastic bandit, 1111  
stochastic block model, 1024  
stochastic computation graph, 246  
stochastic context free grammars, 992  
stochastic EP, 450  
stochastic gradient descent, 232  
stochastic gradient Langevin dynamics, 509  
Stochastic Gradient Riemannian Langevin Dynamics, 509  
stochastic Lanczos quadrature, 691  
stochastic local search, 299, 473  
stochastic matrix, 47  
stochastic meta descent, 724  
stochastic process, 1033  
stochastic relaxation, 264  
stochastic RNN, 1017  
stochastic variance reduced gradient, 510  
stochastic variational inference, 310, 421, 690  
stochastic video generation, 1021  
stochastic volatility model, 1009  
stochastic weight averaging, 633  
stochastically ordered, 1037  
stop gradient, 826  
straight-through estimator, 246, 825  
stratified resampling, 525  
streaks, 317  
streaming variational Bayes, 434  
strict overlap, 1180  
strictly monotonic scalar function, 849  
string kernel, 667  
structural causal models, 177, 179, 1170  
structural equation model, 167, 179  
structural time series, 725, 1004  
structural zeros, 155  
structured conjugate models, 397  
structured kernel interpolation, 691  
structured mean field, 438  
structured prediction, 717  
Structured Prediction Energy Networks, 724  
STS, 725  
Student distribution, 9  
Student network, 127  
student network, 128, 164, 385  
Student t distribution, 9  
style transfer, 919  
sub-Gaussian, 11  
subjective probability, 125  
Submodular, 282  
subphones, 991  
Subscale Pixel Network, 838  
subset of regressors, 685  
subspace identification, 1005  
subspace neural bandit, 1115  
sufficient, 215  
sufficient statistic, 46, 204  
sufficient statistics, 29, 29, 70  
sum-product belief propagation, 375  
sum-product networks, 172  
super-Gaussian, 11  
supervised PCA, 943  
supply chain, 1015  
surrogate function, 247, 264  
survival of the fittest, 523  
susbet-of-data, 682  
SUTVA, 183  
SVB, 434  
SVG, 1021  
SVGP, 687
- SVI, 421  
SVRG-LD, 510  
SWA, 633  
SWAG, 634  
Swendsen Wang, 487  
SWIG, 183  
Swish, 605  
swiss roll, 874  
switching linear dynamical system, 362, 537  
Sylvester flow, 858  
symamd, 394  
symmetric, 468  
symplectic, 490  
synchronous updates, 380  
synergy, 206  
syntactic sugar, 144  
synthetic control, 736  
Synthetic controls, 1215  
systems biology, 1029  
systems identification, 1005  
systolic array, 373
- T5, 812  
tabular representation, 1121  
tacotron, 837  
TAN, 145  
target aware Bayesian inference, 460  
target distribution, 455, 458, 754  
target domain, 918  
target function, 458  
target network, 1139  
target policy, 1154  
targeted attack, 770  
task, 764  
task incremental learning, 764  
task interference, 656  
task-aware continual learning, 765  
Taylor series, 308  
Taylor series expansion, 346  
TD, 1134  
TD error, 1130, 1134  
TD( $\lambda$ ), 1135  
TD3, 1146  
telescoping sum, 887  
temperature, 495  
temperature scaling, 560  
tempered posterior, 629  
tempering, 437  
template, 174  
templates, 934  
temporal difference, 1130, 1134  
tensor decomposition, 984  
tensor train decomposition, 692  
terminal state, 1122  
terminals, 720  
test and roll, 1106  
test statistics, 118  
test-time training, 753  
text to speech, 837  
text-to-speech, 920  
the deadly triad, 1159  
thermodynamic integration, 442  
thermodynamic variational objective, 442
- thin junction tree filter, 1009  
thin shell, 200  
thinning theorem, 1057  
Thompson sampling, 268, 1117  
threat model, 772  
tilted distribution, 447  
time reversal, 542  
time reversible, 54  
time series forecasting, 725, 1004  
time update step, 333  
time-invariant, 46  
time-series forecasting, 167  
Toeplitz, 692  
top-down inference model, 819  
topic, 1047  
topological order, 136  
topological ordering, 125  
total correlation, 205, 804  
total derivative, 244  
total regret, 1119  
total variation distance, 60  
trace plot, 499  
trace rank plot, 500
- 47

- 1
- 2 traceback, 330, 376  
track, 363  
3 tracking, 318  
tracking by detection, 529  
4 tractable substructure, 438  
trajectory, 1129  
5 trunkplot, 500  
trans-dimensional MCMC, 513  
6 transductive active learning, 564  
transductive learning, 749  
7 transfer learning, 744, 755  
transformer, 615, 615  
8 transformer VAE, 812  
Transformer-XL, 622  
9 transformers, 812  
transient, 53  
10 transition, 1120  
transition function, 46  
11 transition kernel, 46  
transition matrix, 47, 47  
12 transition model, 967, 1120  
translation invariant, 965  
13 translation-invariant prior, 99  
Translucence, 1083  
14 Transparency, 1080  
transportable, 741  
15 treatment, 180  
treatment, 1106  
16 treatment effect, 736  
tree decomposition, 391  
17 tree-augmented naive Bayes classifier, 145  
18 treewidth, 388, 392  
trellis diagram, 329  
19 triangulated, 392  
triangulation, 392  
20 triggering kernel, 1059  
trigram model, 48  
21 TRPO, 1144  
TrueSkill, 450  
22 truncated Gaussian, 596  
trust region policy optimization, 1144  
23 TT-GP, 692  
turbo codes, 382  
24 Turing, 177  
turning the Bayesian crank, 303  
25 TVO, 442  
twin network, 183  
26 two part code, 213  
two sample tests, 788  
27 two stage least squares, 1198  
two-filter smoothing, 327, 343  
28 two-moons, 655  
two-sample test, 54  
29 two-sample testing, 748  
two-slice marginal, 327  
30 type II maximum likelihood, 106  
type signature, 173  
31 typical set, 199, 200
- 32 UCB, 268, 1115  
UCBVI, 1132  
UCRL2, 1132  
33 UGM, 146  
34 UKF, 351  
35 ULA, 494  
36 ULD, 511  
37 UME, 59  
Unadjusted Langevin Algorithm, 494  
unary terms, 150, 155  
38 unbiased, 1155  
uncertainty metric, 749  
39 uncertainty quantification, 565  
unclamped phase, 159, 867  
unconstrained monotonic neural networks, 849  
40 underdamped Langevin dynamics, 511  
41 underlying predictive model, 988  
undirected graphical model, 146  
42 undirected local Markov property, 156  
43  
44  
45  
46  
47
- unidentifiable, 66  
Unified Medical Language System, 1027  
uniform dequantization, 785  
unigram model, 48  
unigram statistics, 50  
uninformative, 72, 95  
uninformative prior, 578  
units, 182  
unnormalized mean embedding, 59  
unnormalized target distribution, 459  
unnormalized weights, 460, 520  
unpaired data, 918  
unroll, 174  
unrolled, 144  
unscented Kalman filter, 351, 1006  
unscented Kalman filtering, 377  
unscented particle filter, 534  
unscented transform, 351  
unsupervised domain translation, 918  
untargeted attack, 770  
update, 620  
update step, 322  
UPM, 988  
upper confidence bound, 268, 1115  
user rating profile model, 956  
User studies, 1086  
utility function, 121  
utility nodes, 1103
- v-structure, 131, 132  
VAE, 154, 777, 791  
VAE-RNN, 812  
VAFC, 424  
vague prior, 581  
validation set, 113  
value function, 1122  
value iteration, 1125  
value nodes, 1103  
value of perfect information, 1106  
ValueDICE, 1165  
values, 608  
VAR, 167  
variable binding problem, 842  
variable duration HMM, 986  
variable elimination, 385, 1106  
Variational Approximation with Factor Covariance, 424  
variational autoencoder, 791  
variational autoencoders, 777, 947  
variational Bayes, 308, 406  
variational Bayes EM, 411  
variational continual learning, 434, 654, 767  
variational EM, 251, 256, 411  
variational free energy, 340, 402, 687, 794, 1163  
variational gap, 793  
variational GP, 438  
variational IB, 216  
variational inference, 29, 255, 308, 401, 557, 785  
variational message passing, 260, 417  
variational method, 401  
Variational Online Gauss-Newton, 262, 263  
variational online Gauss-Newton, 631  
Variational Online Generalized Gauss-Newton, 263  
variational optimization, 236, 264  
variational over-pruning, 632  
variational overpruning, 436, 815  
variational parameters, 308, 401  
variational pruning, 823  
variational pruning effect, 416  
variational RNN, 1020  
variational SMC, 536  
varimax, 939  
variogram, 505  
VB, 308  
VCL, 434, 654  
VD-VAE, 820
- vector auto-regressive, 167  
vector quantization, 210  
vector-Jacobian product (VJP), 223  
VERSA, 762  
very deep VAE, 820  
VFE, 687  
VI, 308  
VIB, 216  
VIM, 830  
VIREL, 1162  
virtual samples, 91  
visible nodes, 136  
vision as inverse graphics, 925  
visual clutter, 529  
visual SLAM, 1007  
visual tracking, 529  
Visualization, 1080  
Viterbi algorithm, 321, 329, 977  
Viterbi training, 980  
VMP, 417  
VOGNN, 263  
VOGN, 262, 263, 631  
von Mises, 24  
von Mises-Fisher, 24  
VQ-GAN, 829  
VQ-VAE, 824  
VRNN, 1020
- wake phase, 831  
wake sleep, 792  
wake-phase  $q$  update, 832  
wake-sleep algorithm, 831  
warmup, 492  
Wasserstein-1 distance, 57  
Watanabe-Akaike information criterion, 117  
wavenet, 837  
weak marginalization, 364  
weak prior, 581  
weakly informative, 94  
wealth, 15  
website testing, 1110  
weight degeneracy, 521  
weight of evidence, 191  
weight perturbation, 263  
weight space, 672  
weighted conformal prediction, 745  
weighted least squares, 255  
Weiner process, 1056  
Weinstein–Aronszajn identity, 858  
well-log dataset, 989  
white noise kernel, 678  
white noise process, 505, 1000  
whitebox attack, 770  
whitened coordinate system, 233  
whitening, 960  
widely applicable information criterion, 117  
width, 391  
Wiener noise, 509, 512  
wildcatter, 1103  
Wishart, 24  
witness function, 57  
word error, 376  
word2vec, 782
- Xavier initialization, 626  
XMC-GAN, 842
- z-bias, 1208  
zero-avoiding, 191  
zero-forcing, 191, 445  
zero-inflated Poisson, 573, 1015  
zero-one loss, 123  
zero-shot learning, 756  
zero-sum losses, 903  
ZIP, 573, 1015  
Zipf's law, 15

# Bibliography

- [AA18] D. Amir and O. Amir. "Highlights: Summarizing agent behavior to people". In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 2018, pp. 1168–1176.
- [AB08] C. Archambeau and F. Bach. "Sparse probabilistic projections". In: *NIPS*. 2008.
- [AB17] M. Arjovsky and L. Bottou. "Towards principled methods for training generative adversarial networks". In: (2017).
- [AB21] A. N. Angelopoulos and S. Bates. "A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification". In: (July 2021). arXiv: [2107.07511 \[cs.LG\]](https://arxiv.org/abs/2107.07511).
- [Abadie] A. Abadie. "Using Synthetic Controls: Feasibility, Data Requirements, and Methodological Aspects". In: *J. of Economic Literature* () .
- [Abd+18] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. A. Riedmiller. "Maximum a Posteriori Policy Optimisation". In: *ICLR*. 2018.
- [ABM10] J.-Y. Audibert, S. Bubeck, and R. Munos. "Best Arm Identification in Multi-Armed Bandits". In: *COLT*. 2010, pp. 41–53.
- [ABV21] S. Akbayrak, I. Bocharov, and B. de Vries. "Extended Variational Message Passing for Automated Approximate Bayesian Inference". In: *Entropy* 23 (July 2021).
- [AC17] S. Aminikhanghahi and D. J. Cook. "A Survey of Methods for Time Series Change Point Detection". en. In: *Knowl. Inf. Syst.* 51.2 (May 2017), pp. 339–367.
- [AC93] J. Albert and S. Chib. "Bayesian analysis of binary and polychotomous response data". In: *JASA* 88.422 (1993), pp. 669–679.
- [ACB17] M. Arjovsky, S. Chintala, and L. Bottou. "Wasserstein generative adversarial networks". In: *ICML*. 2017, pp. 214–223.
- [ACL16] L. Aitchison, N. Corradi, and P. E. Latham. "Zipf's Law Arises Naturally When There Are Underlying, Unobserved Variables". en. In: *PLoS Comput. Biol.* 12.12 (Dec. 2016), e1005110.
- [ACP87] S. Arnborg, D. G. Corneil, and A. Proskurowski. "Complexity of finding embeddings in a k-tree". In: *SIAM J. on Algebraic and Discrete Methods* 8 (1987), pp. 277–284.
- [Ada00] L. Adamic. *Zipf, Power-laws, and Pareto - a ranking tutorial*. Tech. rep. 2000.
- [Ada+20] V. Adam, S. Eleftheriadis, N. Durrande, A. Artemev, and J. Hensman. "Doubly Sparse Variational Gaussian Processes". In: *AISTATS*. 2020.
- [Ade+20a] J. Adebayo, J. Gilmer, M. Muell, I. Goodfellow, M. Hardt, and B. Kim. "Sanity Checks for Saliency Maps". 2020. arXiv: [1810.03292 \[cs.CV\]](https://arxiv.org/abs/1810.03292).
- [Ade+20b] J. Adebayo, M. Muell, I. Liccardi, and B. Kim. "Debugging tests for model explanations". In: *arXiv preprint arXiv:2011.05429* (2020).
- [ADH10] C. Andrieu, A. Doucet, and R. Holenstein. "Particle Markov chain Monte Carlo methods". en. In: *J. of Royal Stat. Soc. Series B* 72.3 (June 2010), pp. 269–342.
- [Adl1] "Understanding Double Descent Requires a Fine-Grained Bias-Variance Decomposition". In: *ICML*. 2020.
- [Adl+18] P. Adler, C. Falk, S. A. Friedler, T. Nix, G. Rybeck, C. Scheidegger, B. Smith, and S. Venkatasubramanian. "Auditing black-box models for indirect influence". In: *Knowledge and Information Systems* 54.1 (2018), pp. 95–122.
- [Ado1] "Taking It to the MAX: Adobe Photoshop Gets New NVIDIA AI-Powered Neural Filters". <https://blogs.nvidia.com/blog/2020/10/20/adobe-max-ai/>. Accessed: 2021-08-12.
- [AE+20] D. Agudelo-España, S. Gomez-Gonzalez, S. Bauer, B. Schölkopf, and J. Peters. "Bayesian Online Prediction of Change Points". In: *UAI*. Vol. 124. Proceedings of Machine Learning Research. PMLR, 2020, pp. 320–329.
- [AFD01] C. Andrieu, N. de Freitas, and A. Doucet. "Robust Full Bayesian Learning for Radial Basis Networks". In: *Neural Computation* 13.10 (2001), pp. 2359–2407.
- [AFG19] M. Akten, R. Fiebrink, and M. Grierson. "Learning to See, You Are What You See". In: *ACM SIGGRAPH 2019 Art Gallery*. SIGGRAPH '19. Los Angeles, California: Association for Computing Machinery, 2019.
- [AG11] A. Allahverdyan and A. Galstyan. "Comparative Analysis of Viterbi Training and Maximum Likelihood Estimation for HMMs". In: *NIPS*. 2011, pp. 1674–1682.
- [AG13] S. Agrawal and N. Goyal. "Further Optimal Regret Bounds for Thompson Sampling". In: *AISTATS*. 2013.
- [Aga+14] D. Agarwal, B. Long, J. Traupman, D. Xin, and L. Zhang. "LASER: a scalable response prediction platform for online advertising". In: *WSDM*. 2014.
- [Aga+21a] A. Agarwal, N. Jiang, S. M. Kakade, and W. Sun. *Reinforcement Learning: Theory and Algorithms*. 2021.
- [Aga+21b] R. Agarwal, L. Melnick, N. Frossat, X. Zhang, B. Lengerich, R. Caruana, and G. Hinton. *Neural Additive Models: Interpretable Machine Learning with Neural Nets*. 2021. arXiv: [2004.13912 \[cs.LG\]](https://arxiv.org/abs/2004.13912).
- [AH09] I. Arasaratnam and S. Haykin. "Cubature Kalman Filters". In: *IEEE Trans. Automat. Contr.* 54.6 (June 2009), pp. 1254–1269.
- [AHE07] I. Arasaratnam, S. Haykin, and R. J. Elliott. "Discrete-Time Nonlinear Filtering Algorithms Using Gauss-Hermite Quadrature". In: *Proc. IEEE* 95.5 (May 2007), pp. 953–977.
- [AHG20] L. Ambrogioni, M. Hinne, and M. van Gerven. "Automatic structured variational inference". In: (Feb. 2020). arXiv: [2002.00643 \[stat.ML\]](https://arxiv.org/abs/2002.00643).
- [AHK01] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. "On the Surprising Behavior of Distance Metrics in High Dimensional Space". In: *Database Theory — ICDT 2001*. Springer Berlin Heidelberg, 2001, pp. 420–434.
- [AHK12] A. Anandkumar, D. Hsu, and S. M. Kakade. "A Method of Moments for Mixture Models and Hidden Markov Models". In: *COLT*. Vol. 23. Proceedings of Machine Learning Research. Edinburgh, Scotland: PMLR, 2012, pp. 33.1–33.34.
- [AHK65] K. Abend, T. J. Harley, and L. N. Kanal. "Classification of Binary Random Patterns". In: *IEEE Transactions on Information Theory* 11(4) (1965), pp. 538–544.
- [Ahm+17] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha. "Unsupervised real-time anomaly detection for streaming data". In: *Neurocomputing* 262 (Nov. 2017), pp. 134–147.
- [AHP+05] P. K. Agarwal, S. Har-Peled, et al. "Geometric approximation via coresets". In: *Combinatorial and computational geometry* 52.1-30 (2005), p. 3.
- [AHSS85] D. Ackley, G. Hinton, and T. Sejnowski. "A Learning Algorithm for Boltzmann Machines". In: *Cognitive Science* 9 (1985), pp. 147–169.
- [AHT07] Y. Altun, T. Hofmann, and I. Tsacharidis. "Support Vector Machine Learning for Interdependent and Structured Output Spaces". In: *Predicting Structured Data*. Ed. by G. Bakir, T. Hofmann, B. Schölkopf, A. Smola, B. Taskar, and S. Vishwanathan. MIT Press, 2007.
- [Ahu+21] K. Ahuja, J. Wang, A. Dhurandhar, K. Shanmugam, and K. R. Varshney. "Empirical or Invariant Risk Minimization? A Sample Complexity Perspective". In: *ICLR*. 2021.
- [AI19] AI Artists. *Creative Tools to Generate AI Art*. 2019.
- [Air+08] E. Airolid, D. Blei, S. Fienberg, and E. Xing. "Mixed-membership stochastic blockmodels". In: *JMLR* 9 (2008), pp. 1981–2014.
- [Ait18] L. Aitchison. "A unified theory of adaptive stochastic gradient descent as Bayesian filtering". In: (July 2018). arXiv: [1807.07540 \[stat.ML\]](https://arxiv.org/abs/1807.07540).
- [Ait20] L. Aitchison. "Why bigger is not always better: on finite and infinite neural networks". In: *ICML*. 2020.
- [Ait21] L. Aitchison. "A statistical theory of cold posteriors in deep neural networks". In: *ICLR*. 2021.
- [Aka74] H. Akaike. "A new look at the statistical model identification". In: *IEEE Trans. on Automatic Control* 19.6 (1974).
- [AKO18] S.-I. Amari, R. Karakida, and M. Oizumi. "Fisher Information and Natural Gradient Learning of Random Deep Networks". In: (Aug. 2018). arXiv: [1808.07172 \[cs.LG\]](https://arxiv.org/abs/1808.07172).
- [Ale+16] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy. "Deep Variational Information Bottleneck". In: *ICLR*. 2016.
- [Ale+18] A. A. Alemi, B. Poole, I. Fischer, J. V. Dillon, R. A. Saurous, and K. Murphy. "Fixing a broken ELBO". In: *ICML*. 2018.

- 1
- 2 [Alq21] P. Alquier. "User-friendly introduction to PAC-Bayes  
bounds". In: (Oct. 2021). arXiv: [2110.11216 \[stat.ML\]](https://arxiv.org/abs/2110.11216).
- 3 [Als+19] J. Alsing, T. Charnock, S. Feeney, and B. Wandelt.  
"Fast likelihood-free cosmology with neural density estimators  
and active learning". In: *Monthly Notices of the Royal Astro-  
nomical Society* 488.3 (July 2019), pp. 4440–4458.
- 4 [ALS20] B. Axelrod, Y. P. Liu, and A. Sidford. "Near-optimal  
approximate discrete and continuous submodular function  
minimization". In: *Proceedings of the Fourteenth Annual  
ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2020,  
pp. 837–853.
- 5 [AM00] S. M. Aji and R. J. McEliece. "The Generalized Dis-  
tributive Law". In: *IEEE Trans. Info. Theory* 46.2 (2000),  
pp. 325–343.
- 6 [AM05] E. Amir and S. McIlraith. "Partition-Based Logical Rea-  
soning for First-Order and Propositional Theories". In: *Artifi-  
cial Intelligence* 162.1 (2005), pp. 49–88.
- 7 [AM07] R. P. Adams and D. J. C. MacKay. "Bayesian On-  
line Changepoint Detection". In: (Oct. 2007). arXiv: [0710.3742 \[stat.ML\]](https://arxiv.org/abs/0710.3742).
- 8 [AM+16] M. Auger-Méthé, C. Field, C. M. Albertsen, A. E.  
Derocher, M. A. Lewis, I. D. Jonsen, and J. Mills Flemming.  
"State-space models' dirty little secrets: even simple linear  
Gaussian models can have estimation problems". In: *Sci.  
Rep.* 6 (May 2016), p. 26977.
- 9 [AM74] D. Andrews and C. Mallows. "Scale mixtures of Normal  
distributions". In: *J. of Royal Stat. Soc. Series B* 36 (1974),  
pp. 99–102.
- 10 [AM89] B. D. Anderson and J. B. Moore. *Optimal Control: Lin-  
ear Quadratic Methods*. Prentice-Hall International, Inc., 1989.
- 11 [Ama09] S.-I. Amari. " $\alpha$ -Divergence Is Unique, Belonging to Both f-Divergence and Bregman Divergence Classes". In: *IEEE Trans. Inf. Theory* 55.11 (Nov. 2009), pp. 4925–4931.
- 12 [Ama98] S. Amari. "Natural Gradient Works Efficiently in Learn-  
ing". In: *Neural Comput.* 10.2 (Feb. 1998), pp. 251–276.
- 13 [Ame+19] S. Amershii, D. Weld, M. Vorvoreanu, A. Fournier, B.  
Nushi, P. Collisson, J. Suh, S. Igbal, P. N. Bennett, K. Inkpen,  
et al. "Guidelines for human-AI interaction". In: *Proceedings of  
the 2019 chi conference on human factors in computing sys-  
tems*. 2019, pp. 1–13.
- 14 [Ami01] E. Amir. "Efficient Approximation for Triangulation of  
Minimum Treewidth". In: *UAI*. 2001.
- 15 [AMJ18a] D. Alvarez-Melis and T. S. Jaakkola. "On the ro-  
bustness of interpretability methods". In: *arXiv preprint  
arXiv:1806.08049* (2018).
- 16 [AMJ18b] D. Alvarez-Melis and T. S. Jaakkola. "Towards robust  
interpretability with self-explaining neural networks". In: *arXiv  
preprint arXiv:1806.07538* (2018).
- 17 [Amo+16] D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano,  
J. Schulman, and D. Mané. "Concrete Problems in AI Safety".  
In: *CoRR* abs/1606.06565 (2016). arXiv: [1606.06565](https://arxiv.org/abs/1606.06565).
- 18 [Amo+18] B. Amos, L. Dinh, S. Cabri, T. Rothörl, S. G. Col-  
menarejo, A. Muldal, T. Erez, Y. Tassa, N. de Freitas, and M.  
Denil. "Learning Awareness Models". In: *ICLR*. 2018.
- 19 [Amo22] B. Amos. "Tutorial on amortized optimization for  
learning to optimize over continuous domains". In: (Feb. 2022).  
arXiv: [2202.00665 \[cs.LG\]](https://arxiv.org/abs/2202.00665).
- 20 [AMO88] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. "Net-  
work flows". In: (1988).
- 21 [Ana+14] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M.  
Telgarsky. "Tensor Decompositions for Learning Latent Vari-  
able Models". In: *JMLR* 15 (2014), pp. 2773–2832.
- 22 [And01] J. L. Anderson. "An Ensemble Adjustment Kalman Fil-  
ter for Data Assimilation". In: *Mon. Weather Rev.* 129.12 (Dec.  
2001), pp. 2884–2903.
- 23 [And+03] C. Andrieu, N. de Freitas, A. Doucet, and M. Jordan.  
"An introduction to MCMC for machine learning". In: *Machine  
Learning* 50 (2003), pp. 3–43.
- 24 [And+20] O. M. Andrychowicz et al. "Learning dexterous in-  
hand manipulation". In: *Int. J. Rob. Res.* 39.1 (Jan. 2020),  
pp. 3–20.
- 25 [Ang+18] E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer,  
and C. Rudin. *Learning Certifiably Optimal Rule Lists for Cat-  
egorical Data*. 2018. arXiv: [1704.01701 \[stat.ML\]](https://arxiv.org/abs/1704.01701).
- 26 [Ang+20] C. Angermueller, D. Dohan, D. Belanger, R. Deshpande,  
K. Murphy, and L. Colwell. "Model-based reinforcement  
learning for biological sequence design". In: *ICLR*. 2020.
- 27 [Ang+21] A. N. Angelopoulos, S. Bates, E. J. Candès, M. I.  
Jordan, and L. Lei. "Learn them Test: Calibrating Predictive  
Algorithms to Achieve Risk Control". In: (Oct. 2021). arXiv:  
[2110.01052 \[cs.LG\]](https://arxiv.org/abs/2110.01052).
- 28 [Ani+18] R. Anirudh, J. J. Thiagarajan, B. Kailkhura, and T.  
Bremer. "An Unsupervised Approach to Solving Inverse Prob-  
lems using Generative Adversarial Networks". In: (May 2018).  
arXiv: [1805.07281 \[cs.CV\]](https://arxiv.org/abs/1805.07281).
- 29 [Ang+19] Anonymous. "Neural Tangents: Fast and Easy Infinite  
Neural Networks in Python". In: (Sept. 2019).
- 30 [AO03] J.-H. Ahn and J.-H. Oh. "A Constrained EM Algorithm  
for Principal Component Analysis". In: *Neural Computation* 15  
(2003), pp. 57–65.
- 31 [AOM17] M. G. Azar, I. Osband, and R. Munos. "Minimax  
Regret Bounds for Reinforcement Learning". In: *ICML*. 2017,  
pp. 263–272.
- 32 [AP08] J. D. Angrist and J.-S. Pischke. *Mostly harmless econo-  
metrics: An empiricist's companion*. Princeton university  
press, 2008.
- 33 [AP09] J. Angrist and J.-S. Pischke. *Mostly Harmless Econo-  
metrics*. 2009.
- 34 [AP19] M. Abadi and G. D. Plotkin. "A simple differentiable  
programming language". In: *Proceedings of the ACM on Pro-  
gramming Languages* 4.POPL (2019), pp. 1–28.
- 35 [AR09] C. Andrieu and G. O. Roberts. "The pseudo-marginal  
approach for efficient Monte Carlo computations". en. In: *An-  
nals of Statistics* 37.2 (Apr. 2009), pp. 697–725.
- 36 [Ara+09] A. Aravkin, B. Bell, J. Burke, and G. Pillonetto. *An  
L1-Laplace Robust Kalman Smoother*. Tech. rep. U. Washingt-  
ton, 2009.
- 37 [Ara10] A. Aravkin. *Student's t Kalman Smoother*. Tech. rep.  
U. Washington, 2010.
- 38 [Ara+17] A. Aravkin, J. V. Burke, L. Ljung, A. Lozano, and  
G. Pillonetto. "Generalized Kalman smoothing: Modeling and  
algorithms". In: *Automatica* 86 (Dec. 2017), pp. 63–86.
- 39 [Arb+18] M. Arbel, D. Sutherland, M. Bińkowski, and A. Gret-  
ton. "On gradient regularizers for MMD GANs". In: *Advances  
in neural information processing systems*. 2018, pp. 6700–6710.
- 40 [Arj+19] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-  
Paz. "Invariant Risk Minimization". In: (July 2019). arXiv: [1907.02893 \[stat.ML\]](https://arxiv.org/abs/1907.02893).
- 41 [Arj+20] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-  
Paz. *Invariant Risk Minimization*. 2020. arXiv: [1907.02893 \[stat.ML\]](https://arxiv.org/abs/1907.02893).
- 42 [Arn+10] C. W. Arnold, S. M. El-Saden, A. A. Bui, and R.  
Taira. "Clinical case-based retrieval using latent topic analysis".  
In: *AMIA annual symposium proceedings*. Vol. 2010. American  
Medical Informatics Association. 2010, p. 26.
- 43 [Aro+19] S. Arora, S. S. Du, W. Hu, Z. Li, R. Salakhutdinov,  
and R. Wang. "On Exact Computation with an Infinitely Wide  
Neural Net". In: (Apr. 2019). arXiv: [1904.11985 \[cs.LG\]](https://arxiv.org/abs/1904.11985).
- 44 [Aro+21] R. Arora et al. *Theory of deep learning*. 2021.
- 45 [ARS13] N. S. Arora, S. Russell, and E. Sudderth. "NET-  
VISA: Network Processing Vertically Integrated Seismic  
AnalysisNET-VISA: Network Processing Vertically Integrated  
Seismic Analysis". In: *Bull. Seismol. Soc. Am.* 103.2A (Apr.  
2013), pp. 709–729.
- 46 [Aru+02] M. Arulampalam, S. Maskell, N. Gordon, and  
T. Clapp. "A Tutorial on Particle Filters for Online  
Nonlinear/Gaussian Bayesian Tracking". In: *IEEE Trans.  
on Signal Processing* 50.2 (2002), pp. 174–189.
- 47 [Aru+17] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and  
A. A. Bharath. "A Brief Survey of Deep Reinforcement Learn-  
ing". In: *IEEE Signal Processing Magazine, Special Issue on  
Deep Learning for Image Understanding* (2017).
- 48 [AS17] A. Achille and S. Soatto. "On the Emergence of Invari-  
ance and Disentangling in Deep Representations". In: (June  
2017). arXiv: [1706.01350 \[cs.LG\]](https://arxiv.org/abs/1706.01350).
- 49 [AS18] A. Achille and S. Soatto. "On the Emergence of Invari-  
ance and Disentangling in Deep Representations". In: *JMLR* 18  
(2018), pp. 1–34.
- 50 [AS66] S. M. Ali and S. D. Silvey. "A General Class of Coeffi-  
cients of Divergence of One Distribution from Another". In: *J.  
R. Stat. Soc. Series B Stat. Methodol.* 28.1 (1966), pp. 131–  
142.
- 51 [Asa00] C. Asavathiratham. "The Influence Model: A Tractable  
Representation for the Dynamics of Networked Markov Chains".  
PhD thesis. MIT, Dept. EECS, 2000.
- 52 [ASD20] A. Agrawal, D. Sheldon, and J. Domke. "Advances in  
Black-Box VI: Normalizing Flows, Importance Weighting, and  
Optimization". In: *NIPS*. June 2020.
- 53 [ASM17] A. Azuma, M. Shimbo, and Y. Matsumoto. "An Al-  
gebraic Formalization of Forward and Forward-backward Algo-  
rithms". In: (Feb. 2017). arXiv: [1702.06941 \[cs.LG\]](https://arxiv.org/abs/1702.06941).
- 54 [ASN20] R. Agarwal, D. Schuurmans, and M. Norouzi. "An Op-  
timistic Perspective on Offline Reinforcement Learning". In:  
*ICML*. 2020.
- 55 [ASS19] I. Andrews, J. H. Stock, and L. Sun. "Weak Instruments  
in Instrumental Variables Regression: Theory and Practice". In:  
*Annual Review of Economics* 11.1 (2019).

- 1
- [AT08] C. Andrieu and J. Thoms. "A tutorial on adaptive MCMC". In: *Statistical Computing* 18 (2008), pp. 343–373.
- 2
- [AT20] E. Agustsson and L. Theis. "Universally Quantized Neural Compression". 2020.
- 3
- [Att00] H. Attias. "A Variational Bayesian Framework for Graphical Models". In: *NIPS-12*. 2000.
- 4
- [Aue12] J. E. Auerbach. "Automated evolution of interesting images". In: *Artificial Life 19*. 2012.
- 5
- [AWR17] J. Altschuler, J. Weed, and P. Rigollet. "Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration". In: *arXiv preprint arXiv:1705.09634* (2017).
- 6
- [AXK17] B. Amos, L. Xu, and J. Z. Kolter. "Input convex neural networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 146–155.
- 7
- [AY19] B. Amos and D. Yarats. "The Differentiable Cross-Entropy Method". In: (Sept. 2019). arXiv: 1909.12830 [cs.LG].
- 8
- [AZ17] S. Arora and Y. Zhang. "Do gans actually learn the distribution? an empirical study". In: *arXiv preprint arXiv:1706.08224* (2017).
- 9
- [Aze+20] E. M. Azevedo, A. Deng, J. L. Montiel Olea, J. Rao, and E. G. Weyl. "A/B Testing with Fat Tails". In: *J. Polit. Econ.* (July 2020), pp. 000–000.
- 10
- [BA03] D. Barber and F. Agakov. "The IM Algorithm: A Variational Approach to Information Maximization". In: *NIPS*. NIPS'03. Cambridge, MA, USA: MIT Press, 2003, pp. 201–208.
- 11
- [BA05] W. Bechtel and A. Abrahamsen. "Explanation: A mechanist alternative". In: *Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences* 36.2 (2005), pp. 421–441.
- 12
- [Bac09] F. Bach. "High-Dimensional Non-Linear Variable Selection through Hierarchical Kernel Learning". In: (Sept. 2009). arXiv: 0909.0844 [cs.LG].
- 13
- [Bac+13] F. Bach et al. "Learning with Submodular Functions: A Convex Optimization Perspective". In: *Foundations and Trends® in Machine Learning* 6.2–3 (2013), pp. 145–373.
- 14
- [Bac+15] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: *PloS one* 10.7 (2015), e0130140.
- 15
- [Bac+18] E. Bach, J. Dusart, L. Hellerstein, and D. Kletenik. "Submodular goal value of Boolean functions". In: *Discrete Applied Mathematics* 233 (2018), pp. 1–13.
- 16
- [Bad+18] M. A. Badgley, J. R. Zech, L. Oakden-Rayner, B. S. Glicksberg, M. Liu, W. Gale, M. V. McConnell, B. Percha, T. M. Snyder, and J. T. Dudley. "Deep Learning Predicts Hip Fracture using Confounding Patient and Healthcare Variables". In: *CoRR* abs/1811.03695 (2018). arXiv: 1811.03695.
- 17
- [Bah+20] Y. Bahri, J. Kadmon, J. Pennington, S. Schoenholz, J. Sohl-Dickstein, and S. Ganguli. "Statistical Mechanics of Deep Learning". In: *Annu. Rev. Condens. Matter Phys.* (Mar. 2020).
- 18
- [Bai+15] R. Bairi, R. Iyer, G. Ramakrishnan, and J. Bilmes. "Summarization of multi-document topic hierarchies using submodular mixtures". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2015, pp. 553–563.
- 19
- [Bai95] L. C. Baird. "Residual Algorithms: Reinforcement Learning with Function Approximation". In: *ICML*. 1995, pp. 30–37.
- 20
- [Bak+17] J. Baker, P. Fearnhead, E. B. Fox, and C. Nemeth. "Control Variates for Stochastic Gradient MCMC". In: (June 2017). arXiv: 1706.05439 [stat.CO].
- 21
- [Bal+18] D. Balduzzi, S. Racaniere, J. Martens, J. Foerster, K. Tuyls, and T. Graepel. "The mechanics of n-player differentiable games". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 354–363.
- 22
- [Ban+05] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. "Clustering on the unit hypersphere using von Mises-Fisher distributions". In: *JMLR*. 2005, pp. 1345–1382.
- 23
- [Ban06] A. Banerjee. "On bayesian bounds". In: *ICML*. 2006, pp. 81–88.
- 24
- [Ban+18] A. Bansal, S. Ma, D. Ramanan, and Y. Sheikh. "Recycle-gan: Unsupervised video retargeting". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 119–135.
- 25
- [Baq+16] P. Baqué, T. Bagautdinov, F. Fleuret, and P. Fua. "Principled parallel mean-field inference for discrete random fields". In: *CVPR*. 2016, pp. 5848–5857.
- 26
- [Bar17] D. Barber. *Evolutionary Optimization as a Variational Method*. 2017.
- 27
- [Bar+19] R. F. Barber, E. J. Candès, A. Ramdas, and R. J. Tibshirani. "Predictive inference with the jackknife+". In: (May 2019). arXiv: 1905.02928 [stat.ME].
- 28
- [Bas+01] S. Basu, T. Choudhury, B. Clarkson, and A. Pentland. *Learning Human Interactions with the Influence Model*. Tech. rep. 539. MIT Media Lab, 2001.
- 29
- [Bat+18] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261* (2018).
- 30
- [Bau+17] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. "Network Dissection: Quantifying Interpretability of Deep Visual Representations". In: *Computer Vision and Pattern Recognition*. 2017.
- 31
- [Bau+18] D. Bau, J.-Y. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba. "Gan dissection: Visualizing and understanding generative adversarial networks". In: *arXiv preprint arXiv:1811.10597* (2018).
- 32
- [Bau+20] D. Bau, J.-Y. Zhu, H. Strobelt, A. Lapedriza, B. Zhou, and A. Torralba. "Understanding the role of individual units in a deep neural network". In: *Proceedings of the National Academy of Sciences* (2020).
- 33
- [Bau+70] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. "A maximization technique occurring in the statistical analysis of probabilistic functions in Markov chains". In: *The Annals of Mathematical Statistics* 41 (1970), pp. 164–171.
- 34
- [Bay+15] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. "Automatic differentiation in machine learning: a survey". In: (Feb. 2015). arXiv: 1502.05767 [cs.SC].
- 35
- [BB15a] A. Bendale and T. Boult. "Towards Open World Recognition". In: *CVPR*. 2015.
- 36
- [BB15b] J. Bornschein and Y. Bengio. "Reweighted Wake-Sleep". In: *ICLR*. 2015.
- 37
- [BB17] J. Bilmes and W. Bai. "Deep Submodular Functions". In: *Arxiv* abs/1701.08939 (2017).
- 38
- [BB18] W. Bai and J. Bilmes. "Greed is Still Good: Maximizing Monotone Submodular+Supermodular (BP) Functions". In: *International Conference on Machine Learning (ICML)*. <http://proceedings.mlr.press/v80/bai18a.html>. Stockholm, Sweden, 2018.
- 39
- [BBH12] C. Blundell, J. Beck, and K. A. Heller. "Modelling reciprocating relationships with Hawkes processes". In: *Advances in Neural Information Processing Systems*. 2012, pp. 2600–2608.
- 40
- [BBM07] A. Banerjee, S. Basu, and S. Merugu. "Multi-way Clustering on Relation Graphs". In: *Proc. SIAM Int'l. Conf. on Data Mining (SDM)*. 2007.
- 41
- [BBM10] N. Bhatnagar, C. Bogdanov, and E. Mossel. *The Computational Complexity of Estimating Convergence Time*. Tech. rep. arxiv, 2010.
- 42
- [BBS09] J. O. Berger, J. M. Bernardo, and D. Sun. "The Formal Definition of Reference Priors". In: *Ann. Stat.* 37.2 (2009), pp. 905–938.
- 43
- [BBS95] A. G. Barto, S. J. Bradtko, and S. P. Singh. "Learning to act using real-time dynamic programming". In: *AIJ* 72.1 (1995), pp. 81–138.
- 44
- [BBV11a] R. Benassi, J. Bect, and E. Vazquez. "Bayesian optimization using sequential Monte Carlo". In: (Nov. 2011). arXiv: 1111.4802 [math.OC].
- 45
- [BBV11b] R. Benassi, J. Bect, and E. Vazquez. "Robust Gaussian Process-Based Global Optimization Using a Fully Bayesian Expected Improvement Criterion". en. In: *Intl. Conf. on Learning and Intelligent Optimization (LIION)*. Jan. 2011, pp. 176–190.
- 46
- [BC07] D. Barber and S. Chiappa. "Unified inference for variational Bayesian linear Gaussian state space models". In: *NIPS*. 2007.
- 47
- [BC08] M. Bădoiu and K. L. Clarkson. "Optimal core-sets for balls". In: *Computational Geometry* 40.1 (2008), pp. 14–22.
- 48
- [BC14] J. Ba and R. Caruana. "Do Deep Nets Really Need to be Deep?". In: *Advances in Neural Information Processing Systems* 27 (2014).
- 49
- [BC17] D. Beck and T. Cohn. "Learning Kernels over Strings using Gaussian Processes". In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Vol. 2. 2017, pp. 67–73.
- 50
- [BC89] D. P. Bertsekas and D. A. Castanon. "The auction algorithm for the transportation problem". In: *Annals of Operations Research* 20.1 (1989), pp. 67–96.
- 51
- [BC94] P. Baldi and Y. Chauvin. "Smooth online learning algorithms for Hidden Markov Models". In: *Neural Computation* 6 (1994), pp. 305–316.
- 52
- [BCF10] E. Brochu, V. M. Cora, and N. de Freitas. "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning". In: (Dec. 2010). arXiv: 1012.2599 [cs.LG].

- 1 [BCH20] M. Briers, M. Charalambides, and C. Holmes. "Risk scoring calculation for the current NHSx contact tracing app". In: (May 2020). arXiv: 2005.11057 [cs.CY].
- 2 [BCJ20] A. Buchholz, N. Chopin, and P. E. Jacob. "Adaptive Tuning Of Hamiltonian Monte Carlo Within Sequential Monte Carlo". In: *Bayesian Anal.* (2020).
- 3 [BCN18] L. Bottou, F. E. Curtis, and J. Nocedal. "Optimization Methods for Large-Scale Machine Learning". In: *SIAM Rev.* 60.2 (2018), pp. 223–311.
- 4 [BCNM06] C. Buciluă, R. Caruana, and A. Niculescu-Mizil. "Model compression". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, pp. 535–541.
- 5 [BCVD18] A. Bouchard-Côté, S. J. Vollmer, and A. Doucet. "The Bouncy Particle Sampler: A Nonreversible Rejection-Free Markov Chain Monte Carlo Method". In: *JASA* 113.522 (Apr. 2018), pp. 855–867.
- 6 [BD11] A. Bhattacharya and D. B. Dunson. "Simplex factor models for multivariate unordered categorical data". In: *JASA* (2011).
- 7 [BD87] G. Box and N. Draper. *Empirical Model-Building and Response Surfaces*. Wiley, 1987.
- 8 [BD92] D. Bayer and P. Diaconis. "Trailing the dovetail shuffle to its lair". In: *The Annals of Applied Probability* 2.2 (1992), pp. 294–313.
- 9 [BDM09] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems*. SIAM, 2009.
- 10 [BDM10] M. Briers, A. Doucet, and S. Maskell. "Smoothing algorithms for state-space models". In: *Annals of the Institute of Statistical Mathematics* 62.1 (2010), pp. 61–89.
- 11 [BDM17] M. G. Bellemare, W. Dabney, and R. Munos. "A Distributional Perspective on Reinforcement Learning". In: *ICML*. 2017.
- 12 [BDM18] N. Brosse, A. Durmus, and E. Moulines. "The promises and pitfalls of Stochastic Gradient Langevin Dynamics". In: *NIPS*. Nov. 2018.
- 13 [BDS18] A. Brock, J. Donahue, and K. Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: (Sept. 2018). arXiv: 1809.11096 [cs.LG].
- 14 [Bea03] M. Beal. "Variational Algorithms for Approximate Bayesian Inference". PhD thesis. Gatsby Unit, 2003.
- 15 [Bea19] M. A. Beaumont. "Approximate Bayesian Computation". In: *Annual Review of Statistics and Its Application* 6.1 (2019), pp. 379–403.
- 16 [Béd08] M. Bédard. "Optimal acceptance rates for Metropolis algorithms: Moving beyond 0.234". In: *Stochastic Process. Appl.* 118.12 (Dec. 2008), pp. 2198–2222.
- 17 [Beh+19] J. Behrmann, W. Grathwohl, R. T. Q. Chen, D. Duvenaud, and J.-H. Jacobsen. "Invertible Residual Networks". In: *ICML*. 2019.
- 18 [Bel03] A. J. Bell. "The co-information lattice". In: *ICA conference*. 2003.
- 19 [Bel+16] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. "Unifying Count-Based Exploration and Intrinsic Motivation". In: *NIPS*. 2016.
- 20 [Bel+18] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm. "Mutual Information Neural Estimation". In: *ICML*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, 2018, pp. 531–540.
- 21 [Bel+19a] D. Belanger, S. Vora, Z. Mariet, R. Deshpande, D. Dohan, C. Angermüller, K. Murphy, O. Chapelle, and L. Colwell. "Biological Sequence Design using Batched Bayesian Optimization". In: *NIPS workshop on ML for the sciences*. 2019.
- 22 [Bel+19b] M. Belkin, D. Hsu, S. Ma, and S. Mandal. "Reconciling modern machine-learning practice and the classical bias-variance trade-off". en. In: *PNAS* 116.32 (Aug. 2019), pp. 15849–15854.
- 23 [Ben13] Y. Bengio. "Estimating or Propagating Gradients Through Stochastic Neurons". In: (May 2013). arXiv: 1305.2982 [cs.LG].
- 24 [Ben+19] G. W. Benton, W. J. Maddox, J. P. Salkey, J. Albinati, and A. G. Wilson. "Function-space Distributions over Kernels". In: *Advances in Neural Information Processing Systems*. 2019.
- 25 [Ben+20] K. Benidis et al. "Neural forecasting: Introduction and literature overview". In: (Apr. 2020). arXiv: 2004.10240 [cs.LG].
- 26 [Bén+21] C. Bénard, G. Biau, S. Veiga, and E. Scornet. "Interpretable random forests via rule extraction". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 937–945.
- 27 [Ben+21] G. W. Benton, W. J. Maddox, S. Lotfi, and A. G. Wilson. "Loss Surface Simplexes for Mode Connecting Volumes and Fast Ensembling". In: *ICML*. 2021.
- 28 [Ber05] J. M. Bernardo. "Reference Analysis". In: *Handbook of Statistics*. Ed. by D. K. Dey and C. R. Rao. Vol. 25. Elsevier, Jan. 2005, pp. 17–90.
- 29 [Ber15] D. Bertsekas. *Convex Optimization Algorithms*. Athena Scientific, 2015.
- 30 [Ber16] D. Bertsekas. *Nonlinear Programming*. Third. Athena Scientific, 2016.
- 31 [Ber+18] R. van den Berg, L. Hasenclever, J. M. Tomczak, and M. Welling. "Sylvester normalizing flows for variational inference". In: *AISTATS*. 2018.
- 32 [Ber+19] H. Berard, G. Gidel, A. Almahairi, P. Vincent, and S. Lacoste-Julien. "A Closer Look at the Optimization Landscapes of Generative Adversarial Networks". In: *International Conference on Learning Representations*. 2019.
- 33 [Ber19] D. Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, 2019.
- 34 [Ber+21a] J. Berner, P. Grohs, G. Kutyniok, and P. Petersen. "The Modern Mathematics of Deep Learning". In: (May 2021). arXiv: 2105.04026 [cs.LG].
- 35 [Ber+21b] B. Beronov, C. Weilbach, F. Wood, and T. Campbell. "Sequential core-set Monte Carlo". In: *UAI*. Ed. by C. de Campos and M. H. Maathuis. Vol. 161. Proceedings of Machine Learning Research. PMLR, 2021, pp. 2165–2175.
- 36 [Ber85] J. Berger. "Bayesian Salesmanship". In: *Bayesian Inference and Decision Techniques with Applications: Essays in Honor of Bruno deFinetti*. Ed. by P. K. Goel and A. Zellner. North-Holland, 1985.
- 37 [Ber96] J. Bertoin. *Lévy processes*. Vol. 121. Cambridge university press Cambridge, 1996.
- 38 [Ber97] D. Bertsekas. *Parallel and Distribution Computation: Numerical Methods*. Athena Scientific, 1997.
- 39 [Ber99] A. Berchtold. "The double chain Markov model". In: *Comm. Stat. Theor. Methods* 28 (1999), pp. 2569–2589.
- 40 [Bes75] J. Besag. "Statistical analysis of non-lattice data". In: *The Statistician* 24 (1975), pp. 179–196.
- 41 [Bet13] M. Betancourt. "A General Metric for Riemannian Manifold Hamiltonian Monte Carlo". In: *Geometric Science of Information*. Springer Berlin Heidelberg, 2013, pp. 327–334.
- 42 [Bet17] M. Betancourt. "A Conceptual Introduction to Hamiltonian Monte Carlo". In: (Oct. 2017). arXiv: 1701.02434 [stat.ME].
- 43 [BFH75] Y. Bishop, S. Fienberg, and P. Holland. *Discrete Multivariate Analysis: Theory and Practice*. MIT Press, 1975.
- 44 [BG06] M. Beal and Z. Ghahramani. "Variational Bayesian Learning of Directed Graphical Models with Hidden Variables". In: *Bayesian Analysis* 1.4 (2006).
- 45 [BG13] M. J. Betancourt and M. Girolami. "Hamiltonian Monte Carlo for Hierarchical Models". In: (Dec. 2013). arXiv: 1312.0906 [stat.ME].
- 46 [BG96] A. Becker and D. Geiger. "A sufficiently fast algorithm for finding close to optimal junction trees". In: *UAI*. 1996.
- 47 [BGHM17] J. Boyd-Graber, Y. Hu, and D. Mimno. "Applications of Topic Models". In: *Foundations and Trends® in Information Retrieval* 11.2–3 (2017), pp. 143–296.
- 48 [BGM17] J. Ba, R. Grosse, and J. Martens. "Distributed Second-Order Optimization using Kronecker-Factored Approximations". In: *ICLR*. openreview.net, 2017.
- 49 [BGS16] Y. Burda, R. Grosse, and R. Salakhutdinov. "Importance Weighted Autoencoders". In: *ICLR*. 2016.
- 50 [BGT93] C. Berrou, A. Glavieux, and P. Thitimajshima. "Near Shannon limit error-correcting coding and decoding: Turbo codes". In: *Proc. IEEE Intl. Comm. Conf.* (1993).
- 51 [BH11] M.-F. Balcan and N. J. Harvey. "Learning submodular functions". In: *Proceedings of the forty-third annual ACM symposium on Theory of computing*. 2011, pp. 793–802.
- 52 [BH12] R. G. Brown and P. Y. C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises*. 4th ed. Wiley, 2012.
- 53 [BH20] M. T. Bahadori and D. Heckerman. "Debiasing Concept-based Explanations with Causal Analysis". In: *International Conference on Learning Representations*. 2020.
- 54 [BH92] D. Barry and J. A. Hartigan. "Product partition models for change point problems". In: *Annals of statistics* 20 (1992), pp. 260–279.
- 55 [Bha+19] A. Bhadra, J. Datta, N. G. Polson, and B. T. Willard. "Lasso Meets Horseshoe: a survey". In: *Bayesian Anal.* 34.3 (2019), pp. 405–427.
- 56 [Bha+21] K. Bhatia, N. Kuang, Y. Ma, and Y. Wang. "Statistical and computational tradeoffs in variational Bayes: a case study of inferential model selection". Tech. rep. 2021.
- 57 [BHC19] M. Binkowski, D. Hjelm, and A. Courville. "Batch weight for domain adaptation with mass shift". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1844–1853.

- 1
- [BHP02] M. Bădoiu, S. Har-Peled, and P. Indyk. "Approximate clustering via core-sets". In: *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*. 2002, pp. 250–257.
- 2
- [BHW16] P. G. Bissiri, C. Holmes, and S. Walker. "A General Framework for Updating Belief Distributions". In: *JRSSB* 78.5 (2016), 1103–1130.
- 3
- [Bic09] D. Bickson. "Gaussian Belief Propagation: Theory and Application". PhD thesis. Hebrew University of Jerusalem, 2009.
- 4
- [Bie06] G. J. Bierman. *Factorization Methods for Discrete Sequential Estimation* (Dover Books on Mathematics). en. Illustrated edition. Dover Publications, May 2006.
- 5
- [Big+11] B. Biggio, G. Fumerà, I. Pillai, and F. Roli. "A survey and experimental evaluation of image spam filtering techniques". In: *Pattern recognition letters* 32.10 (2011), pp. 1436–1446.
- 6
- [Bil01] J. A. Bilmes. *Graphical Models and Automatic Speech Recognition*. Tech. rep. UWEETR-2001-0005. Univ. Washington, Dept. of Elec. Eng., 2001.
- 7
- [Bil22] J. Bilmes. "Submodularity In Machine Learning and Artificial Intelligence". In: (2022). arXiv: 2202.00132 [cs.LG].
- 8
- [Biń+18] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton. "Demystifying MMD GANs". In: *ICLR*. 2018.
- 9
- [Biń+18] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton. "Demystifying MMD GANs". In: *International Conference on Learning Representations*. 2018.
- 10
- [Bin+19] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman. "Pyro: Deep Universal Probabilistic Programming". In: *JMLR* 20.28 (2019), pp. 1–6.
- 11
- [Biń+19] M. Bińkowski, J. Donahue, S. Dieleman, A. Clark, E. Elsen, N. Casagrande, L. C. Cobo, and K. Simonyan. "High Fidelity Speech Synthesis with Adversarial Networks". In: *International Conference on Learning Representations*. 2019.
- 12
- [Bin+97] J. Binder, D. Koller, S. J. Russell, and K. Kanazawa. "Adaptive Probabilistic Networks with Hidden Variables". In: *Machine Learning* 29 (1997), pp. 213–244.
- 13
- [Bis06] C. Bishop. *Pattern recognition and machine learning*. Springer, 2006.
- 14
- [Bis99] C. Bishop. "Bayesian PCA". In: *NIPS*. 1999.
- 15
- [Bit16] S. Bitzer. *The UKF exposed: How it works, when it works and when it's better to sample*. 2016.
- 16
- [Bit+21] J. Bitterwolf, A. Meinke, M. Augustin, and M. Hein. "Revisiting out-of-distribution detection: A simple baseline is surprisingly effective". In: *ICML Workshop on Uncertainty in Deep Learning (UDL)*. 2021.
- 17
- [BJ05] F. Bach and M. Jordan. *A probabilistic interpretation of canonical correlation analysis*. Tech. rep. 688. U. C. Berkeley, 2005.
- 18
- [BJ+06] D. M. Blei, M. I. Jordan, et al. "Variational inference for Dirichlet process mixtures". In: *Bayesian analysis* 1.1 (2006), pp. 121–143.
- 19
- [BJ06] W. Buntine and A. Jakulin. "Discrete Component Analysis". In: *Subspace, Latent Structure and Feature Selection: Statistical and Optimization Perspectives* Workshop. 2006.
- 20
- [BK01] Y. Boykov and V. Kolmogorov. "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Computer Vision". In: *Third International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. 2001.
- 21
- [BK10] R. Bardehnet and B. Kegl. "Surrogating the surrogate: accelerating Gaussian-process-based global optimization with a mixture cross-entropy algorithm". In: *ICML*. 2010.
- 22
- [BK15] D. Belanger and S. Kakade. "A Linear Dynamical System Model for Text". en. In: *ICML*. June 2015, pp. 833–842.
- 23
- [BK19] M. Bonvini and E. H. Kennedy. "Sensitivity Analysis via the Proportion of Unmeasured Confounding". In: *arXiv e-prints*, arXiv:1912.02793 (Dec. 2019), arXiv:1912.02793. ArXiv: 1912.02793 [stat.ME].
- 24
- [BKB17] O. Bastani, C. Kim, and H. Bastani. "Interpreting blackbox models via model extraction". In: *arXiv preprint arXiv:1705.08504* (2017).
- 25
- [BKH16] J. L. Ba, J. R. Kiros, and G. E. Hinton. "Layer Normalization". In: (2016). arXiv: 1607.06450 [stat.ML].
- 26
- [BKKJ13] T. Broderick, B. Kulis, and M. Jordan. "MAD-Bayes: MAP-based asymptotic derivations from Bayes". In: *International Conference on Machine Learning*. PMLR. 2013, pp. 226–234.
- 27
- [BKM16] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. "Variational Inference: A Review for Statisticians". In: *JASA* (2016).
- 28
- [BKS19] A. Bennett, N. Kallus, and T. Schnabel. "Deep Generalized Method of Moments for Instrumental Variable Analysis". In: *Advances in Neural Information Processing Systems*. 2019, pp. 3564–3574.
- 29
- [BL06] D. Blei and J. Lafferty. "Dynamic topic models". In: *ICML*. 2006, pp. 113–120.
- 30
- [BLBV12] N. Boulaenger-Lewandowski, Y. Bengio, and P. Vincent. "Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription". In: *ICML*. June 2012.
- 31
- [BLC18] A. J. Bose, H. Ling, and Y. Cao. "Adversarial Contrastive Estimation". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 1021–1032.
- 32
- [Ble12] D. M. Blei. "Probabilistic topic models". In: *Commun. ACM* 55.4 (2012), pp. 77–84.
- 33
- [Ble17] D. Blei. *Variational inference: foundations and innovations* (Lecture). Simons Institute Lecture. 2017.
- 34
- [BLM16] S. Boucheron, G. Lugosi, and P. Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2016.
- 35
- [Blo+16] A. Bloniarz, H. Liu, C.-H. Zhang, J. S. Sekhon, and B. Yu. "Lasso adjustments of treatment effect estimates in randomized experiments". In: *Proceedings of the National Academy of Sciences* 113.27 (2016), pp. 7383–7390.
- 36
- [BLSS11] G. Blanchard, G. Lee, and C. Scott. "Generalizing from several related classification tasks to a new unlabeled sample". In: *NIPS*. 2011.
- 37
- [BLS17] J. Ballé, V. Laparra, and E. P. Simoncelli. "End-to-end Optimized Image Compression". In: *ICLR*. 2017.
- 38
- [Blu+15] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. "Weight Uncertainty in Neural Networks". In: *ICML*. May 2015.
- 39
- [BM+18] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, T. B. Dhrava, A. Muldal, N. Heess, and T. Lillicrap. "Distributed Distributional Deterministic Policy Gradients". In: *ICLR*. 2018.
- 40
- [Bla19] Y. Blau and T. Michaeli. "Rethinking Lossy Compression: The Rate-Distortion-Perception Tradeoff". In: *ICML*. 2019.
- 41
- [BM+73] D. Blackwell, J. B. MacQueen, et al. "Ferguson distributions via Pólya urn schemes". In: *The annals of statistics* 1.2 (1973), pp. 353–355.
- 42
- [BMK20] Z. Boros, M. Mutny, and A. Krause. "Coresets via Bilevel Optimization for Continual Learning and Streaming". In: *Advances in Neural Information Processing Systems* 33 (2020).
- 43
- [BMP19] A. Brunel, D. Mazza, and M. Pagani. "Backpropagation in the Simply-Typed Lambda-Calculus with Linear Negation". In: *Proc. ACM Program. Lang.* 4.POPL (Dec. 2019).
- 44
- [BMR97a] J. Binder, K. Murphy, and S. Russell. "Space-efficient inference in dynamic probabilistic networks". In: *IJCAI*. 1997.
- 45
- [BMR97b] S. Bistarelli, U. Montanari, and F. Rossi. "Semiring-Based Constraint Satisfaction and Optimization". In: *JACM* 44.2 (1997), pp. 201–236.
- 46
- [BMS11] S. Bubeck, R. Munos, and G. Stoltz. "Pure Exploration in Finitely-armed and Continuous-armed Bandits". In: *Theoretical Computer Science* 412.19 (2011), pp. 1832–1852.
- 47
- [BNJ03a] D. Blei, A. Ng, and M. Jordan. "Latent Dirichlet allocation". In: *JMLR* 3 (2003), pp. 993–1022.
- 48
- [BNJ03b] D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent dirichlet allocation". In: *JMLR* 3 (2003), pp. 993–1022.
- 49
- [BO14] J. Bayer and C. Osendorfer. "Learning Stochastic Recurrent Networks". In: *Workshop on Advances in Variational Inference*. Nov. 2014.
- 50
- [Boh92] D. Boning. "Multinomial logistic regression algorithm". In: *Annals of the Inst. of Statistical Math.* 44 (1992), pp. 197–200.
- 51
- [Bol89] K. Bollen. *Structural Equation Models with Latent Variables*. John Wiley & Sons, 1989.
- 52
- [Bon64] G. Bonnet. "Transformations des signaux aléatoires à travers les systèmes non linéaires sans mémoire". In: *Annales des Telecommunications* 19 (1964).
- 53
- [Bor] A. Borodin. "Determinantal point processes". In: *The Oxford Handbook of Random Matrix Theory*.
- 54
- [Bor16] S. M. Borekachev. "Recursive least squares method of regression coefficients estimation as a special case of Kalman filter". In: *Intl. Conf. of numerical analysis and applied mathematics*. Vol. 1738. American Institute of Physics, 2016, p. 110013.
- 55
- [Bös+17] J.-H. Böse, V. Flunkert, J. Gasthaus, T. Januschowski, D. Lange, D. Salinas, S. Schelter, M. Seeger, and Y. Wang. "Probabilistic Demand Forecasting at Scale". In: *Proceedings VLDB Endowment* 10.12 (Aug. 2017), pp. 1694–1705.
- 56
- [Bot+13] L. Bottou, J. Peters, J. Quiñonero-Candela, D. X. Charles, D. M. Chickering, E. Portugaly, D. Ray, P. Simard, and E. Snelson. "Counterfactual Reasoning and Learning Sys-

- tems: The Example of Computational Advertising". In: *JMLR* 14 (2013), pp. 3207–3260.
- [Bow+16a] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. "Generating Sentences from a Continuous Space". In: *CoNLL*. 2016.
- [Bow+16b] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. "Generating Sentences from a Continuous Space". In: *CoNLL*. 2016.
- [BP13] K. A. Bollen and J. Pearl. "Eight Myths About Causality and Structural Equation Models". In: *Handbook of Causal Analysis for Social Research*, Ed. by S. L. Morgan. Dordrecht: Springer Netherlands, 2013, pp. 301–328.
- [BP16] E. Bareinboim and J. Pearl. "Causal inference and the data-fusion problem". en. In: *Proc. Natl. Acad. Sci. U. S. A.* 113.27 (July 2016), pp. 7345–7352.
- [BP21] T. Bricken and C. Pehlevan. "Attention Approximates Sparse Distributed Memory". In: *NIPS*. May 2021.
- [BP92] J. Blair and B. Peyton. *An introduction to chordal graphs and clique trees*. Tech. rep. Oak Ridge National Lab, 1992.
- [BPK16] S. Bulo, L. Porzi, and P. Kontschieder. "Distillation dropout". In: *ICML*. 2016.
- [BPK18] M. Benatan and E. O. Pyzer-Knapp. "Practical Considerations for Probabilistic Backpropagation". In: *NIPS Workshop on Bayesian Deep Learning*. 2018.
- [BPS16] A. G. Baydin, B. A. Pearlmutter, and J. M. Siskind. "DiffSharp: An AD library for .NET languages". In: *arXiv preprint arXiv:1611.03423* (2016).
- [BR05] H. Bang and J. M. Robins. "Doubly Robust Estimation in Missing Data and Causal Inference Models". In: *Biometrics* 61.4 (2005), pp. 962–973.
- [BR18] B. Biggio and F. Roli. "Wild patterns: Ten years after the rise of adversarial machine learning". In: *Pattern Recognition* 84 (2018), pp. 317–331.
- [BR98] S. Brooks and G. Roberts. "Assessing convergence of Markov Chain Monte Carlo algorithms". In: *Statistics and Computing* 8 (1998), pp. 319–335.
- [Bra+18] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, and S. Wanderman-Milne. *JAX: composable transformations of Python+NumPy programs*. Version 0.1.55. 2018.
- [Bra96] M. Brand. *Coupled hidden Markov models for modeling interacting processes*. Tech. rep. 405. MIT Lab for Perceptual Computing, 1996.
- [Bre01] L. Breiman. "Statistical modeling: The two cultures (with comments and rejoinder by the author)". In: *Statistical science* 16.3 (2001), pp. 199–231.
- [Bre+17] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*. Routledge, 2017.
- [Bre+20a] J. Brehmer, G. Louppe, J. Pavese, and K. Cranmer. "Mining gold from implicit models to improve likelihood-free inference". en. In: *Proc. Natl. Acad. Sci. U. S. A.* 117.10 (Mar. 2020), pp. 5242–5249.
- [Bre+20b] R. Brekelmans, V. Masrani, F. Wood, G. Ver Steeg, and A. Galstyan. "All in the Exponential Family: Bregman Duality in Thermodynamic Variational Inference". In: *ICML*. 2020.
- [Bre67] L. M. Bregman. "The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming". In: *USSR Computational Mathematics and Mathematical Physics* 7.3 (Jan. 1967), pp. 200–217.
- [Bre91] Y. Brenier. "Polar factorization and monotone rearrangement of vector-valued functions". In: *Communications on pure and applied mathematics* 44.4 (1991), pp. 375–417.
- [Bre92] J. Breese. "Construction of belief and decision networks". In: *Computational Intelligence* 8 (1992), 624–647.
- [Bre96] L. Breiman. "Stacked regressions". In: *Mach. Learn.* 24.1 (July 1996), pp. 49–64.
- [BRG20] R. Bai, V. Rockova, and E. I. George. "Spike-and-slab meets LASSO: A review of the spike-and-slab LASSO". In: (Oct. 2020). arXiv: 2010.06451 [stat.ME].
- [Bri50] G. W. Brier. "Verification of forecasts expressed in terms of probability". In: *Monthly Weather Review* 78.1 (Jan. 1950), pp. 1–3.
- [Bro09] G. Brown. "A new perspective on information theoretic feature selection". In: *AISTATS*. 2009.
- [Bro+13] T. Broderick, N. Boyd, A. Wibisono, A. C. Wilson, and M. I. Jordan. "Streaming Variational Bayes". In: *NIPS*. 2013.
- [Bro+15] K. H. Brodersen, F. Gallusser, J. Koehler, N. Remy, and S. L. Scott. "Inferring causal impact using Bayesian structural time-series models". In: *Ann. Appl. Stat.* 9.1 (2015), pp. 247–274.
- [Bro18] T. Broderick. *Tutorial: Variational Bayes and Beyond*. 2018.
- [Bro19] J. Brownlee. *Generative Adversarial Networks with Python*. Accessed: 2019-8-27. Machine Learning Mastery, Sept. 2019.
- [Bro+20a] D. Brown, R. Coleman, R. Srinivasan, and S. Nieku. "Safe imitation learning via fast bayesian reward inference from preferences". In: *International Conference on Machine Learning*. PMLR, 2020, pp. 1165–1177.
- [Bro+20b] T. B. Brown et al. "Language Models are Few-Shot Learners". In: (May 2020). arXiv: 2005.14165 [cs.CL].
- [BRS17] E. Balkanski, R. Rubinstein, and Y. Singer. "The Limitations of Optimization from Samples". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2017. Montreal, Canada: Association for Computing Machinery, 2017, 1016–1027.
- [BRSS18] N. Bou-Rabee and J. M. Sanz-Serna. "Geometric integrators and the Hamiltonian Monte Carlo method". In: *Acta Numer.* (2018).
- [Bru+18] M. Brundage et al. "The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation". In: (Feb. 2018). arXiv: 1802.07228 [cs.AI].
- [BS17] E. Balkanski and Y. Singer. "Minimizing a Submodular Function from Samples". In: *NIPS*. 2017, pp. 814–822.
- [BS18] S. Barratt and R. Sharma. "A note on the inception score". In: *arXiv preprint arXiv:1801.01973* (2018).
- [BS20] E. Balkanski and Y. Singer. "A lower bound for parallel submodular minimization". In: *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. 2020, pp. 130–139.
- [BS21] S. Bubeck and M. Sellke. "A Universal Law of Robustness via Isoperimetry". In: (May 2021). arXiv: 2105.12806 [cs.LG].
- [BS95] A. J. Bell and T. J. Sejnowski. "An information maximization approach to blind separation and blind deconvolution". In: *Neural Computation* 7.6 (1995), pp. 1129–1159.
- [BSA83] A. G. Barto, R. S. Sutton, and C. W. Anderson. "Neuronlike adaptive elements that can solve difficult learning control problems". In: *SMC* 13.5 (Sept. 1983), pp. 834–846.
- [BSF88] Y. Bar-Shalom and T. Fortmann. *Tracking and data association*. Academic Press, 1988.
- [BSL93] Y. Bar-Shalom and X. Li. *Estimation and Tracking: Principles, Techniques and Software*. Artech House, 1993.
- [BSWT11] Y. Bar-Shalom, P. K. Willett, and X. Tian. *Tracking and Data Fusion: A Handbook of Algorithms*. en. Yaakov Bar-Shalom, Apr. 2011.
- [BT00] C. Bishop and M. Tipping. "Variational relevance vector machines". In: *UAI*. 2000.
- [BT03] A. Beck and M. Teoule. "Mirror descent and nonlinear projected subgradient methods for convex optimization". In: *Operations Research Letters* 31.3 (2003), pp. 167–175.
- [BT73] G. Box and G. Tiao. *Bayesian inference in statistical analysis*. Addison-Wesley, 1973.
- [BTEGN09] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *R robust optimization*. Vol. 28. Princeton University Press, 2009.
- [Buc+12] N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz. "A tight (1/2) linear-time approximation to unconstrained submodular maximization". In: *In FOCS* (2012).
- [Buc+17] C. L. Buckley, C. S. Kim, S. McGregor, and A. K. Seth. "The free energy principle for action and perception: A mathematical review". In: *J. Math. Psychol.* 81 (Dec. 2017), pp. 55–79.
- [Bul11] A. D. Bull. "Convergence rates of efficient global optimization algorithms". In: *JMLR* 12 (2011), 2879–2904.
- [Bul+20] S. Bulusu, B. Kaikhura, B. Li, P. K. Varshney, and D. Song. "Anomalous Example Detection in Deep Learning: A Survey". In: *IEEE Access* 8 (2020), pp. 132330–132347.
- [BV04] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge, 2004.
- [BVHP18] S. Beery, G. Van Horn, and P. Perona. "Recognition in terra incognita". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 456–473.
- [BVW02] H. Bui, S. Venkatesh, and G. West. "Policy Recognition in the Abstract Hidden Markov Model". In: *JAIR* 17 (2002), pp. 451–499.
- [BVZ01] Y. Boykov, O. Veksler, and R. Zabih. "Fast Approximate Energy Minimization via Graph Cuts". In: *IEEE PAMI* 23.11 (2001).

- 1
- [BVZ99] Y. Boykov, O. Veksler, and R. Zabih. "Fast Approximate Energy Minimization via Graph Cuts". In: *ICCV* (1). 1999, pp. 377–384.
- 2
- [BW20] G. J. van den Burg and C. K. I. Williams. "An Evaluation of Change Point Detection Algorithms". In: (Mar. 2020). arXiv: 2003.06222 [stat.ML].
- 3
- [BW21] G. J. van den Burg and C. K. Williams. "On Memorization in Probabilistic Deep Generative Models". In: *NIPS*. 2021.
- 4
- [BWM18] A. Buchholz, F. Wenzel, and S. Mandt. "Quasi-Monte Carlo Variational Inference". In: *ICML*. 2018.
- 5
- [BWR16] M. Bauer, M. van der Wilk, and C. E. Rasmussen. "Understanding Probabilistic Sparse Gaussian Process Approximations". In: *NIPS*. 2016, pp. 1533–1541.
- 6
- [BYH20] O. Bohdal, Y. Yang, and T. Hospedales. "Flexible Dataset Distillation: Learn Labels Instead of Images". In: *arXiv preprint arXiv:2006.08572* (2020).
- 7
- [BYM17] D. Belanger, B. Yang, and A. McCallum. "End-to-End Learning for Structured Prediction Energy Networks". In: *ICML*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 429–439.
- 8
- [Byr+16] R. Byrd, S. Hansen, J. Nocedal, and Y. Singer. "A Stochastic Quasi-Newton Method for Large-Scale Optimization". In: *SIAM J. Optim.* 26.2 (Jan. 2016), pp. 1008–1031.
- 9
- [BZ20] A. Barbu and S.-C. Zhu. *Monte Carlo Methods*. en. Springer, 2020.
- 10
- [CA13] E. F. Camacho and C. B. Alba. *Model predictive control*. Springer, 2013.
- 11
- [Cac+18] M. Caccia, L. Caccia, W. Fedus, H. Larochelle, J. Pineau, and L. Charlin. "Language GANs Falling Short". In: *CoRR abs/1811.02549* (2018). arXiv: 1811.02549.
- 12
- [CAII20] V. Coscato, M. H. de Almeida Inácio, and R. Izicki. "The NN-Stacking: Feature weighted linear stacking through neural networks". In: *Neurocomputing* (2020).
- 13
- [Cal+07] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. "Maximizing a submodular set function subject to a matroid constraint". In: *Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*. 2007, pp. 182–196.
- 14
- [Cal20] O. Calin. *Deep Learning Architectures: A Mathematical Approach*. en. 1st ed. Springer, Feb. 2020.
- 15
- [Can04] J. Canny. "GaP: a factor model for discrete data". In: *SIGIR*. 2004, pp. 122–129.
- 16
- [Cao+15] Y. Cao, M. A. Brubaker, D. J. Fleet, and A. Hertzmann. "Efficient Optimization for Sparse Gaussian Process Regression". en. In: *IEEE PAMI* 37.12 (Dec. 2015), pp. 2415–2427.
- 17
- [Car03] P. Carbonetto. "Unsupervised Statistical Models for General Object Recognition". MA thesis. University of British Columbia, 2003.
- 18
- [Car+15] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad. "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission". In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015, pp. 1721–1730.
- 19
- [Car+17] B. Carpenter, A. Gelman, M. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell. "Stan: A Probabilistic Programming Language". In: *Journal of Statistical Software, Articles* 76.1 (2017), pp. 1–32.
- 20
- [Car+19] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin. "On evaluating adversarial robustness". In: *arXiv preprint arXiv:1902.06705* (2019).
- 21
- [Car97] R. Caruana. "Multitask Learning". In: *Machine Learning* 28.1 (1997), pp. 41–75.
- 22
- [Cat08] A. Caticha. *Lectures on Probability, Entropy, and Statistical Physics*. 2008. arXiv: 0808.0012 [physics.data-an].
- 23
- [Cat+11] A. Caticha, A. Mohammad-Djafari, J.-F. Bercher, and P. Bessière. "Entropic Inference". In: *AIP Conference Proceedings* 1305.1 (2011), pp. 20–29. eprint: <https://aip.scitation.org/doi/pdf/10.1063/1.3573619>.
- 24
- [Cau+20] M. Cauchois, S. Gupta, A. Ali, and J. C. Duchi. "Robust Validation: Confident Predictions Even When Distributions Shift". In: (Aug. 2020). arXiv: 2008.04267 [stat.ML].
- 25
- [CB20] Y. Chen and P. Bühlmann. "Domain adaptation under structural causal models". In: *JMLR* (Oct. 2020).
- 26
- [CBL20] K. Cranmer, J. Brehmer, and G. Louppe. "The frontier of simulation-based inference". In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30055–30062.
- 27
- [CBR20] Z. Chen, Y. Bei, and C. Rudin. "Concept whitening for interpretable image recognition". In: *Nature Machine Intelligence* 2.12 (2020), pp. 772–782.
- 28
- [CConf20] M. Conforti and G. Cornuejols. "Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem". In: *Discrete Applied Mathematics* 7.3 (1984), pp. 251–274.
- 29
- [CC96] M. Cowles and B. Carlin. "Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review". In: *JASA* 91 (1996), pp. 883–904.
- 30
- [CCS22] A. Corenflos, N. Chopin, and S. Särkkä. "De-Sequentialized Monte Carlo: a parallel-in-time particle smoother". In: (Feb. 2022). arXiv: 2202.02264 [stat.CO].
- 31
- [CDC15] C. Chen, N. Ding, and L. Carin. "On the Convergence of Stochastic Gradient MCMC Algorithms with High-Order Integrators". In: *NIPS*. 2015.
- 32
- [CDS02] M. Collins, S. Dasgupta, and R. E. Schapire. "A Generalization of Principal Components Analysis to the Exponential Family". In: *NIPS*. 2002.
- 33
- [CDS19] A. Clark, J. Donahue, and K. Simonyan. "Adversarial video generation on complex datasets". In: *arXiv preprint arXiv:1907.06571* (2019).
- 34
- [Cér+12] F. Cérou, P. Del Moral, T. Furon, and A. Guyader. "Sequential Monte Carlo for rare event estimation". In: *Stat. Comput.* 22.3 (May 2012), pp. 795–808.
- 35
- [CFG14a] T. Chen, E. B. Fox, and C. Guestrin. "Stochastic Gradient Hamiltonian Monte Carlo". In: *ICML*. 2014.
- 36
- [CFG14b] T. Chen, E. B. Fox, and C. Guestrin. "Stochastic Gradient Hamiltonian Monte Carlo". In: *ICML*. 2014.
- 37
- [CG00] S. S. Chen and R. A. Gopinath. "Gaussianization". In: *NIPS*. 2000, pp. 423–429.
- 38
- [CG18] C. Ceylan and M. U. Gutmann. "Conditional Noise-Contrastive Estimation of Unnormalised Models". In: *International Conference on Machine Learning*. 2018, pp. 726–734.
- 39
- [CG96] S. Chen and J. Goodman. "An empirical study of smoothing techniques for language modeling". In: *Proc. 34th ACL*. 1996, pp. 310–318.
- 40
- [CG98] S. Chen and J. Goodman. *An empirical study of smoothing techniques for language modeling*. Tech. rep. TR-10-98. Dépt. Comp. Sci., Harvard, 1998.
- 41
- [CGR06] S. R. Cook, A. Gelman, and D. B. Rubin. "Validation of Software for Bayesian Models Using Posterior Quantiles". In: *J. Comput. Graph. Stat.* 15.3 (Sept. 2006), pp. 675–692.
- 42
- [CH20] C. Cinelli and C. Hazlett. "Making sense of sensitivity: extending omitted variable bias". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 82.1 (2020), pp. 39–67. eprint: <https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/rssb.12348>.
- 43
- [Cha12] K. M. A. Chai. "Variational Multinomial Logit Gaussian Process". In: *JMLR* 13.Jun (2012), pp. 1745–1808.
- 44
- [Cha14] N. Chapados. "Effective Bayesian Modeling of Groups of Related Count Time Series". In: *ICML*. 2014.
- 45
- [Cha+17] D. Chakrabarty, Y. T. Lee, A. Sidford, and S. C.-w. Wong. "Subquadratic submodular function minimization". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. 2017, pp. 1220–1231.
- 46
- [Cha+18] N. S. Chatterji, N. Flammarion, Y.-A. Ma, P. L. Bartlett, and M. I. Jordan. "On the theory of variance reduction for stochastic gradient Monte Carlo". In: *ICML*. Feb. 2018.
- 47
- [Cha+19a] C. Chan, S. Ginosar, T. Zhou, and A. A. Efros. "Everybody dance now". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 5933–5942.
- 48
- [Cha+19b] O. Chang, Y. Yao, D. Williams-King, and H. Lipson. "Ensemble model patching: A parameter-efficient variational Bayesian neural network". In: *arXiv preprint arXiv:1905.09453* (2019).
- 49
- [Cha+19c] T. Chavdarova, G. Gidel, F. Fleuret, and S. Lacoste-Julien. "Reducing noise in GAN training with variance reduction extragradient". In: *Advances in Neural Information Processing Systems*. 2019, pp. 393–403.
- 50
- [Cha+20] P. E. Chang, W. J. Wilkinson, M. E. Khan, and A. Solin. "Fast Variational Learning in State-Space Gaussian Process Models". In: *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. Sept. 2020, pp. 1–6.
- 51
- [Cha21] S. H. Chan. *Introduction to Probability for Data Science*. Michigan Publishing, 2021.
- 52
- [Che+05] G. Chechik, A. Globerson, N. Tishby, and Y. Weiss. "Information Bottleneck for Gaussian Variables". In: *JMLR* 6.Jan (2005), pp. 165–188.
- 53
- [Che14] C. Chekuri. "Treewidth, Applications, and some Recent Developments". In: *NIPS Tutorial*. 2014.
- 54
- [Che+15] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs". In: *ICLR*. 2015.
- 55
- [Che+17] T. Che, Y. Li, R. Zhang, R. D. Hjelm, W. Li, Y. Song, and Y. Bengio. "Maximum-likelihood augmented discrete generative adversarial networks". In: *arXiv preprint arXiv:1702.07983* (2017).

- 1
- 2 [Che17] C. Chelba. *Language Modeling in the Era of Abundant Data*. AI with the best. 2017.
- 3 [Che+17a] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”. In: *IEEE PAMI* (2017).
- 4 [Che+17b] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. “Variational Lossy Autoencoder”. In: *ICLR*. 2017.
- 5 [Che+17c] X. Chen, N. Mishra, M. Rohaninejad, and P. Abbeel. “PixelSNAIL: An Improved Autoregressive Generative Model”. In: (Dec. 2017). arXiv: 1712.09763 [cs.LG].
- 6 [Che+17d] V. Chernozhukov, D. Chetverikov, M. Demirer, E. Duflo, C. Hansen, and W. Newey. “Double/Debiased/Neyman Machine Learning of Treatment Effects”. In: *American Economic Review* 107.5 (2017), pp. 261–65.
- 7 [Che+17e] V. Chernozhukov, D. Chetverikov, M. Demirer, E. Duflo, C. Hansen, W. Newey, and J. Robins. “Double/Debiased Machine Learning for Treatment and Structural parameters”. In: *The Econometrics Journal* (2017).
- 8 [Che+18a] C. Chen, W. Wang, Y. Zhang, Q. Su, and L. Carin. “A convergence analysis for a class of practical variance-reduction stochastic gradient MCMC”. In: *Sci. China Inf. Sci.* 62.1 (Dec. 2018), p. 12101.
- 9 [Che+18b] C. Chen, O. Li, C. Tao, A. J. Barnett, J. Su, and C. Rudin. “This looks like that: deep learning for interpretable image recognition”. In: *arXiv preprint arXiv:1806.10574* (2018).
- 10 [Che+18c] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. “Neural Ordinary Differential Equations”. In: *NIPS*. 2018.
- 11 [Che+18d] X. Cheng, N. S. Chatterji, P. L. Bartlett, and M. I. Jordan. “Underdamped Langevin MCMC: A non-asymptotic analysis”. In: *COLT*. 2018.
- 12 [Che+19] R. T. Q. Chen, J. Behrmann, D. Duvenaud, and J.-H. Jacobsen. “Residual Flows for Invertible Generative Modeling”. In: *NIPS*. 2019.
- 13 [Che+20a] M. Chen, Y. Wang, T. Liu, Z. Yang, X. Li, Z. Wang, and T. Zhao. “On computation and generalization of generative adversarial imitation learning”. In: *arXiv preprint arXiv:2001.02792* (2020).
- 14 [Che+20b] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan. “WaveGrad: Estimating Gradients for Waveform Generation”. In: *arXiv preprint arXiv:2009.00719* (2020).
- 15 [Che95] Y. Cheng. “Mean shift, mode seeking, and clustering”. In: *IEEE PAMI* 17.8 (1995).
- 16 [Chi14] S. Chiappa. “Explicit-Duration Markov Switching Models”. In: *Foundations and Trends in Machine Learning* 7.6 (2014), pp. 803–886.
- 17 [Chi21] R. Child. “Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images”. In: *ICLR*. 2021.
- 18 [Cho02] N. Chopin. “A Sequential Particle Filter Method for Static Models”. In: *Biometrika* 89.3 (2002), pp. 539–551.
- 19 [Cho11] M. J. Choi. “Trees and Beyond: Exploiting and Improving Tree-Structured Graphical Models”. PhD thesis. MIT, 2011.
- 20 [Cho+15] Y. Chow, A. Tamar, S. Mannor, and M. Pavone. “Risk-Sensitive and Robust Decision-Making: A CVaR Optimization Approach”. In: *NIPS*. 2015, pp. 1522–1530.
- 21 [Cho21] F. Chollet. *Deep learning with Python (second edition)*. Manning, 2021.
- 22 [Chr57] N. Chomsky. *Syntactic Structures*. Mouton, 1957.
- 23 [Chr+21] R. Christiansen, N. Pfister, M. E. Jakobsen, N. Gnecco, and J. Peters. “A causal framework for distribution generalization”. In: *IEEE PAMI* (2021).
- 24 [Chu+15] J. Chung, K. Kastner, L. Dinh, K. Goel, A. Courville, and Y. Bengio. “A Recurrent Latent Variable Model for Sequential Data”. In: *NIPS*. 2015.
- 25 [Chu+18] K. Chua, R. Calandra, R. McAllister, and S. Levine. “Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models”. In: *NIPS*. 2018.
- 26 [Chw+15] K. Chwialkowski, A. Ramdas, D. Sejdinovic, and A. Gretton. “Fast Two-Sample Testing with Analytic Representations of Probability Measures”. In: *NIPS*. 2015.
- 27 [Cox80] D. R. Cox and V. Isham. *Point processes*. Vol. 12. CRC Press, 1980.
- 28 [CJ21] A. D. Cobb and B. Jalaian. “Scaling Hamiltonian Monte Carlo inference for Bayesian neural networks with symmetric splitting”. In: *UAI*. Vol. 161. Proceedings of Machine Learning Research. PMLR, 2021, pp. 675–685.
- 29 [CK05] M. Collins and T. Koo. “Discriminative Reranking for Natural Language Parsing”. In: *Proc. ACL*. 2005.
- 30 [CK07] J. F. Commandeur and S. J. Koopman. *An Introduction to State Space Time Series Analysis (Practical Econometrics)*. en. 1st ed. Oxford University Press, Aug. 2007.
- 31 [CK94a] D. Chakrabarty and S. Khanna. “Better and simpler error analysis of the Sinkhorn-Knopp algorithm for matrix scaling”. In: *Mathematical Programming* 188.1 (2021), pp. 395–407.
- 32 [CK94b] D. Card and A. B. Krueger. “Minimum Wages and Employment: A Case Study of the Fast-Food Industry in New Jersey and Pennsylvania”. In: *American Economic Review* 84.4 (1994), pp. 772–793.
- 33 [CK94b] C. Carter and R. Kohn. “On Gibbs sampling for state space models”. In: *Biometrika* 81.3 (1994), pp. 541–553.
- 34 [CK17] L. Chen, A. Krause, and A. Karbasi. “Interactive Submodular Bandit”. In: *NIPS*. 2017, pp. 141–152.
- 35 [CL00] R. Chen and S. Liu. “Mixture Kalman Filters”. In: *J. Royal Stat. Soc. B* (2000).
- 36 [CL07] L. Carvalho and C. Lawrence. “Centroid estimation in discrete high-dimensional spaces with applications in biology”. In: *PNAS* 105.4 (2007).
- 37 [CL11] O. Chapelle and L. Li. “An empirical evaluation of Thompson sampling”. In: *NIPS*. 2011.
- 38 [CL18] Z. Chen and B. Liu. *Lifelong Machine Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan Claypool, 2018.
- 39 [CL96] B. P. Carlin and T. A. Louis. *Bayes and Empirical Bayes Methods for Data Analysis*. Chapman and Hall, 1996.
- 40 [Cla20] P. Clavier. “Sum-Product Network in the context of missing data”. en. MA thesis. KTH, 2020.
- 41 [Cla21] A. Clayton. *Bernoulli’s Fallacy: Statistical Illogic and the Crisis of Modern Science*. en. Columbia University Press, Aug. 2021.
- 42 [CLD18] C. Cremer, X. Li, and D. Duvenaud. “Inference Suboptimality in Variational Autoencoders”. In: *ICML*. 2018.
- 43 [Clo+19] J. R. Clough, I. Oksuz, E. Puyol-Antón, B. Ruijsink, A. P. King, and J. A. Schnabel. “Global and local interpretability for cardiac MRI classification”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2019, pp. 656–664.
- 44 [Clo20] Cloudera. *Causality for ML*. 2020.
- 45 [CLV19] M. Cox, T. van de Laar, and B. de Vries. “A factor graph approach to automated design of Bayesian signal processing algorithms”. In: *Int. J. Approx. Reason.* 104 (Jan. 2019), pp. 185–204.
- 46 [CLW18] Y. Chen, L. Li, and M. Wang. “Scalable Bilinear  $\pi$ -Learning Using State and Action Features”. In: *ICML*. 2018, pp. 833–842.
- 47 [CM09] O. Cappé and E. Moulines. “Online EM Algorithm for Latent Data Models”. In: *J. of Royal Stat. Soc. Series B* 71.3 (2009), pp. 593–613.
- 48 [CM18] D. Crisan and J. Míguez. “Nested particle filters for online parameter estimation in discrete-time state-space Markov models”. en. In: *Bernoulli* 24.4A (Nov. 2018), pp. 3039–3086.
- 49 [CMD17] C. Cremer, Q. Morris, and D. Duvenaud. “Reinterpreting Importance-Weighted Autoencoders”. In: *ICLR Workshop*. 2017.
- 50 [CMO08] J. Cornebise, E. Moulines, and J. Olsson. “Adaptive methods for sequential importance sampling with application to state space models”. In: *Statistics and Computing* (Mar. 2008).
- 51 [CMR05] O. Cappe, E. Moulines, and T. Ryden. *Inference in Hidden Markov Models*. Springer, 2005.
- 52 [CN01] H. Choset and K. Nagatani. “Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization”. In: *IEEE Trans. Robotics and Automation* 17.2 (2001).
- 53 [CNW20] M. Collier, A. Nazabal, and C. K. I. Williams. “VAEs in the Presence of Missing Data”. In: *ICML Workshop on the Art of Learning with Missing Values*. June 2020.
- 54 [CO06] N. Chater and M. Oaksford. “Mental mechanisms”. In: *Information sampling and adaptive cognition* (2006), pp. 210–236.
- 55 [COB18] L. Chizat, E. Oyallon, and F. Bach. “On Lazy Training in Differentiable Programming”. In: (Dec. 2018). arXiv: 1812.07956 [math.OC].
- 56 [Cor+12] G. Cormode, M. Garofalakis, P. J. Haas, and C. Jermaine. “Synopses for massive data: Samples, histograms, wavelets, sketches”. In: *Foundations and Trends in Databases* 4.1–3 (2012), pp. 1–294.
- 57 [Cor17] G. Cormode. “Data sketching”. In: *Communications of the ACM* 60.9 (2017), pp. 48–55.
- 58 [Cor+59] J. Cornfield, W. Haenszel, E. C. Hammond, A. M. Lilienfeld, M. B. Shimkin, and E. L. Wynder. “Smoking and Lung Cancer: Recent Evidence and a Discussion of Some Questions”. In: *JNCI: Journal of the National Cancer Institute* 22.1 (Jan. 1959), pp. 173–203. eprint: <https://academic.oup.com/jnci/article-pdf/22/1/173/2704718/22-1-173.pdf>.

- 1 [Cor+87] A Corana, M Marchesi, C Martini, and S Ridella. “Minimizing Multimodal Functions of Continuous Variables with the “Simulated Annealing” Algorithm”. In: *ACM Trans. Math. Softw.* 13.3 (Sept. 1987), pp. 262–280.
- 2 [Ceu16] Council of European Union. *General Data Protection Regulation*. 2016.
- 3 [Cov99] T. M. Cover. *Elements of information theory*. John Wiley & Sons, 1999.
- 4 [Cow+99] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.
- 5 [CP20a] R. Chen and I. C. Paschalidis. *Distributionally Robust Learning*. NOW Foundations and Trends in Optimization, 2020.
- 6 [CP20b] N. Chopin and O. Papaspiliopoulos. *An Introduction to Sequential Monte Carlo* (*Springer Series in Statistics*). en. 1st ed. Springer, Oct. 2020.
- 7 [CPD17] P. Constantiniou and A Philip Dawid. “Extended conditional independence and applications in causal inference”. en. In: *Ann. Stat.* 45.6 (Dec. 2017), pp. 2618–2653.
- 8 [CPS10] C. Carvalho, N. Polson, and J. Scott. “The horseshoe estimator for sparse signals”. In: *Biometrika* 97.2 (2010), p. 465.
- 9 [CRK19] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter. “Certified adversarial robustness via randomized smoothing”. In: *arXiv preprint arXiv:1902.02918* (2019).
- 10 [Cro+11] D. F. Crouse, P. Willett, K. Pattipati, and L. Svensson. “A look at Gaussian mixture reduction algorithms”. In: *14th International Conference on Information Fusion*. July 2011, pp. 1–8.
- 11 [CS04] I. Csiszár and P. C. Shields. “Information theory and statistics: A tutorial”. In: (2004).
- 12 [CS09] Y. Cho and L. K. Saul. “Kernel Methods for Deep Learning”. In: *NIPS*. 2009, pp. 342–350.
- 13 [CS18] P. Chaudhari and S. Soatto. “Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks”. In: *ICLR*. 2018.
- 14 [Csi67] I. Csiszar. “Information-Type Measures of Difference of Probability Distributions and Indirect Observations”. In: *Acta Scientiarum Mathematicarum Hungarica* 2 (1967), pp. 299–318.
- 15 [CSN21] J. Couillon, L. South, and C. Nemeth. “Stochastic Gradient MCMC with Multi-Armed Bandit Tuning”. In: (May 2021). arXiv: 2105.13059 [[stat.CO](#)].
- 16 [CT06] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. 2nd edition. John Wiley, 2006.
- 17 [CT+19] M. F. Cusumano-Towner, F. A. Saad, A. K. Lew, and V. K. Mansinghka. “Gen: a general-purpose probabilistic programming system with programmable inference”. In: *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI 2019. Phoenix, AZ, USA: Association for Computing Machinery, June 2019, pp. 221–236.
- 18 [CT91] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley, 1991.
- 19 [CTM17] M. F. Cusumano-Towner and V. K. Mansinghka. “AIDE: An algorithm for measuring the accuracy of probabilistic inference algorithms”. In: *NIPS*. 2017.
- 20 [CTN17] Y. Chali, M. Tanvee, and M. T. Nayem. “Towards abstractive multi-document summarization using submodular function-based framework, sentence compression and merging”. In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. 2017, pp. 418–424.
- 21 [CTS78] C. Cunningham, E. A. Thompson, and M. H. Skolnick. “Probability functions in complex pedigrees”. In: *Advances in Applied Probability* 10 (1978), pp. 26–61.
- 22 [CUH16] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)”. In: *ICLR*. 2016.
- 23 [Cun83] W. H. Cunningham. “Decomposition of submodular functions”. In: *Combinatorica* 3.1 (1983), pp. 53–68.
- 24 [Cut13] M. Cuturi. “Sinkhorn Distances: Lightspeed Computation of Optimal Transportation Distances”. In: *NIPS*. 2013.
- 25 [CW07] C. M. Carvalho and M. West. “Dynamic Matrix-Variate Graphical Models”. In: *Bayesian Analysis* 2.1 (2007), pp. 69–98.
- 26 [CW16] T. Cohen and M. Welling. “Group Equivariant Convolutional Networks”. en. In: *ICML*. June 2016, pp. 2990–2999.
- 27 [CWG20] D. C. Castro, I. Walker, and B. Glocker. “Causality matters in medical imaging”. en. In: *Nat. Commun.* 11.1 (July 2020), p. 3673.
- 28 [CY20] G. Cormode and K. Yi. *Small Summaries for Big Data*. Cambridge University Press, 2020.
- 29 [Cza+20] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison. “DeepFactors: Real-Time Probabilistic Dense Monocular SLAM”. In: *ICRA*. 2020.
- 30 [CZG20] B. Charpentier, D. Zügner, and S. Günnemann. “Posterior network: Uncertainty estimation without ood samples via density-based pseudo-counts”. In: *arXiv preprint arXiv:2006.09239* (2020).
- 31 [D'A+21] A. D'Amour, P. Ding, A. Feller, L. Lei, and J. Sekhon. “Overlap in observational studies with high-dimensional covariates”. In: *Journal of Econometrics* 221.2 (2021), pp. 644–654.
- 32 [Dag+21] N. Dagan, N. Barda, E. Kepten, O. Miron, S. Perchik, M. A. Katz, M. A. Hernán, M. Lipsitch, B. Reis, and R. D. Balicer. “BNT162b2 mRNA Covid-19 Vaccine in a Nationwide Mass Vaccination Setting”. In: *New England Journal of Medicine* 384.15 (2021), pp. 1412–1423. eprint: <https://doi.org/10.1056/NEJMoa2101765>.
- 33 [Dai+17] H. Dai, B. Dai, Y.-M. Zhang, S. Li, and L. Song. “Recurrent Hidden Semi-Markov Model”. In: *ICLR*. 2017.
- 34 [Dai+18] B. Dai, A. Shaw, L. Li, L. Xiao, N. He, Z. Liu, J. Chen, and L. Song. “SBEED: Convergent Reinforcement Learning with Nonlinear Function Approximation”. In: *ICML*. 2018, pp. 1133–1142.
- 35 [Dai+19a] B. Dai, H. Dai, A. Gretton, L. Song, D. Schuurmans, and N. He. “Kernel exponential family estimation via doubly dual embedding”. In: *AISTATS*. PMLR, 2019, pp. 2321–2330.
- 36 [Dai+19b] B. Dai, Z. Liu, H. Dai, N. He, A. Gretton, L. Song, and D. Schuurmans. “Exponential family estimation via adversarial dynamics embedding”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 10979–10990.
- 37 [Dai+19c] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov. “Transformer-XL: Attentive Language Models beyond a Fixed-Length Context”. In: *Proc. ACL*. 2019, pp. 2978–2988.
- 38 [Dai+20a] C. Dai, J. Heng, P. E. Jacob, and N. Whiteley. “An invitation to sequential Monte Carlo samplers”. In: (July 2020). arXiv: 2007.11936 [[stat.CO](#)].
- 39 [Dai+20b] Z. Dai, G. Lai, Y. Yang, and Q. V. Le. “Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing”. In: *NIPS*. June 2020.
- 40 [Dai+04] N. Dalvi, P. Domingos, S. Sanghavi, and D. Verma. “Adversarial classification”. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2004, pp. 99–108.
- 41 [Dar03] A. Darwiche. “A Differential Approach to Inference in Bayesian Networks”. In: *J. ACM* 50.3 (May 2003), pp. 280–305.
- 42 [Dar09] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge, 2009.
- 43 [Dar+11] S. Darolles, Y. Fan, J.-P. Florens, and E. Renault. “Nonparametric instrumental regression”. In: *Econometrica* 79.5 (2011), pp. 1541–1565.
- 44 [Dar80] R. A. Darton. “Rotation in Factor Analysis”. In: *Journal of the Royal Statistical Society. Series D (The Statistician)* 29.3 (1980), pp. 167–194.
- 45 [Dau07] H. Daume. “Fast search for Dirichlet process mixture models”. In: *AISTATS*. 2007.
- 46 [Dav+04] T. A. Davis, J. R. Gilbert, S. I. Larimore, and E. G. Ng. “A Column Approximate Minimum Degree Ordering Algorithm”. In: *ACM Trans. Math. Softw.* 30.3 (Sept. 2004), pp. 353–376.
- 47 [Dav+18] T. R. Davidson, L. Falorsi, N. De Cao, T. Kipf, and J. M. Tomczak. “Hyperspherical Variational Auto-Encoders”. In: *UAI*. 2018.
- 48 [Daw00] A. P. Dawid. “Causal Inference Without Counterfactuals”. In: *JASA* 95.450 (2000), pp. 407–424.
- 49 [Daw02] A. P. Dawid. “Influence diagrams for causal modelling and inference”. In: *Intl. Stat. Review* 70 (2002). Corrections p437, pp. 161–189.
- 50 [Daw15] A. P. Dawid. “Statistical Causality from a Decision-Theoretic Perspective”. In: *Annu. Rev. Stat. Appl.* 2.1 (Apr. 2015), pp. 273–303.
- 51 [Daw82] A. P. Dawid. “The Well-Calibrated Bayesian”. In: *JASA* 77.379 (1982), pp. 605–610.
- 52 [Daw92] A. P. Dawid. “Applications of a general propagation algorithm for probabilistic expert systems”. In: *Statistics and Computing* 2 (1992), pp. 25–36.
- 53 [Dax+21] E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig. “Laplace Redux—Effortless Bayesian Deep Learning”. In: *NIPS*. 2021.
- 54 [Day95] P. Dayan, G. Hinton, R. Neal, and R. Zemel. “The Helmholtz machine”. In: *Neural Networks* 9.8 (1995).
- 55 [DB+13] P. De Blasi, S. Favaro, A. Lijoi, R. H. Mena, J. Prünster, and M. Ruggiero. “Are Gibbs-type priors the most natural generalization of the Dirichlet process?” In: *IEEE PAMI* 37.2 (2013), pp. 212–229.

- 1
- 2 [DBB20] S. Daulton, M. Balandat, and E. Bakshy. "Differentiable Expected Hypervolume Improvement for Parallel Multi-Objective Bayesian Optimization". In: *NIPS*. 2020.
- 3 [DBP19] C. Durkan, A. Bekasov, and I. M. G. Papamakarios. "Neural Spline Flows". In: *NIPS*. 2019.
- 4 [DBW20] I. A. Delbridge, D. S. Bindel, and A. G. Wilson. "Randomly Projected Additive Gaussian Processes for Regression". In: *International Conference on Machine Learning*. 2020.
- 5 [DC20] H.-D. Dau and N. Chopin. "Waste-free Sequential Monte Carlo". In: (Nov. 2020). arXiv: 2011.02328 [stat.CO].
- 6 [DCF+15] E. Denton, S. Chintala, R. Fergus, et al. "Deep generative image models using a Laplacian pyramid of adversarial networks". In: *NIPS*. 2015.
- 7 [DE00] R. Dahlhaus and M. Eichler. "Causality and graphical models for time series". In: *Highly structured stochastic systems*. Ed. by P. Green, N. Hjort, and S. Richardson. Oxford University Press, 2000.
- 8 [DE04] J. Dow and J. Endersby. "Multinomial probit and multinomial logit: a comparison of choice models for voting research". In: *Electoral Studies* 23.1 (2004), pp. 107–122.
- 9 [Dec03] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.
- 10 [Dec19] R. Dechter. "Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms (2nd edn)". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 7.3 (2019), pp. 1–191.
- 11 [Dec96] R. Dechter. "Bucket elimination: a unifying framework for probabilistic inference". In: *UAI*. 1996.
- 12 [DeG70] M. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, 1970.
- 13 [DEL20] H. M. Dolatabadi, S. Erfani, and C. Leckie. "Invertible Generative Modeling using Linear Rational Splines". In: *AISTATS*. 2020, pp. 4236–4246.
- 14 [Del+21] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardi, G. Slabaugh, and T. Tuytelaars. "A continual learning survey: Defying forgetting in classification tasks". en. In: *IEEE PAMI* (Feb. 2021).
- 15 [Den+02] D. Denison, C. Holmes, B. Mallick, and A. Smith. *Bayesian methods for nonlinear classification and regression*. Wiley, 2002.
- 16 [Den+20] Y. Deng, A. Bakhtin, M. Ott, A. Szlam, and M. Ranzato. "Residual Energy-Based Models for Text Generation". In: *International Conference on Learning Representations*. 2020.
- 17 [Dev+19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *NAACL*. 2019.
- 18 [Dev+21] L. Devlin, P. Horridge, P. L. Green, and S. Maskell. "The No-U-Turn Sampler as a Proposal Distribution in a Sequential Monte Carlo Sampler with a Near-Optimal L-Kernel". In: (Aug. 2021). arXiv: 2108.02498 [stat.CO].
- 19 [Dev85] P. A. Devijver. "Baum's forward-backward algorithm revisited". In: *Pattern Recognition Letters* 3.6 (1985), pp. 369–373.
- 20 [Dex] Dex: Research language for array processing in the Haskell/ML family. [https://github.com/google-research/dex\\_lang](https://github.com/google-research/dex_lang). 2019.
- 21 [DF18] E. Denton and R. Fergus. "Stochastic Video Generation with a Learned Prior". In: *ICML*. 2018.
- 22 [DF19] X. Ding and D. J. Freedman. "Learning Deep Generative Models with Annealed Importance Sampling". In: (June 2019). arXiv: 1906.04904 [stat.ML].
- 23 [DFF21] S. Dozinski, U. Feige, and M. Feldman. "Are Gross Substitutes a Substitute for Submodular Valuations?". In: *arXiv preprint arXiv:2102.13343* (2021).
- 24 [DFR15] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. "Gaussian Processes for Data-Efficient Learning in Robotics and Control". en. In: *IEEE PAMI* 37.2 (Feb. 2015), pp. 408–423.
- 25 [DFS16] A. Daniely, R. Frostig, and Y. Singer. "Toward Deeper Understanding of Neural Networks: The Power of Initialization and a Dual View on Expressivity". In: *NIPS*. 2016, pp. 2253–2261.
- 26 [DG17] P. Dabkowski and Y. Gal. "Real time image saliency for black box classifiers". In: *NeurIPS* (2017).
- 27 [DG84] P. J. Diggle and R. J. Gratton. "Monte Carlo methods of inference for implicit statistical models". In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1984), pp. 193–227.
- 28 [DGA00] A. Doucet, S. Godsill, and C. Andrieu. "On sequential Monte Carlo Sampling Methods for Bayesian Filtering". In: *Statistics and Computing* 10.3 (2000), pp. 197–208.
- 29 [DGF20] Y. Dubois, J. Gordon, and A. Y. Foong. *Neural Process Family*. <http://yanndub.github.io/Neural-Process-Family/>. 2020.
- 30 [DGK01] A. Doucet, N. Gordon, and V. Krishnamurthy. "Particle Filters for State Estimation of Jump Markov Linear Systems". In: *IEEE Trans. on Signal Processing* 49.3 (2001), pp. 613–624.
- 31 [DH22] F. Dellaert and S. Hutchinson. *Introducion to Robotics and Perception*. 2022.
- 32 [DHK14] A. Deshpande, L. Hellerstein, and D. Kletenik. "Approximation algorithms for stochastic boolean function evaluation and stochastic submodular set cover". In: *Proceedings of the twenty-fifth annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2014, pp. 1453–1466.
- 33 [Dia88] P. Diaconis. "Sufficiency as statistical symmetry". In: *Proceedings of the AMS Centennial Symposium*. 1988, pp. 15–26.
- 34 [Die+07] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber. "Fast Direct Multiple Shooting Algorithms for Optimal Robot Control". In: *Lecture Notes in Control and Inform. Sci.* 340 (July 2007).
- 35 [Die10] L. Dietz. *Directed Factor Graph Notation for Generative Models*. Tech. rep. MPI, 2010.
- 36 [Die+17] A. B. Dieng, D. Tran, R. Ranganath, J. Paisley, and D. Blei. "Variational Inference via chi Upper Bound Minimization". In: *NIPS*. Curran Associates, Inc., 2017, pp. 2732–2741.
- 37 [Die+19a] A. B. Dieng, Y. Kim, A. M. Rush, and D. M. Blei. "Avoiding Latent Variable Collapse With Generative Skip Models". In: *AISTATS*. 2019.
- 38 [Die+19b] A. B. Dieng, F. J. Ruiz, D. M. Blei, and M. K. Titsias. "Prescribed generative adversarial networks". In: *arXiv preprint arXiv:1910.04302* (2019).
- 39 [Dik+20] N. Dikkala, G. Lewis, L. Mackey, and V. Syrgkanis. "Minimax Estimation of Conditional Moment Models". In: *Advances in Neural Information Processing Systems*. 2020.
- 40 [Din+17] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. "Sharp Minima Can Generalize For Deep Nets". In: (2017). arXiv: 1703.04933 [cs.LG].
- 41 [Din+21] S. U. Din, J. Shao, J. Kumar, C. B. Mawuli, S. M. H. Mahmud, Y. Zhang, and Q. Yang. "Data-stream classification with novel class detection: A review, comparison and challenges". In: *Knowl. Inf. Syst.* 63.9 (Sept. 2021), pp. 2231–2276.
- 42 [DJ11] A. Doucet and A. M. Johansen. "A Tutorial on Particle Filtering and Smoothing: Fifteen years later". In: *Handbook of Nonlinear Filtering*. Ed. by D Crisan and B Rozovskii. 2011.
- 43 [DJ15] S. Dray and J. Josse. "Principal component analysis with missing values: a comparative survey of methods". In: *Plant Ecol.* 216.5 (May 2015), pp. 657–667.
- 44 [DJK18] J. Djolonga, S. Jegelka, and A. Krause. "Provable Variational Inference for Constrained Log-Submodular Models". In: *NeurIPS*. 2018.
- 45 [DJL21] A. J. DeGrave, J. Janizek, and S.-I. Lee. "AI for radiographic COVID-19 detection selects shortcuts over signal". en. In: *Nature Machine Intelligence* 3.7 (May 2021), pp. 610–619.
- 46 [DK12] J. Durbin and S. J. Koopman. *Time Series Analysis by State Space Methods: Second Edition*. en. Revised ed. edition. Oxford University Press, July 2012.
- 47 [DK14] J. Djolonga and A. Krause. "From map to marginals: Variational inference in bayesian submodular models". In: *Advances in Neural Information Processing Systems*. 2014, pp. 244–252.
- 48 [DK15a] J. Djolonga and A. Krause. "Scalable variational inference in log-submodular models". In: *International Conference on Machine Learning*. PMLR, 2015, pp. 1804–1813.
- 49 [DK15b] G. Durrett and D. Klein. "Neural CRF Parsing". In: *Proc. ACL*. 2015.
- 50 [DKB15] L. Dinh, D. Krueger, and Y. Bengio. "NICE: Non-linear Independent Components Estimation". In: *ICLR*. 2015.
- 51 [DKD16] J. Donahue, P. Krahenbuhl, and T. Darrell. "Adversarial feature learning". In: *arXiv preprint arXiv:1605.09782* (2016).
- 52 [DL09] P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for AI*. Morgan & Claypool, 2009.
- 53 [DL10] J. V. Dillon and G. Lebanon. "Stochastic Composite Likelihood". In: *J. Mach. Learn. Res.* 11 (2010), pp. 2597–2633.
- 54 [DL13] A. Damianou and N. Lawrence. "Deep Gaussian Processes". en. In: *AISTATS*. Apr. 2013, pp. 207–215.
- 55 [DL93] P. Dagum and M. Luby. "Approximating probabilistic inference in Bayesian belief networks is NP-hard". In: *Artificial Intelligence* 60 (1993), pp. 141–153.
- 56 [DLB17] C. Dann, T. Lattimore, and E. Brunskill. "Unifying PAC and Regret: Uniform PAC Bounds for Episodic Reinforcement Learning". In: *NIPS*. 2017, pp. 5717–5727.
- 57 [DLM09] H. Daumé III, J. Langford, and D. Marcu. "Search-based Structured Prediction". In: *MLJ* 75.3 (2009), pp. 297–325.
- 58 [DLM99] B. Delyon, M. Lavielle, and E. Moulines. "Convergence of a stochastic approximation version of the EM algorithm". In: *Annals of Statistics* 27.1 (1999), pp. 94–128.

- 1
- [DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *J. of the Royal Statistical Society, Series B* 34 (1977), pp. 1–38.
- 2
- [DLT21] M. De Lange and T. Tuytelaars. "Continual Prototype Evolution: Learning Online from Non-Stationary Data Streams". In: *ICCV*. 2021.
- 3
- [DM01] D. van Dyk and X.-L. Meng, "The Art of Data Augmentation". In: *J. Computational and Graphical Statistics* 10.1 (2001), pp. 1–50.
- 4
- [DM19a] Y. Du and I. Mordatch. "Implicit Generation and Generalization in Energy-Based Models". In: (Mar. 2019). arXiv: 1903.08689 [cs.LG].
- 5
- [DM19b] Y. Du and I. Mordatch. "Implicit Generation and Modeling with Energy Based Models". In: *NIPS*. 2019, pp. 3608–3618.
- 6
- [DMDJ12] P. Del Moral, A. Doucet, and A. Jasra. "An adaptive sequential Monte Carlo method for approximate Bayesian computation". In: *Stat. Comput.* 22.5 (Sept. 2012), pp. 1009–1020.
- 7
- [DMKM22] G. Duran-Martin, A. Kara, and K. Murphy. "Efficient Online Bayesian Inference for Neural Bandits". In: *AISTATS*. 2022.
- 8
- [DMM17] A. P. Dawid, M. Musio, and R. Murtas. "The probability of causation". In: *Law, Probability and Risk* 16.4 (Dec. 2017), pp. 163–179.
- 9
- [DMP18] C. Donahue, J. McAuley, and M. Puckette. "Adversarial Audio Synthesis". In: *International Conference on Learning Representations*. 2018.
- 10
- [DMV15] S. Dash, D. M. Malioutov, and K. R. Varshney. "Learning interpretable classification rules using sequential rowsampling". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2015, pp. 3337–3341.
- 11
- [DN21] P. Dhariwal and A. Nichol. "Diffusion Models Beat GANs on Image Synthesis". In: (May 2021). arXiv: 2105.05233 [cs.LG].
- 12
- [Dnp] .
- 13
- [DNR11] D. Duvenaud, H. Nickisch, and C. E. Rasmussen. "Additive Gaussian Processes". In: *NIPS*. 2011.
- 14
- [Dom+06] P. Domingos, S. Kok, H. Poon, M. Richardson, and P. Singla. "Unifying Logical and Statistical AI". In: *IJCAI*. 2006.
- 15
- [Dom+19] A.-K. Dombrowski, M. Alber, C. J. Anders, M. Aekermann, K.-R. Müller, and P. Kessel. "Explanations can be manipulated and geometry is to blame". In: *arXiv preprint arXiv:1906.07983* (2019).
- 16
- [Don+17a] K. Dong, D. Eriksson, H. Nickisch, D. Bindel, and A. G. Wilson. "Scalable Log Determinants for Gaussian Process Kernel Learning". In: *NIPS*. 2017, pp. 6327–6337.
- 17
- [Don+17b] Y. Dong, H. Su, J. Zhu, and F. Bao. "Towards interpretable deep neural networks by leveraging adversarial examples". In: *arXiv preprint arXiv:1708.05493* (2017).
- 18
- [Dor+16] V. Dorie, M. Harada, N. B. Carnegie, and J. Hill. "A flexible, interpretable framework for assessing sensitivity to unmeasured confounding". In: *Statistics in Medicine* 35.20 (2016), pp. 3453–3470. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/sim.6973>.
- 19
- [Doy+07] K. Doya, S. Ishii, A. Pouget, and R. P. N. Rao, eds. *Bayesian Brain: Probabilistic Approaches to Neural Coding*. MIT Press, 2007.
- 20
- [DR11] M. P. Deisenroth and C. E. Rasmussen. "PILCO: A Model-Based and Data-Efficient Approach to Policy Search". In: *ICML*. 2011.
- 21
- [DR17] G. K. Dziugaite and D. M. Roy. "Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data". In: *UAI*. 2017.
- 22
- [DRG15] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani. "Training generative neural networks via Maximum Mean Discrepancy optimization". In: *ICML*. 2015.
- 23
- [Dru08] J. Drugowitsch. *Bayesian linear regression*. Tech. rep. U. Rochester, 2008.
- 24
- [DS15] J. Dahlin and T. B. Schön. "Getting Started with Particle Metropolis-Hastings for Inference in Nonlinear Dynamical Models". In: *J. Stat. Softw.* (Nov. 2015).
- 25
- [DS18] J. Domke and D. R. Sheldon. "Importance Weighting and Variational Inference". In: *NIPS*. Curran Associates, Inc., 2018, pp. 4474–4483.
- 26
- [DS19] J. Donahue and K. Simonyan. "Large scale adversarial representation learning". In: *arXiv preprint arXiv:1907.02544* (2019).
- 27
- [DSDB17] L. Dinh, J. Sohl-Dickstein, and S. Bengio. "Density estimation using Real NVP". In: *ICLR*. 2017.
- 28
- [DSK16] V. Dumoulin, J. Shlens, and M. Kudlur. "A Learned Representation For Artistic Style". In: (2016).
- 29
- [DSZ16] A. Datta, S. Sen, and Y. Zick. "Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems". In: *2016 IEEE symposium on security and privacy (SP)*. IEEE. 2016, pp. 598–617.
- 30
- [DTK16] J. Djolonga, S. Tschiatschek, and A. Krause. "Variational inference in mixed probabilistic submodular models". In: *Advances in Neural Information Processing Systems*. 2016, pp. 1759–1767.
- 31
- [Du+16] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song. "Recurrent marked temporal point processes: Embedding event history to vector". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 1555–1564.
- 32
- [Du+18] J. Du, S. Ma, Y.-C. Wu, S. Kar, and J. M. F. Moura. "Convergence Analysis of Distributed Inference with Vector-Valued Gaussian Belief Propagation". In: *JMLR* 18.172 (2018), pp. 1–38.
- 33
- [Du+19] S. Du, K. Hou, B. Póczos, R. Salakhutdinov, R. Wang, and K. Xu. "Graph Neural Tangent Kernel: Fusing Graph Neural Networks with Graph Kernels". In: (May 2019). arXiv: 1905.13192 [cs.LG].
- 34
- [Du+20] Y. Du, S. Li, J. Tenenbaum, and I. Mordatch. "Improved Contrastive Divergence Training of Energy Based Models". In: *arXiv preprint arXiv:2012.01316* (2020).
- 35
- [Du+21] C. Du, Z. Gao, S. Yuan, L. Gao, Z. Li, Y. Zeng, X. Zhu, J. Xu, K. Gai, and K.-C. Lee. "Exploration in Online Advertising Systems with Deep Uncertainty-Aware Learning". In: *KDD '21. Virtual Event, Singapore: Association for Computing Machinery*, Aug. 2021, pp. 2792–2801.
- 36
- [Dua+87] S. Duane, A. Kennedy, B. Pendleton, and D. Roweth. "Hybrid Monte Carlo". In: *Physics Letters B* 195.2 (1987), pp. 216–222.
- 37
- [Dub+16] A. Dubey, S. J. Reddi, B. Póczos, A. J. Smola, E. P. Xing, and S. A. Williamson. "Variance Reduction in Stochastic Gradient Langevin Dynamics". In: *NIPS*. Vol. 29. Dec. 2016, pp. 1154–1162.
- 38
- [Dud13] J. Duda. "Asymmetric numeral systems: entropy coding combining speed of Huffman coding with compression rate of arithmetic coding". In: (Nov. 2013). arXiv: 1311.2540 [cs.IT].
- 39
- [Duf02] M. Duff. "Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes". PhD thesis. U. Mass. Dept. Comp. Sci., 2002.
- 40
- [Dum+16] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. "Adversarially learned inference". In: *arXiv preprint arXiv:1606.00704* (2016).
- 41
- [Dum+21] V. Dumoulin, N. Houlsby, U. Evci, X. Zhai, R. Goroshin, S. Gelly, and H. Larochelle. "A Unified Few-Shot Classification Benchmark to Compare Transfer and Meta Learning Approaches". In: *NIPS Datasets and Benchmarks Track*. June 2021.
- 42
- [Dur+19] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios. "Cubic-Spline Flows". In: *ICML Workshop on Invertible Neural Networks and Normalizing Flows*. 2019.
- 43
- [Dur+98] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.
- 44
- [Dus+20] M. Dusenberry, G. Jerfel, Y. Wen, Y. Ma, J. Snoek, K. Heller, B. Lakshminarayanan, and D. Tran. "Efficient and scalable bayesian neural nets with rank-1 factors". In: *International conference on machine learning*. PMLR. 2020, pp. 2782–2792.
- 45
- [Dut+21] V. Dutordoir, J. Hensman, M. van der Wilk, C. H. Ek, Z. Ghahramani, and N. Durrande. "Deep Neural Networks as Point Estimates for Deep Gaussian Processes". In: (May 2021). arXiv: 2105.04504 [stat.ML].
- 46
- [Duv+13] D. Duvenaud, J. Lloyd, R. Grosse, J. Tenenbaum, and G. Zoubin. "Structure Discovery in Nonparametric Regression through Compositional Kernel Search". In: *ICML*. Feb. 2013, pp. 1166–1174.
- 47
- [Duv14] D. Duvenaud. "Automatic Model Construction with Gaussian Processes". PhD thesis. Computational and Biological Learning Laboratory, University of Cambridge, 2014.
- 48
- [dVo04] D. P. de Farias and B. Van Roy. "On Constraint Sampling in the Linear Programming Approach to Approximate Dynamic Programming". In: *Mathematics of Operations Research* 29.3 (2004), pp. 462–478.
- 49
- [DV+17] H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A.-C. Courville. "Modulating early visual processing by language". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6594–6604.
- 50
- [DV75] M. D. Donsker and S. S. Varadhan. "Asymptotic evaluation of certain Markov process expectations for large time, I". In: *Communications on Pure and Applied Mathematics* 28.1 (1975), pp. 1–47.
- 51
- [DV99] A. P. Dawid and V. Vovk. "Prequential probability: Principles and properties". In: *Bernoulli* 5 (1999), pp. 125–162.
- 52
- [DVK17] F. Doshi-Velez and B. Kim. *Towards A Rigorous Science of Interpretable Machine Learning*. 2017. eprint: 1702.08608 [stat.ML].

- 1
- 2 [DW19] B. Dai and D. Wipf. "Diagnosing and Enhancing VAE  
Models". In: *ICLR*. 2019.
- 3 [DWS12] T. Degris, M. White, and R. S. Sutton. "Off-Policy  
Actor-Critic". In: *ICML*. 2012.
- 4 [DWW19] B. Dai, Z. Wang, and D. Wipf. "The Usual Suspects?  
Reassessing Blame for VAE Posterior Collapse". In: (Dec. 2019).  
arXiv: [1912.10702 \[cs.LG\]](#).
- 5 [DWW20] B. Dai, Z. Wang, and D. Wipf. "The Usual Suspects?  
Reassessing Blame for VAE Posterior Collapse". In: *ICML*. Ed.  
by H. D. Iii and A. Singh. Vol. 119. Proceedings of Machine  
Learning Research. PMLR, 2020, pp. 2313–2322.
- 6 [DY79] P. Diaconis and D. Ylvisaker. "Conjugate priors for ex-  
ponential distributions". In: vol. 7. 1979, pp. 269–281.
- 7
- 8 [ECM18] P. Etoori, M. Chinnakotla, and R. Ramidi. "Automatic  
Spelling Correction for Resource-Scarce Languages using  
Deep Learning". In: *Proceedings of ACL 2018, Student Re-  
search Workshop*, Melbourne, Australia: Association for Com-  
putational Linguistics, 2018, pp. 146–152.
- 9 [ED05] D. Earl and M. Deem. "Parallel tempering: Theory,  
applications, and new perspectives". In: *Phys. Chem. Chem.  
Phys.* 7 (2005), p. 3910.
- 10 [Edm69] H. P. Edmundson. "New methods in automatic extract-  
ing". In: *Journal of the ACM (JACM)* 16.2 (1969), pp. 264–  
285.
- 11 [Edm70] J. Edmonds. "Matroids, submodular functions, and cer-  
tain polyhedra". In: *Combinatorial Structures and Their Ap-  
plications* (1970), pp. 69–87.
- 12 [EFL04] E. Erosheva, S. Fienberg, and J. Lafferty. "Mixed-  
membership models of scientific publications". In: *PNAS* 101  
(2004), pp. 5220–2227.
- 13 [Efr11] B. Efron. "Tweedie's Formula and Selection Bias". en. In:  
*J. Am. Stat. Assoc.* 106.496 (2011), pp. 1602–1614.
- 14 [Efr86] B. Efron. "Why Isn't Everyone a Bayesian?" In: *The  
American Statistician* 40.1 (1986).
- 15 [EGW05] D. Ernst, P. Geurts, and L. Wehenkel. "Tree-Based  
Batch Mode Reinforcement Learning". In: *JMLR* 6 (2005),  
pp. 503–556.
- 16 [Eis16] J. Eisner. "Inside-Outside and Forward-Backward Algo-  
rithms Are Just Backprop (Tutorial Paper)". In: *EMNLP Work-  
shop on Structured Prediction for NLP*. 2016.
- 17 [Eke+13] M. Ekeberg, C. Lövkvist, Y. Lan, M. Weigt, and E.  
Aurell. "Improved contact prediction in proteins: using pseudo-  
likelihoods to infer Potts models". en. In: *Phys. Rev. E Stat.  
Nonlin. Soft Matter Phys.* 87.1 (Jan. 2013), p. 012707.
- 18 [Ele+17] S. Eleftheriadis, T. F. W. Nicholson, M. P. Deisen-  
roth, and J. Hensman. "Identification of Gaussian Process State  
Space Models". In: *NIPS*. 2017.
- 19 [EMH19] T. Elsken, J. H. Metzen, and F. Hutter. "Neural Ar-  
chitecture Search: A Survey". In: *JMLR* 20 (2019), pp. 1–21.
- 20 [EMK06] G. Elidan, I. McGraw, and D. Koller. "Residual be-  
 lief propagation: Informed scheduling for asynchronous mes-  
sage passing". In: *UAI*. 2006.
- 21 [Eng+18] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C.  
Donahue, and A. Roberts. "GANSynth: Adversarial Neural Au-  
dio Synthesis". In: *International Conference on Learning Rep-  
resentations*. 2018.
- 22 [Erh+09] D. Erhan, Y. Bengio, A. Courville, and P. Vincent.  
"Visualizing higher-layer features of a deep network". In: *Uni-  
versity of Montreal* 1341.3 (2009), p. 1.
- 23 [ERO21] P. Esser, R. Rombach, and B. Ommer. "Taming Trans-  
formers for High-Resolution Image Synthesis". In: *CVPR*. 2021.
- 24 [Eve09] G. Evensen. *Data Assimilation: The Ensemble Kalman  
Filter*. en. 2nd ed. 2009 edition. Springer, Aug. 2009.
- 25 [EW95] M. D. Escobar and M. West. "Bayesian density estima-  
tion and inference using mixtures". In: *JASA* 90.430 (1995),  
pp. 577–588.
- 26 [Ewe72] W. J. Ewens. "The sampling theory of selectively neu-  
tral alleles". In: *Theoretical population biology* 3.1 (1972),  
pp. 87–112.
- 27 [Ewe90] W. Ewens. "Population genetics theory - the past and  
the future". In: *Mathematical and Statistical Developments in  
Evolutionary Theory*. Ed. by S. Lessard. Reidel, 1990, pp. 177–  
227.
- 28 [EY09] M. Elad and I. Yavneh. "A plurality of sparse represen-  
tations is better than the sparest one alone". In: *IEEE Trans.  
on Info. Theory* 55.10 (2009), pp. 4701–4714.
- 29 [Eyk+18] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rah-  
mati, C. Xiao, A. Prakash, T. Kohno, and D. Song. "Robust  
Physical-World Attacks on Deep Learning Models". In: *CVPR*.  
2018.
- 30 [Eys+21] B. Eysenbach, A. Khazatsky, S. Levine, and R.  
Salakhutdinov. "Mismatched No More: Joint Model-Policy Opti-  
mization for Model-Based RL". In: (Oct. 2021). arXiv: [2110.02758 \[cs.LG\]](#).
- 31
- 32
- 33
- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44
- 45
- 46
- 47
- [Fad+20] S. G. Fadel, S. Mair, R. da S. Torres, and U. Brefeld.  
"Principled Interpolation in Normalizing Flows". In: (Oct.  
2020). arXiv: [2010.12059 \[stat.ML\]](#).
- [Fal17] C. Finn, P. Abbeel, and S. Levine. "Model-Agnostic  
Meta-Learning for Fast Adaptation of Deep Networks". In:  
*ICML*. 2017.
- [Fan+17] H. Fang, N. Tian, Y. Wang, M. Zhou, and M. A. Haile.  
"Nonlinear Bayesian Estimation: From Kalman Filtering to a  
Broader Horizon". In: (Dec. 2017). arXiv: [1712.01406 \[cs.SV\]](#).
- [Fan+19] C. Fang, Y. Gu, W. Zhang, and T. Zhang. "Convex  
Formulation of Overparameterized Deep Neural Networks". In:  
(Nov. 2019). arXiv: [1911.07626 \[cs.LG\]](#).
- [Fan+18] L. Faury, F. Vasile, C. Calauzènes, and O. Fercoq.  
"Neural Generative Models for Global Optimization with Gra-  
dients". In: (May 2018). arXiv: [1805.08594 \[cs.NE\]](#).
- [FB19] E. M. Feit and R. Berman. "Test & Roll: Profit-  
Maximizing A/B Tests". In: *Marketing Science* 38.6 (Nov.  
2019), pp. 1038–1058.
- [FBW21] M. Finzi, G. Benton, and A. G. Wilson. "Residual  
Pathway Priors for Soft Equivariance Constraints". In: *NIPS*.  
2021.
- [FC03] P. Fearnhead and P. Clifford. "On-line inference for hid-  
den Markov models via particle filters". en. In: *J. of Royal Stat.  
Soc. Series B* 65.4 (Nov. 2003), pp. 887–899.
- [FCR14] R. Frigola, Y. Chen, and C. E. Rasmussen. "Variational  
Gaussian Process State-Space Models". In: *NIPS*. 2014.
- [FD07a] B. Frey and D. Dueck. "Clustering by Passing Mes-  
sages Between Data Points". In: *Science* 315 (2007), 972–976.
- [FD07b] B. J. Frey and D. Dueck. "Clustering by passing  
messages between data points". In: *science* 315.5814 (2007),  
pp. 972–976.
- [FDF19] A. M. Franks, A. D'Amour, and A. Feller. "Flexible  
Sensitivity Analysis for Observational Studies Without Observ-  
able Implications". In: *Journal of the American Statistical As-  
sociation* 0.0 (2019), pp. 1–33. eprint: <https://doi.org/10.1080/01621459.2019.1604369>.
- [FDZ19] A. Fasano, D. Durante, and G. Zanella. "Scalable and  
Accurate Variational Bayes for High-Dimensional Binary Re-  
gression Models". In: (Nov. 2019). arXiv: [1911.06743 \[stat.ME\]](#).
- [FE73] M. Fischer and R. Elschlager. "The representation and  
matching of pictorial structures". In: *IEEE Trans. on Com-  
puter* 22.1 (1973).
- [Fed+18] W. Fedus, M. Rosca, B. Lakshminarayanan, A. M.  
Dai, S. Mohamed, and I. Goodfellow. "Many Paths to Equilib-  
rium: GANs Do Not Need to Decrease a Divergence at Every  
Step". In: *International Conference on Learning Representa-  
tions*. 2018.
- [Fei98] U. Feige. "A threshold of  $\ln n$  for approximating set  
cover". In: *Journal of the ACM (JACM)* (1998).
- [Fel+10] P. Felzenszwalb, R. Girshick, D. McAllester, and D.  
Ramanan. "Object Detection with Discriminatively Trained  
Part Based Models". In: *IEEE PAMI* 32.9 (2010).
- [Fel+19] M. Fellows, A. Mahajan, T. G. J. Rudner, and S.  
Whiteson. "VIREL: A Variational Inference Framework for Re-  
inforcement Learning". In: *NeurIPS*. 2019, pp. 7120–7134.
- [Fen+21] S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S.  
Vosoughi, T. Mitamura, and E. Hovy. "A Survey of Data Aug-  
mentation Approaches for NLP". In: (May 2021). arXiv: [2105.03075 \[cs.CL\]](#).
- [Fer73] T. S. Ferguson. "A Bayesian analysis of some nonpara-  
metric problems". In: *The annals of statistics* (1973), pp. 209–  
230.
- [Feu+15] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg,  
M. Blum, and F. Hutter. "Efficient and Robust Automated Ma-  
chine Learning". In: *NIPS*. 2015, pp. 2962–2970.
- [FG15] N. Fournier and A. Guillin. "On the rate of convergence  
in Wasserstein distance of the empirical measure". In: *Probab-  
ility Theory and Related Fields* 162.3 (2015), pp. 707–738.
- [FG18] S. Farquhar and Y. Gal. "Towards Robust Evaluations of  
Continual Learning". In: (May 2018). arXiv: [1805.09733 \[stat.ML\]](#).
- [FGG97] N. Friedman, D. Geiger, and M. Goldszmidt. "Bayesian  
network classifiers". In: *ML* 29 (1997), pp. 131–163.
- [FH17] N. Frosst and G. Hinton. *Distilling a Neural Network  
Into a Soft Decision Tree*. 2017. arXiv: [1711.09784 \[cs.LG\]](#).
- [FH20] E. Fong and C. Holmes. "On the marginal likelihood and  
cross-validation". In: *Biometrika* 107.2 (May 2020).
- [FH21] E. Fong and C. C. Holmes. "Conformal Bayesian Com-  
putation". In: *NIPS*. May 2021.
- [FH75] K. Fukunaga and L. Hostetler. "The estimation of the gra-  
dient of a density function, with applications in pattern recog-  
nition". In: *IEEE Trans. Inf. Theory* 21.1 (Jan. 1975), pp. 32–  
40.

- 1 [FH97] B. J. Frey and G. Hinton. "Efficient stochastic source  
2 coding and an application to a Bayesian network source model".  
3 In: *Computer Journal* (1997).
- 4 [FHDV20] J. Futoma, M. C. Hughes, and F. Doshi-Velez. "Pop-  
5 corn: Partially observed prediction constrained reinforcement  
6 learning". In: *AISTATS* (2020).
- 7 [FHK03] P. Felzenszwalb, D. Huttenlocher, and J. Kleinberg.  
8 "Fast Algorithms for Large State Space HMMs with Applications  
9 to Web Usage Analysis". In: *NIPS*. 2003.
- 10 [FHL19] S. Fort, H. Hu, and B. Lakshminarayanan. "Deep En-  
11 sembles: A Loss Landscape Perspective". In: (Dec. 2019). arXiv:  
12 1912.02757 [stat.ML].
- 13 [FHM18] S. Fujimoto, H. van Hoof, and D. Meger. "Addressing  
14 Function Approximation Error in Actor-Critic Methods". In:  
15 *ICLR*. 2018.
- 16 [FHT08] J. Friedman, T. Hastie, and R. Tibshirani. "Sparse in-  
17 verse covariance estimation via the graphical lasso". In: *Biostatistics*  
18 9.3 (2008), pp. 432–441.
- 19 [FI10] A. Fischer and C. Igel. "Empirical analysis of the di-  
20 vergence of Gibbs sampling based learning algorithms for re-  
21 stricted Boltzmann machines". In: *International conference on  
22 artificial neural networks*. Springer. 2010, pp. 208–217.
- 23 [Fie70] S. Fiernberg. "An Iterative Procedure for Estimation in  
24 Contingency Tables". In: *Annals of Mathematical Statistics*  
25 41.3 (1970), pp. 907–917.
- 26 [Fin+16] C. Finn, P. Christiano, P. Abbeel, and S. Levine. "A  
27 connection between generative adversarial networks, inverse  
28 reinforcement learning, and energy-based models". In: *arXiv preprint arXiv:1611.03852* (2016).
- 29 [Fis20] I. Fischer. "The Conditional Entropy Bottleneck". In:  
30 *Entropy* 22.9 (2020).
- 31 [Fis25] R. Fisher. *Statistical Methods for Research Workers*. Bi-  
32 ological monographs and manuals. Oliver and Boyd, 1925.
- 33 [FJ02] M. A. T. Figueiredo and A. K. Jain. "Unsupervised  
34 Learning of Finite Mixture Models". In: *IEEE PAMI* 24.3  
35 (2002), pp. 381–396.
- 36 [FLJ18] R. Frostig, M. J. Johnson, and C. Leary. "Compiling  
37 machine learning programs via high-level tracing". In: *Machine  
38 Learning and Systems (MLSys)* (2018).
- 39 [FK13a] V. Feldman and P. Kothari. "Learning Coverage Func-  
40 tions". In: *CoRR* abs/1304.2079 (2013). arXiv: 1304.2079.
- 41 [FK13b] M. Frei and H. R. Künsch. "Bridging the ensemble  
42 Kalman and particle filters". In: *Biometrika* 100.4 (Dec. 2013),  
43 pp. 781–800.
- 44 [FK14] V. Feldman and P. Kothari. "Learning Coverage Func-  
45 tions and Private Release of Marginals". In: *Proceedings of The  
46 27th Conference on Learning Theory*. Ed. by M. F. Balcan, V.  
47 Feldman, and C. Szepesvári. Vol. 35. Proceedings of Machine  
48 Learning Research. Barcelona, Spain: PMLR, 2014, pp. 679–  
49 702.
- 50 [FK21] A. Fisher and E. H. Kennedy. "Visually Communicat-  
51 ing and Teaching Intuition for Influence Functions". In: *The  
52 American Statistician* 75.2 (2021), pp. 162–172. eprint: <https://doi.org/10.1080/00031305.2020.1717620>.
- 53 [FKH17] S. Falkner, A. Klein, and F. Hutter. "Combining Hy-  
54 perband and Bayesian Optimization". In: *NIPS 2017 Bayesian  
55 Optimization Workshop*. Dec. 2017.
- 56 [FKV13] V. Feldman, P. Kothari, and J. Vondrák. "Representation  
57 Approximation and Learning of Submodular Functions Us-  
58 ing Low-rank Decision Trees". In: *Proceedings of the 26th An-  
59 nual Conference on Learning Theory*. Ed. by S. Shalev-Shwartz  
60 and I. Steinwart. Vol. 30. Proceedings of Machine Learning Re-  
61 search. Princeton, NJ, USA: PMLR, 2013, pp. 711–740.
- 62 [FKV14] V. Feldman, P. Kothari, and J. Vondrák. "Nearly tight  
63 bounds on  $\ell_1$  approximation of self-bounding functions". In:  
64 *CoRR*, abs/1404.4702 1 (2014).
- 65 [FKV17] V. Feldman, P. Kothari, and J. Vondrák. "Tight  
66 Bounds on  $\ell_1$  Approximation and Learning of Self-Bounding  
67 Functions". In: *International Conference on Algorithmic  
68 Learning Theory*. PMLR. 2017, pp. 540–559.
- 69 [FKV20] V. Feldman, P. Kothari, and J. Vondrák. "Tight  
70 bounds on  $\ell_1$  approximation and learning of self-bounding func-  
71 tions". In: *Theoretical Computer Science* 808 (2020), pp. 86–  
72 98.
- 73 [FL07] P. Fearnhead and Z. Liu. "Online Inference for Multiple  
74 Change-point Problems". In: *J. of Royal Stat. Soc. Series B* 69  
75 (2007), pp. 589–605.
- 76 [FL11] P. Fearnhead and Z. Liu. "Efficient Bayesian analysis of  
77 multiple changepoint models with dependence across segments".  
78 In: *Statistics and Computing* 21.2 (2011), pp. 217–229.
- 79 [FL+18] V. François-Lavet, P. Henderson, R. Islam, M. G. Belle-  
80 mare, and J. Pineau. "An Introduction to Deep Reinforcement  
81 Learning". In: *Foundations and Trends in Machine Learning*  
82 11.3 (2018).
- 83 [FLA16] C. Finn, S. Levine, and P. Abbeel. "Guided Cost Learn-  
84 ing: Deep Inverse Optimal Control via Policy Optimization". In:  
85 *ICML*. 2016, pp. 49–58.
- 86 [Fla+16] S. Flaxman, D. Sejdinovic, J. P. Cunningham, and S.  
87 Filippi. "Bayesian Learning of Kernel Embeddings". In: *UAI*.  
88 2016.
- 89 [FLL18] J. Fu, K. Luo, and S. Levine. "Learning Robust Re-  
90 wards with Adversarial Inverse Reinforcement Learning". In:  
91 *ICLR*. 2018.
- 92 [FL19] Y. Feng, L. Li, and Q. Liu. "A Kernel Loss for Solving  
93 the Bellman Equation". In: *NeurIPS*. 2019, pp. 15430–15441.
- 94 [FMM18] M. Figurnov, S. Mohamed, and A. Mnih. "Implicit  
95 Reparameterization Gradients". In: *NIPS*. 2018.
- 96 [FMP19] S. Fujimoto, D. Meger, and D. Precup. "Off-Policy  
97 Deep Reinforcement Learning without Exploration". In: *ICML*.  
98 2019, pp. 2052–2062.
- 99 [FNG00] N. de Freitas, M. Niranjan, and A. Gee. "Hierarchical  
100 Bayesian models for regularisation in sequential learning". In:  
101 *Neural Computation* 12.4 (2000), pp. 955–993.
- 102 [FNW78] M. Fisher, G. Nemhauser, and L. Wolsey. "An analysis  
103 of approximations for maximizing submodular set functions—  
104 II". In: *Polyhedral combinatorics* (1978), pp. 73–87.
- 105 [FO20] F. Farnia and A. Ozdaglar. "Do GANs always have Nash  
106 equilibria?" In: *Proceedings of the 37th International Confer-  
107 ence on Machine Learning*. Ed. by H. D. III and A. Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR,  
108 2020, pp. 3029–3039.
- 109 [Fon+21] D. Fontanelli, F. Cermelli, M. Mancini, and B. Caputo.  
110 "On the Challenges of Open World Recognition under Shifting  
111 Visual Domains". In: *ICRA*. July 2021.
- 112 [For01] G. D. Forney. "Codes on graphs: normal realizations".  
113 In: *IEEE Trans. Inf. Theory* 47.2 (Feb. 2001), pp. 520–548.
- 114 [For+18a] V. Fortuin, G. Dresdner, H. Strathmann, and G.  
115 Rätsch. "Scalable Gaussian Processes on Discrete Domains". In:  
116 (Oct. 2018). arXiv: 1810.10368 [stat.ML].
- 117 [For+18b] M. Fortunato et al. "Noisy Networks for Exploration".  
118 In: *ICLR*. 2018.
- 119 [For+19] N. Ford, J. Gilmer, N. Carlini, and D. Cubuk. "Adver-  
120 sarial Examples Are a Natural Consequence of Test Error in  
121 Noise". In: (Jan. 2019). arXiv: 1901.10513 [cs.LG].
- 122 [For21] V. Fortuin. "Priors in Bayesian Deep Learning: A Re-  
123 view". In: (May 2021). arXiv: 2105.06868 [stat.ML].
- 124 [For+95] J. Forbes, T. Huang, K. Kanazawa, and S. Russell.  
125 "The BATmobile: Towards a Bayesian Automated Taxi". In: *IJ-  
126 CAI*. 1995.
- 127 [Fot+14] N. Foti, J. Xu, D. Laird, and E. Fox. "Stochastic vari-  
128 ational inference for hidden Markov models". In: *NIPS*. 2014,  
129 pp. 3599–3607.
- 130 [FP08] N. Friel and A. N. Pettitt. "Marginal Likelihood Estima-  
131 tion via Power Posteriors". In: *J. of Royal Stat. Soc. Series B*  
132 70.3 (2008), pp. 589–607.
- 133 [FP69] D. C. Fraser and J. E. Potter. "The optimum linear  
134 smoother as a combination of two optimum linear filters". In:  
135 *IEEE Trans. on Automatical Control* (1969), pp. 387–390.
- 136 [FPD09] P. Frazier, W. Powell, and S. Dayanik. "The knowledge-  
137 gradient policy for correlated normal beliefs". In: *INFORMS J.  
138 on Computing* 21.4 (2009), pp. 599–613.
- 139 [Fra08] A. Fraser. *Hidden Markov Models and Dynamical Sys-  
140 tems*. SIAM Press, 2008.
- 141 [Fra+16] M. Fraccaro, S. K. Sønderby, U. Paquet, and O.  
142 Winther. "Sequential Neural Models with Stochastic Layers".  
143 In: *NIPS*. 2016.
- 144 [Fra18] P. I. Frazier. "Bayesian Optimization". In: *Recent Ad-  
145 vances in Optimization and Modeling of Contemporary Prob-  
146 lems*. INFORMS TutORials in Operations Research. INFORMS,  
147 Oct. 2018, pp. 255–278.
- 148 [Fre14] A. A. Freitas. "Comprehensible classification models: a  
149 position paper". In: *ACM SIGKDD explorations newsletter*  
150 15.1 (2014), pp. 1–10.
- 151 [Fre98] B. Frey. *Graphical Models for Machine Learning and  
152 Digital Communication*. MIT Press, 1998.
- 153 [Fre99] R. M. French. "Catastrophic forgetting in connectionist  
154 networks". In: *Trends in Cognitive Science* (1999).
- 155 [Fri03] K. Friston. "Learning and inference in the brain". en. In:  
156 *Neural Netw.* 16.9 (Nov. 2003), pp. 1325–1352.
- 157 [Fri09] K. Friston. "The free-energy principle: a rough guide to  
158 the brain?" en. In: *Trends Cogn. Sci.* 13.7 (July 2009), pp. 293–  
159 301.
- 160 [Fro+21] R. Frostig, M. Johnson, D. Maclaurin, A. Paszke, and  
161 A. Radul. "Decomposing reverse-mode automatic differentia-  
162 tion". In: *LAFI workshop at POPL 2021*. 2021.
- 163 [FS07] S. Frühwirth-Schnatter. *Finite Mixture and Markov  
164 Switching Models*. Springer, 2007.

- 1 [FSF10] S. Frühwirth-Schnatter and R. Frühwirth. "Data Augmentation and MCMC for Binary and Multinomial Logit Models". In: *Statistical Modelling and Regression Structures*. Ed. by T. Kneib and G. Tutz. Springer, 2010, pp. 111–132.
- 2 [FST98] S. Fine, Y. Singer, and N. Tishby. "The Hierarchical Hidden Markov Model: Analysis and Applications". In: *Machine Learning* 32 (1998), p. 41.
- 3 [FT05] M. Faching and C. Tomasi. "Mean shift is a bound optimization", en. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 27.3 (Mar. 2005), pp. 471–474.
- 4 [FT19] A. Finke and A. H. Thiery. "On the relationship between variational inference and adaptive importance sampling". In: (July 2019). arXiv: 1907.10477 [stat.ML].
- 5 [FT74] J. H. Friedman and J. W. Tukey. "A Projection Pursuit Algorithm for Exploratory Data Analysis". In: *IEEE Trans. Comput.* C-23.9 (Sept. 1974), pp. 881–890.
- 6 [Fu15] M. Fu, ed. *Handbook of Simulation Optimization*. 1st ed. Springer-Verlag New York, 2015.
- 7 [Fu+17] Z. Fu, X. Tan, N. Peng, D. Zhao, and R. Yan. "Style transfer in text: Exploration and evaluation". In: *arXiv preprint arXiv:1711.06861* (2017).
- 8 [Fu+19] H. Fu, C. Li, X. Liu, J. Gao, A. Celikyilmaz, and L. Carin. "Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing". In: *NAACL*. Mar. 2019.
- 9 [Fu+20] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. *D4RL: Datasets for Deep Data-Driven Reinforcement Learning*. arXiv:2004.07219. 2020.
- 10 [Fuji05] S. Fujishige. *Submodular functions and optimization*. Vol. 58. Elsevier Science, 2005.
- 11 [Ful+20] I. R. Fulcher, I. Shpitser, S. Marealle, and E. J. Tchetgen Tchetgen. "Robust inference on population indirect causal effects: the generalized front door criterion". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 82.1 (2020), pp. 199–214. eprint: <https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/rssb.12345>.
- 12 [FV15] V. Feldman and J. Vondrák. "Tight Bounds on Low-Degree Spectral Concentration of Submodular and XOS Functions". In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. 2015, pp. 923–942.
- 13 [FV16] V. Feldman and J. Vondrák. "Optimal bounds on approximation of submodular and XOS functions by junta". In: *SIAM Journal on Computing* 45.3 (2016), pp. 1129–1170.
- 14 [FV17] R. C. Fong and A. Vedaldi. "Interpretable explanations of black boxes by meaningful perturbation". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 3429–3437.
- 15 [FW12] N. Friel and J. Wyse. "Estimating the evidence – a review". In: *Stat. Neerl.* 66.3 (Aug. 2012), pp. 288–308.
- 16 [FW21] R. Friedman and Y. Weiss. "Posterior Sampling for Image Restoration using Explicit Patch Priors". In: (Apr. 2021). arXiv: 2104.09895 [cs.CV].
- 17 [FWW21] M. Finzi, M. Welling, and A. G. Wilson. "A Practical Method for Constructing Equivariant Multilayer Perceptrons for Arbitrary Matrix Groups". In: *ICML*. 2021.
- 18 [Gaj+19] A. Gajewski, J. Clune, K. O. Stanley, and J. Lehman. "Evolvability ES: Scalable and Direct Optimization of Evolvability". In: *Proc. of the Conf. on Genetic and Evolutionary Computation*. 2019.
- 19 [Gan+16a] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, and others. "Domain-adversarial training of neural networks". In: *JMLR* (2016).
- 20 [Gan+16b] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. "Domain-adversarial training of neural networks". In: *The journal of machine learning research* 17.1 (2016), pp. 2096–2030.
- 21 [Gao+18] R. Gao, J. Xie, S.-C. Zhu, and Y. N. Wu. "Learning Grid-like Units with Vector Representation of Self-Position and Matrix Representation of Self-Motion". In: *arXiv preprint arXiv:1810.05597* (2018).
- 22 [Gao+20] R. Gao, E. Nijkamp, D. P. Kingma, Z. Xu, A. M. Dai, and Y. N. Wu. "Flow contrastive estimation of energy-based models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 7518–7528.
- 23 [Gär03] T. Gärtner. "A Survey of Kernels for Structured Data". In: *SIGKDD Explor. Newslett.* 5.1 (July 2003), pp. 49–58.
- 24 [GAR16] R. B. Grosse, S. Ancha, and D. M. Roy. "Measuring the reliability of MCMC inference with bidirectional Monte Carlo". In: *NIPS*. June 2016.
- 25 [Gar+18a] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson. "GP-Torch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration". In: *NIPS*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 7576–7586.
- 26 [Gar+18b] J. R. Gardner, G. Pleiss, R. Wu, K. Q. Weinberger, and A. G. Wilson. "Product Kernel Interpolation for Scalable Gaussian Processes". In: *AISTATS*. 2018.
- 27 [Gar+18c] T. Garipov, P. Izmailov, D. Podoprikhin, D. Vetrov, and A. G. Wilson. "Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs". In: *NIPS*.
- 28 [Gar+18d] M. Garnelo, D. Rosenthal, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. M. A. Esfandiari. "Conditional Neural Processes". In: *ICML*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1704–1713.
- 29 [Gar+18e] M. Garnelo, J. Schwarz, D. Rosenthal, F. Viola, D. J. Rezende, S. M. Ali Esfandiari, and Y. W. Teh. "Neural Processes". In: *ICML workshop on Theoretical Foundations and Applications of Deep Generative Models*.
- 30 [Gar+19] S. Garg, V. Perot, N. Limtiaco, A. Taly, E. H. Chi, and A. Beutel. "Counterfactual Fairness in Text Classification through Robustness". In: *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. AIES '19. Association for Computing Machinery, 2019, 219–226.
- 31 [Gar22] R. Garnett. *Bayesian Optimization*. in preparation. Cambridge University Press, 2022.
- 32 [Gas+19] J. Gasthaus, K. Benidis, Y. Wang, S. S. Rangapuram, D. Salinas, V. Flunkert, and T. Januschowski. "Probabilistic Forecasting with Spline Quantile Function RNNs". In: *ICML*. Vol. 89. Proceedings of Machine Learning Research. PMLR, 2019, pp. 1901–1910.
- 33 [GAS18] D. G. A. Smith and J. Gray. "opt-einsum - A Python package for optimizing contraction order for einsum-like expressions". In: *JOSS* 3.26 (June 2018), p. 753.
- 34 [GAZ19] A. Ghorbani, A. Abid, and J. Zou. "Interpretation of neural networks is fragile". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 3681–3688.
- 35 [GB00] Z. Ghahramani and M. Beal. "Variational inference for Bayesian mixtures of factor analysers". In: *NIPS-12*. 2000.
- 36 [GB09] A. Guillory and J. Bilmes. "Label Selection on Graphs". In: *NIPS*. Vancouver, Canada, 2009.
- 37 [GB10] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *AISTATS*. 2010, pp. 249–256.
- 38 [GB11] A. Guillory and J. Bilmes. "Active Semi-Supervised Learning using Submodular Functions". In: *UAI*. Barcelona, Spain: ÅUAI, 2011.
- 39 [GB+18] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparragirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules". en. In: *American Chemical Society Central Science* 4.2 (Feb. 2018), pp. 268–276.
- 40 [GBB11] X. Glorot, A. Bordes, and Y. Bengio. "Deep Sparse Rectifier Neural Networks". In: *AISTATS*. 2011.
- 41 [GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- 42 [GBT95] W. Gilks, N. Best, and K. Tan. "Adaptive rejection Metropolis sampling". In: *Applied Statistics* 44 (1995), pp. 455–472.
- 43 [GC11] M. Girolami and B. Calderhead. "Riemann manifold Langevin and Hamiltonian Monte Carlo methods". In: *J. of Royal Stat. Soc. Series B* 73.2 (Mar. 2011), pp. 213–241.
- 44 [GC90] R. P. Goldman and E. Charniak. "Dynamic Construction of Belief Networks". In: *UAI*. 1990.
- 45 [GCW19] M. Gerber, N. Chopin, and N. Whiteley. "Negative association, ordering and convergence of resampling methods". In: *Ann. Stat.* 47.4 (2019), pp. 2236–2260.
- 46 [GD20] T. Geffner and J. Domke. "A Rule for Gradient Estimator Selection, with an Application to Variational Inference". In: *AISTATS*. Ed. by S. Chiappa and R. Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1803–1812.
- 47 [GDFY16] S. Ghosh, F. M. Delle Fave, and J. Yedidia. "Assumed Density Filtering Methods for Learning Bayesian Neural Networks". In: *AAAI*. 2016.
- 48 [Geb+21] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. D. III, and K. Crawford. "Datasheets for datasets". In: *Communications of the ACM* 64.12 (2021), pp. 86–92.
- 49 [Gei+20a] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann. "Shortcut Learning in Deep Neural Networks". In: *arXiv preprint arXiv:2004.07780* (2020).
- 50 [Gei+20b] R. Geirhos, J. Jacobsen, C. Michaelis, R. S. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann. "Shortcut Learning in Deep Neural Networks". In: *CoRR abs/2004.07780* (2020). arXiv: 2004.07780.
- 51 [Gel+04] A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian data analysis*. 2nd edition. Chapman and Hall, 2004.

- 1
- [Gel06] A. Gelman. "Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper)". en. In: *Bayesian Anal.* 1.3 (Sept. 2006), pp. 515–534.
- 2
- [Gel+08] A. Gelman, A. Jakulin, M. G. Pittau, and Y.-S. Su. "A weakly informative default prior distribution for logistic and other regression models". en. In: *The Annals of Applied Statistics* 2.4 (Dec. 2008), pp. 1360–1383.
- 3
- [Gel+14a] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian Data Analysis, Third Edition*. Third edition. Chapman and Hall/CRC, 2014.
- 4
- [Gel+14b] A. Gelman, A. Vehtari, P. Jylänki, C. Robert, N. Chopin, and J. P. Cunningham. "Expectation propagation as a way of life". In: (Dec. 2014). arXiv: [1412.4869 \[stat.CO\]](#).
- 5
- [Gel+20] A. Gelman, A. Vehtari, D. Simpson, C. C. Margossian, B. Carpenter, Y. Yao, L. Kennedy, J. Gabry, P.-C. Bürkner, and M. Modrák. "Bayesian Workflow". In: (Nov. 2020). arXiv: [2011.01808 \[stat.ME\]](#).
- 6
- [Gel90] M. Gelbrich. "On a formula for the L2 Wasserstein metric between measures on Euclidean and Hilbert spaces". In: *Mathematische Nachrichten* 147.1 (1990), pp. 185–203.
- 7
- [Gen+19] A. Genevay, L. Chizat, F. Bach, M. Cuturi, and G. Peyré. "Sample complexity of sinkhorn divergences". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1574–1583.
- 8
- [Geo+18] T. George, C. Laurent, X. Bouthillier, N. Ballas, and P. Vincent. "Fast Approximate Natural Gradient Descent in a Kronecker Factored Eigenbasis". In: *NIPS*. Curran Associates, Inc., 2018, pp. 9550–9560.
- 9
- [Geo88] H.-O. Georgii. *Gibbs Measures and Phase Transitions*. en. Walter De Gruyter Inc, Oct. 1988.
- 10
- [Ger+15] M. Germain, K. Gregor, I. Murray, and H. Larochelle. "MADE: Masked Autoencoder for Distribution Estimation". In: *ICML*. Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 881–889.
- 11
- [Gér19] A. Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques for Building Intelligent Systems (2nd edition)*. en. O'Reilly Media, Incorporated, 2019.
- 12
- [Ger19] S. J. Gershman. "What does the free energy principle tell us about the brain?". In: *Neurons, Behavior, Data Analysis, and Theory* (Jan. 2019).
- 13
- [Gey92] C. Geyer. "Practical Markov chain Monte Carlo". In: *Statistical Science* 7 (1992), pp. 473–483.
- 14
- [GF00] E. George and D. Foster. "Calibration and empirical Bayes variable selection". In: *Biometrika* 87.4 (2000), pp. 731–747.
- 15
- [GF09] I. E. Givoni and B. J. Frey. "A Binary Variable Model for Affinity Propagation". In: *Neural Computation* 21.6 (2009), pp. 1589–1600.
- 16
- [GF17] B. Goodman and S. Flaxman. "European Union regulations on algorithmic decision-making and a 'right to explanation'". In: *AI magazine* 38.3 (2017), pp. 50–57.
- 17
- [GF沃20] T. Galy-Fajou, F. Wenzel, and M. Opper. "Automated Augmented Conjugate Inference for Non-conjugate Gaussian Process Models". In: *AISTATS*, 2020.
- 18
- [GG11] T. L. Griffiths and Z. Ghahramani. "The Indian Buffet Process: An Introduction and Review". In: *JMLR* 12.4 (2011).
- 19
- [GG14] S. J. Gershman and N. D. Goodman. "Amortized Inference in Probabilistic Reasoning". In: *36th Annual Conference of the Cognitive Science Society*, 2014.
- 20
- [GG16] Y. Gal and Z. Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *ICML*, 2016.
- 21
- [GG84] S. Geman and D. Geman. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". In: *IEEE PAMI* 6.6 (1984).
- 22
- [GGA15] R. B. Grosse, Z. Ghahramani, and R. P. Adams. "Sandwiching the marginal likelihood using bidirectional Monte Carlo". In: (Nov. 2015). arXiv: [1511.02543 \[stat.ML\]](#).
- 23
- [GGG15] M. Gygli, H. Grabner, and L. Gool. "Video summarization by learning submodular mixtures of objectives". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 3090–3098.
- 24
- [GGO19] F. Gurkan, B. Gunsel, and C. Ozer. "Robust object tracking via integration of particle filtering with deep detection". In: *Digit. Signal Process.* 87 (Apr. 2019), pp. 112–124.
- 25
- [GGT15] S. Gu, Z. Ghahramani, and R. E. Turner. "Neural Adaptive Sequential Monte Carlo". In: *NIPS*, 2015.
- 26
- [GH07] A. Gelman and J. Hill. *Data analysis using regression and multilevel/hierarchical models*. Cambridge, 2007.
- 27
- [GH10] M. Gutmann and A. Hyvärinen. "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 2010, pp. 297–304.
- 28
- [GH12a] F. Gustafsson and G. Hendeby. "Some Relations Between Extended and Unscented Kalman Filters". In: *IEEE Trans. Signal Process.* 60.2 (Feb. 2012), pp. 545–555.
- 29
- [GH12b] M. Gutmann and J.-i. Hayama. "Bregman divergence as general framework to estimate unnormalized statistical models". In: [arXiv:1202.3727](#) (2012).
- 30
- [GH96a] Z. Ghahramani and G. Hinton. *Parameter estimation for linear dynamical systems*. Tech. rep. CRG-TR-96-2. Dept. Comp. Sci., Univ. Toronto, 1996.
- 31
- [GH96b] Z. Ghahramani and G. Hinton. *The EM Algorithm for Mixtures of Factor Analyzers*. Tech. rep. Dept. of Comp. Sci., Uni. Toronto, 1996.
- 32
- [Gha+15] M. Ghavamzadeh, S. Mannor, J. Pineau, and A. Tamar. "Bayesian Reinforcement Learning: A Survey". In: *Foundations and Trends in ML* (2015).
- 33
- [Gha+21] A. Ghaddeharion, B. Kim, C.-L. Li, B. Jou, B. Eoff, and R. W. Picard. *DISSECT: Disentangled Simultaneous Explanations via Concept Traversals*. 2021. arXiv: [2105.15164 \[cs.LG\]](#).
- 34
- [GHC20] C. Geng, S.-J. Huang, and S. Chen. "Recent Advances in Open Set Recognition: A Survey". In: *IEEE PAMI* (2020).
- 35
- [GHC21] S. Gould, R. Hartley, and D. J. Campbell. "Deep Declarative Networks". en. In: *IEEE PAMI PP* (Feb. 2021).
- 36
- [GHK17] Y. Gal, J. Hron, and A. Kendall. "Concrete Dropout". In: (May 2017). arXiv: [1705.07832 \[stat.ML\]](#).
- 37
- [Gho+19] A. Ghorbani, J. Wexler, J. Zou, and B. Kim. *Towards Automatic Concept-based Explanations*. 2019. arXiv: [1902.03129 \[stat.ML\]](#).
- 38
- [GHV20] A. Gelman, J. Hill, and A. Vehtari. *Regression and Other Stories*. en. 1st ed. Cambridge University Press, July 2020.
- 39
- [Gil+17] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. "Neural message passing for quantum chemistry". In: *ICML*, 2017, pp. 1263–1272.
- 40
- [Gil+18a] J. Gilmer, R. P. Adams, I. Goodfellow, D. Andersen, and G. E. Dahl. "Motivating the rules of the game for adversarial example research". In: [arXiv preprint arXiv:1807.06732](#) (2018).
- 41
- [Gil+18b] J. Gilmer, L. Metz, F. Faghri, S. S. Schoenholz, M. Raghu, M. Wattenberg, and I. Goodfellow. "Adversarial spheres". In: [arXiv preprint arXiv:1801.02774](#) (2018).
- 42
- [Gil88] J. R. Gilbert. "Some nested dissection order is nearly optimal". In: *Inf. Process. Lett.* 26.6 (Jan. 1988), pp. 325–328.
- 43
- [Gir+15] R. Girshick, F. Iandola, T. Darrell, and J. Malik. "Deformable Part Models are Convolutional Neural Networks". In: *CVPR*, 2015.
- 44
- [Gir+21] L. Girin, S. Leglaive, X. Bie, J. Diard, T. Hueber, and X. Alameda-Pineda. "Dynamical Variational Autoencoders: A Comprehensive Review". In: *Foundations and Trends® in Machine Learning* 15.1-2 (2021), pp. 1–175.
- 45
- [Git89] J. Gittins. *Multi-armed Bandit Allocation Indices*. Wiley, 1989.
- 46
- [GJ97] Z. Ghahramani and M. Jordan. "Factorial Hidden Markov Models". In: *Machine Learning* 29 (1997), pp. 245–273.
- 47
- [GK19] L. Graesser and W. L. Keng. *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*. en. 1 edition. Addison-Wesley Professional, Dec. 2019.
- 48
- [GKS05] C. Guestin, A. Krause, and A. P. Singh. "Near-optimal sensor placements in gaussian processes". In: *Proceedings of the 22nd international conference on Machine learning*. 2005, pp. 265–272.
- 49
- [GL10] K. Gregor and Y. LeCun. "Learning fast approximations of sparse coding". In: *ICML*, 2010, pp. 399–406.
- 50
- [Gla03] P. Glasserman. *Monte Carlo Methods in Financial Engineering*. 1st ed. Stochastic Modelling and Applied Probability. Springer-Verlag New York, 2003.
- 51
- [Gle02] S. Glennan. "Rethinking mechanistic explanation". In: *Philosophy of science* 69.S3 (2002), S342–S353.
- 52
- [GLM15] J. Ghosh, Y. Li, and R. Mitra. "On the Use of Cauchy Prior Distributions for Bayesian Logistic Regression". In: (July 2015). arXiv: [1507.07170 \[stat.ME\]](#).
- 53
- [GLP21] I. Gulrajani and D. Lopez-Paz. "In Search of Lost Domain Generalization". In: *ICLR*, 2021.
- 54
- [GLS81] M. Grötschel, L. Lovász, and A. Schrijver. "The ellipsoid method and its consequences in combinatorial optimization". In: *Combinatorica* 1.2 (1981), pp. 169–197.
- 55
- [GM12] G. Gordon and J. McNulty. *Matroids: a geometric introduction*. Cambridge University Press, 2012.

- 1
- 2 [GM15] J. Gorham and L. Mackey. "Measuring sample quality  
with Stein's method". In: *Advances in Neural Information Processing Systems*. 2015, pp. 226–234.
- 3 [GM16] R. Grossé and J. Martens. "A Kronecker-factored ap-  
proximate Fisher matrix for convolution layers". In: *ICML*. 2016.
- 4 [GM98] A. Gelman and X.-L. Meng. "Simulating normalizing  
constants: from importance sampling to bridge sampling to  
path sampling". In: *Statistical Science* 13 (1998), pp. 163–185.
- 5 [GMAR16] Y. Gal, R. T. Mc Allister, and C. E. Rasmussen. "Im-  
proving PILCO with Bayesian Neural Network Dynamics Models". In: *ICML workshop on Data-efficient machine learning*. 2016.
- 6 [GMH20] M. Gorinova, D. Moore, and M. Hoffman. "Automatic  
Reparameterisation of Probabilistic Programs". In: *ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 3648–3657.
- 7 [GNM19] D. Greenberg, M. Nonnenmacher, and J. Macke. "Au-  
tomatic Posterior Transformation for Likelihood-Free Infer-  
ence". In: *ICML*. 2019.
- 8 [GO17] P. Grünwald and T. van Ommen. "Inconsistency of  
Bayesian Inference for Misspecified Linear Models, and a Pro-  
posal for Repairing It". en. In: *Bayesian Analysis* 12.4 (Dec.  
2017), pp. 1069–1103.
- 9 [Goe+09] M. X. Goemans, N. J. Harvey, S. Iwata, and V. Mir-  
rokni. "Approximating submodular functions everywhere". In:  
*Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 2009, pp. 535–544.
- 10 [Gol+17] N. Gold, M. G. Frasch, C. Herry, B. S. Richardson,  
and X. Wang. "A Doubly Stochastic Change Point Detection  
Algorithm for Noisy Biological Signals". en. In: *Front. Physiol.*  
(Oct. 2017), p. 106088.
- 11 [Gom+17] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse.  
"The Reversible Residual Network: Backpropagation Without  
Storing Activations". In: *NIPS*. 2017.
- 12 [Gon+11] J. Gonzalez, Y. Low, A. Gretton, and C. Guestrin.  
"Parallel gibbs sampling: From colored fields to thin junction  
trees". In: *ASTATS*. 2011, pp. 324–332.
- 13 [Gon+14] B. Gong, W.-L. Chao, K. Grauman, and F. Sha.  
"Diverse sequential subset selection for supervised video summa-  
rization". In: *Advances in neural information processing sys-  
tems* 27 (2014), pp. 2069–2077.
- 14 [Gon+20] P. J. Goncalves et al. "Training deep neural density  
estimators to identify mechanistic models of neural dynamics".  
In: *Elife* 9 (2020).
- 15 [Goo+14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu,  
D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Gen-  
erative Adversarial Networks". In: *NIPS*. 2014.
- 16 [Goo16] I. Goodfellow. "NIPS 2016 Tutorial: Generative Adver-  
sarial Networks". In: *NIPS Tutorial*. 2016.
- 17 [Goo85] I. Good. "Weight of evidence: A brief survey". In:  
*Bayesian statistics* 2 (1985), pp. 249–270.
- 18 [Gor+14] A. D. Gordon, T. A. Henzinger, A. V. Nori, and  
S. K. Rajamani. "Probabilistic programming". In: *Intl. Conf.  
on Software Engineering*. 2014.
- 19 [Gor+19] J. Gordon, J. Bronskill, M. Bauer, S. Nowozin, and  
R. E. Turner. "Meta-Learning Probabilistic Inference For Pre-  
diction". In: *ICLR*. 2019.
- 20 [Gor93] N. Gordon. "Novel Approach to Nonlinear/Non-  
Gaussian Bayesian State Estimation". In: *IEE Proceedings (F)*  
140.2 (1993), pp. 107–113.
- 21 [Gor95] G. J. Gordon. "Stable Function Approximation in Dy-  
namic Programming". In: *ICML*. 1995, pp. 261–268.
- 22 [Gou+96] C. Gourieroux, M. Gourieroux, A. Monfort, and D. A.  
Monfort. *Simulation-based econometric methods*. Oxford uni-  
versity press, 1996.
- 23 [Goy+19] Y. Goyal, Z. Wu, J. Ernst, D. Batra, D. Parikh, and  
S. Lee. *Counterfactual Visual Explanations*. 2019. arXiv: 1904.  
07451 [cs.LG].
- 24 [GPS89] D. Greig, B. Porteous, and A. Seheult. "Exact maxi-  
mum posterior estimation for binary images". In: *J. of Royal  
Stat. Soc. Series B* 51.2 (1989), pp. 271–279.
- 25 [GR06a] M. Girolami and S. Rogers. "Variational Bayesian Multi-  
nomial Probit Regression with Gaussian Process Priors". In:  
*Neural Comput.* 18.8 (Aug. 2006), pp. 1790–1817.
- 26 [GR06b] M. Girolami and S. Rogers. "Variational Bayesian multi-  
nomial probit regression with Gaussian process priors". In:  
*Neural Computation* 18.8 (2006), pp. 1790–1817.
- 27 [GR07a] T. Gneiting and A. E. Raftery. "Strictly Proper Scoring  
Rules, Prediction, and Estimation". In: *JASA* 102.477  
(2007), pp. 359–378.
- 28 [GR07b] T. Gneiting and A. E. Raftery. "Strictly proper scoring  
rules, prediction, and estimation". In: *Journal of the American  
statistical Association* 102.477 (2007), pp. 359–378.
- 29 [Gra+10] T. Graepel, J. Quinonero-Candela, T. Borchert, and  
R. Herbrich. "Web-Scale Bayesian Click-Through Rate Pre-  
diction for Sponsored Search Advertising in Microsoft's Bing  
Search Engine". In: *ICML*. 2010.
- 30 [Gra11] A. Graves. "Practical Variational Inference for Neural  
Networks". In: *NIPS*. 2011.
- 31 [Gra+18] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I.  
Sutskever, and D. Duvenaud. "FFJORD: Free-form Continuous  
Dynamics for Scalable Reversible Generative Models". In: (Oct.  
2018). arXiv: 1810.01367 [cs.LG].
- 32 [Gra+20a] W. Grathwohl, J. Kelly, M. Hashemi, M. Norouzi, K.  
Swersky, and D. Duvenaud. "No MCML for me! Amortized sam-  
pling for fast and stable training of energy-based models". In:  
*arXiv preprint arXiv:2010.04230* (2020).
- 33 [Gra+20b] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duve-  
naud, M. Norouzi, and K. Swersky. "Your classifier is secretly  
an energy-based model and you should treat it like one". In:  
*ICLR*. 2020.
- 34 [Gra+20c] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duve-  
naud, and R. Zemel. "Cutting out the Middle-Man: Training  
and Evaluating Energy-Based Models without Sampling". In:  
*arXiv preprint arXiv:2002.05616* (2020).
- 35 [Gre03] P. Green. "Tutorial on trans-dimensional MCMC". In:  
*Highly Structured Stochastic Systems*. Ed. by P. Green, N.  
Hjort, and S. Richardson. OUP, 2003.
- 36 [Gre+12] A. Gretton, K. M. Borgwardt, M. J. Rasch, B.  
Schölkopf, and A. Smola. "A Kernel Two-Sample Test". In:  
*JMLR* 13.Mar (2012), pp. 723–773.
- 37 [Gre+14] K. Gregor, I. Danihelka, A. Mnih, C. Blundell, and D.  
Wierstra. "Deep AutoRegressive Networks". In: *ICML*. 2014.
- 38 [Gre20] F. Greenlee. *Transformer VAE*. 2020.
- 39 [Gre+22] P. L. Green, R. E. Moore, R. J. Jackson, J. Li, and S.  
Maskell. "Increasing the efficiency of Sequential Monte Carlo  
samplers through the use of approximately optimal L-kernels".  
In: *Mech. Syst. Signal Process.* 162 (2022).
- 40 [Gre98] P. Green. "Reversible Jump Markov Chain Monte  
Carlo computation and Bayesian model determination". In:  
*Biometrika* 82 (1998), pp. 711–732.
- 41 [Gri20] T. L. Griffiths. "Understanding Human Intelligence  
through Human Limitations". en. In: *Trends Cogn. Sci.* 24.11  
(Nov. 2020), pp. 873–883.
- 42 [GRS96] W. Gilks, S. Richardson, and D. Spiegelhalter. *Markov  
Chain Monte Carlo in Practice*. Chapman and Hall, 1996.
- 43 [GS08] Y. Guo and D. Schuurmans. "Efficient global optimization  
for exponential family PCA and low-rank matrix factorization".  
In: *2008 6th Annual Allerton Conference on Communication,  
Control, and Computing*. Sept. 2008, pp. 1100–1107.
- 44 [GS15] R. B. Grosse and R. Salakhutdinov. "Scaling Up Natu-  
ral Gradients by Sparsely Factorizing the Inverse Fisher Matrix".  
In: *ICML*. 2015.
- 45 [GS90] A. Gelfand and A. Smith. "Sampling-based approaches  
to calculating marginal densities". In: *JASA* 85 (1990), pp. 385–  
409.
- 46 [GS92] G. Grimmett and D. Stirzaker. *Probability and Random  
Processes*. Oxford, 1992.
- 47 [GSA14] M. A. Gelbart, J. Snoek, and R. P. Adams. "Bayesian  
Optimization with Unknown Constraints". In: *UAI*. 2014.
- 48 [GSD13] A. Guez, D. Silver, and P. Dayan. "Scalable and Effi-  
cient Bayes-Adaptive Reinforcement Learning Based on Monte-  
Carlo Tree Search". In: *JAIR* 48 (2013), pp. 841–883.
- 49 [GSJ19] A. Gretton, D. Sutherland, and W. Jitkrittum. *NIPS  
tutorial on interpretable comparison of distributions and mod-  
els*. 2019.
- 50 [GSS15] I. J. Goodfellow, J. Shlens, and C. Szegedy. "Explain-  
ing and Harnessing Adversarial Examples". In: *ICLR*. 2015.
- 51 [GSZ21] P. Grünewald, T. Steinke, and L. Zakythinos. "PAC-  
Bayes, MAC-Bayes and Conditional Mutual Information: Fast  
rate bounds that handle general VC classes". In: *COLT*. June  
2021.
- 52 [Gue19] B. Guedj. "A primer on PAC-Bayesian learning". In:  
*arXiv preprint arXiv:1901.05953* (2019).
- 53 [Gul+17] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin,  
and A. C. Courville. "Improved training of wasserstein gans".  
In: *NIPS*. 2017, pp. 5767–5777.
- 54 [Gul+20] C. Gulcehre et al. *RL Unplugged: Benchmarks for Of-  
fline Reinforcement Learning*. arXiv:2006.13888. 2020.
- 55 [Gum54] E. J. Gumbel. *Statistical theory of extreme values  
and some practical applications: A series of lectures (United  
States. National Bureau of Standards. Applied mathematics  
series)*. en. 1st edition. U.S. Govt. Print. Office, 1954.
- 56 [Guo09] Y. Guo. "Supervised exponential family principal com-  
ponent analysis via convex optimization". In: *NIPS*. 2009.
- 57 [Guo+17] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. "On  
Calibration of Modern Neural Networks". In: *ICML*. 2017.

- 1
- [Gur+18] S. Gururangan, S. Swayamdipta, O. Levy, R. Schwartz, S. R. Bowman, and N. A. Smith. “Annotation Artifacts in Natural Language Inference Data”. In: *CoRR* abs/1803.02324 (2018). arXiv: 1803.02324.
- 2
- [Gus01] M. Gustafsson. “A probabilistic derivation of the partial least-squares algorithm”. In: *Journal of Chemical Information and Modeling* 41 (2001), pp. 288–294.
- 3
- [Gut+14] M. U. Gutmann, R. Dutta, S. Kaski, and J. Corander. “Statistical inference of intractable generative models via classification”. In: *arXiv preprint arXiv:1407.4981* (2014).
- 4
- [GW08] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Second. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2008.
- 5
- [GW92] W. Gilks and P. Wild. “Adaptive rejection sampling for Gibbs sampling”. In: *Applied Statistics* 41 (1992), pp. 337–348.
- 6
- [GXG18] H. Ge, K. Xu, and Z. Ghahramani. “Turing: a language for flexible probabilistic inference”. In: *AISTATS*. 2018, pp. 1682–1690.
- 7
- [GZG19] S. K. S. Ghasemipour, R. S. Zemel, and S. Gu. “A Divergence Minimization Perspective on Imitation Learning Methods”. In: *CORL*. 2019, pp. 1259–1277.
- 8
- [HA21] R. J. Hyndman and G. Athanassopoulos. *Forecasting: Principles and Practice*. en 3rd ed. Otexts, May 2021.
- 9
- [Haa+17] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. “Reinforcement learning with deep energy-based policies”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. 2017, pp. 1352–1361.
- 10
- [Haa+18a] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. In: *International Conference on Machine Learning*. 2018, pp. 1861–1870.
- 11
- [Haa+18b] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. In: *ICML*. 2018.
- 12
- [Haa+18c] T. Haarnoja et al. “Soft Actor-Critic Algorithms and Applications”. In: (Dec. 2018). arXiv: 1812.05905 [cs.LG].
- 13
- [Had+20] R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu. “Embracing Change: Continual Learning in Deep Neural Networks”. en. In: *Trends Cogn. Sci.* 24.12 (Dec. 2020), pp. 1028–1040.
- 14
- [Haf18] D. Hafner. *Building Variational Auto-Encoders in TensorFlow*. Blog post. 2018.
- 15
- [Haf+19] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. “Learning Latent Dynamics for Planning from Pixels”. In: *ICML*. 2019.
- 16
- [Haf+20] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. “Dream to Control: Learning Behaviors by Latent Imagination”. In: *ICLR*. 2020.
- 17
- [Hag+17] M. Hagen, M. Potthast, M. Göhsen, A. Rathgeber, and B. Stein. “A Large-Scale Query Spelling Correction Corpus”. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’17. Shinjuku, Tokyo, Japan: ACM, 2017, pp. 1261–1264.
- 18
- [Háj08] A. Hájek. “Dutch Book Arguments”. In: *The Oxford Handbook of Rational and Social Choice*. Ed. by P. Anand, P. Pattanaik, and C. Puppe. Oxford University Press, 2008.
- 19
- [Haj88] B. Hajek. “Cooling Schedules for Optimal Annealing”. In: *Math. Oper. Res.* 13.2 (1988), pp. 311–329.
- 20
- [Hal76] R. Halin. “S-functions for graphs”. en. In: *J. Geom.* 8.1–2 (Mar. 1976), pp. 171–186.
- 21
- [Ham90] J. Hamilton. “Analysis of time series subject to changes in regime”. In: *J. Econometrics* 45 (1990), pp. 39–70.
- 22
- [Han+20] K. Han et al. “A Survey on Vision Transformer”. In: (Dec. 2020). arXiv: 2012.12556 [cs.CV].
- 23
- [Ham80] T. S. Han. “Multiple mutual informations and multiple interactions in frequency data”. In: *Information and Control* 46.1 (1980), pp. 26–45.
- 24
- [Har+17] J. Hartford, G. Lewis, K. Leyton-Brown, and M. Taddy. “Deep IV: A flexible approach for counterfactual prediction”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1414–1423.
- 25
- [Har18] K. Hartnett. “To Build Truly Intelligent Machines, Teach Them Cause and Effect”. In: *Quanta Magazine* (2018).
- 26
- [Har90] A. C. Harvey. *Forecasting, Structural Time Series Models, and the Kalman Filter*. Cambridge University Press, 1990.
- 27
- [Has10] H. van Hasselt. “Double Q-learning”. In: *NIPS*. Ed. by J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta. Curran Associates, Inc., 2010, pp. 2613–2621.
- 28
- [Has70] W. Hastings. “Monte Carlo Sampling Methods Using Markov Chains and Their Applications”. In: *Biometrika* 57.1 (1970), pp. 97–109.
- 29
- 30
- 31
- 32
- 33
- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44
- 45
- 46
- 47
- [Has97] J. Haslett. “On the sample variogram and the sample autocovariance for non-stationary time series”. In: *J Royal Statistical Soc D* 46.4 (Dec. 1997), pp. 475–484.
- [Hau+10] J. R. Hauser, O. Toubia, T. Evgeniou, R. Befurt, and D. Dzyabura. “Disjunctions of conjunctions, cognitive simplicity, and consideration sets”. In: *Journal of Marketing Research* 47.3 (2010), pp. 485–496.
- [Haw71] A. G. Hawkes. “Point spectra of some mutually exciting point processes”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 33.3 (1971), pp. 438–443.
- [HBW19] E. Hoogeboom, R. van den Berg, and M. Welling. “Emerging Convolutions for Generative Normalizing Flows”. In: *ICML*. 2019.
- [HC93] G. Hinton and D. V. Camp. “Keeping Neural Networks Simple by Minimizing the Description Length of the Weights”. In: *in Proc. of the 6th Ann. ACM Conf. on Computational Learning Theory*. ACM Press, 1993, pp. 5–13.
- [HCG20] S. Huang, Y. Cao, and R. Grosse. “Evaluating Lossy Compression Rates of Deep Generative Models”. In: *ICML*. 2020.
- [HD19] D. Hendrycks and T. Dietterich. “Benchmarking Neural Network Robustness to Common Corruptions and Perturbations”. In: *ICLR*. 2019.
- [HDL17] D. Ha, A. M. Dai, and Q. V. Le. “HyperNetworks”. In: *ICLR*. 2017.
- [He+16a] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition”. In: *CVPR*. 2016.
- [HE16a] J. Ho and S. Ermon. “Generative adversarial imitation learning”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 2016, pp. 4572–4580.
- [He+16b] K. He, X. Zhang, S. Ren, and J. Sun. “Identity Mappings in Deep Residual Networks”. In: *ECCV*. 2016.
- [HE16b] J. Ho and S. Ermon. “Generative Adversarial Imitation Learning”. In: *NIPS*. 2016, pp. 4565–4573.
- [HE18] D. Ha and D. Eck. “A Neural Representation of Sketch Drawings”. In: *ICLR*. 2018.
- [He+19] J. He, D. Spokony, G. Neubig, and T. Berg-Kirkpatrick. “Lagging Inference Networks and Posterior Collapses in Variational Autoencoders”. In: *ICLR*. Jan. 2019.
- [Hei13] M. Heinz. “Tree-Decomposition Graph Minor Theory and Algorithmic Implications”. PhD thesis. U. Wien, 2013.
- [Hel17] J. Helse. “KFAS: Exponential Family State Space Models in R”. In: *J. Stat. Softw.* (2017).
- [Hen+15] J. Hensman, A. Matthews, M. Filippone, and Z. Ghahramani. “MCMC for Variationally Sparse Gaussian Processes”. In: *NIPS*. 2015, pp. 1648–1656.
- [Hen+16] L. A. Hendricks, Z. Akata, M. Rohrbach, J. Donahue, B. Schiele, and T. Darrell. “Generating visual explanations”. In: *European conference on computer vision*. Springer, 2016, pp. 3–19.
- [Hen+18] G. E. Henter, J. Lorenzo-Trueba, X. Wang, and J. Yamagishi. “Deep Encoder-Decoder Models for Unsupervised Learning of Controllable Speech Synthesis”. In: (July 2018). arXiv: 1807.11470 [eess.AS].
- [Hen+19a] O. J. Henaff, A. Razavi, C. Doersch, S. M. Ali Eslami, and A. van den Oord. “Data-Efficient Image Recognition with Contrastive Predictive Coding”. In: *arXiv [cs.CV]* (May 2019).
- [Hen+19b] D. Hendrycks, S. Basart, M. Mazeika, A. Zou, J. Kwon, M. Mostajabi, J. Steinhardt, and D. Song. “Scaling Out-of-Distribution Detection for Real-World Settings”. In: (Nov. 2019). arXiv: 1911.11132 [cs.CV].
- [Hen+20] D. Hendrycks\*, N. Mu\*, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan. “AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty”. In: *ICLR*. 2020.
- [Hen+21] C. Henning, M. R. Cervera, F. D’Angelo, J. von Oswald, R. Traber, B. Ehret, S. Kobayashi, B. F. Grewe, and J. Sacramento. “Posterior Meta-Replay for Continual Learning”. In: *NIPS*. Mar. 2021.
- [Hes00] T. Heskes. “On ‘Natural’ Learning and Pruning in Multilayered Perceptrons”. In: *Neural Comput.* 12.4 (Apr. 2000), pp. 881–901.
- [Hes+18] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. “Rainbow: Combining Improvements in Deep Reinforcement Learning”. In: *AAAI*. 2018.
- [Heu+17a] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. “GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium”. In: *NIPS*. 2017.
- [Heu+17b] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. “Gans trained by a two time-scale update rule converge to a local Nash equilibrium”. In: *Advances in neural information processing systems*. 2017, pp. 6626–6637.

- 1
- 2 [HF09] R. Hess and A. Fern. "Discriminatively Trained Particle Filters for Complex Multi-Object Tracking". In: *CVPR*. 2009.
- 3 [HFL13] J. Hensman, N. Fusi, and N. D. Lawrence. "Gaussian Processes for Big Data". In: *UAI*. 2013.
- 4 [HFM17] D. W. Hogg and D. Foreman-Mackey. "Data analysis recipes: Using Markov Chain Monte Carlo". In: (Oct. 2017). arXiv: 1710.06068 [*astro-ph.IM*].
- 5
- 6 [HG12] D. I. Hastie and P. J. Green. "Model Choice using Reversible Jump Markov Chain Monte Carlo". In: *Statistica Neerlandica* 66 (2012), pp. 309–338.
- 7
- 8 [HG14] M. D. Hoffman and A. Gelman. "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo". In: *JMLR* 15 (2014), pp. 1593–1623.
- 9 [HG16] D. Hendrycks and K. Gimpel. "Gaussian Error Linear Units (GELOS)". In: *arXiv [cs.LG]* (June 2016).
- 10
- 11 [HGMG18] J. Hron, A. G. de G. Matthews, and Z. Ghahramani. "Variational Bayesian dropout: pitfalls and fixes". In: *ICML*. 2018.
- 12 [HGS16] H. van Hasselt, A. Guez, and D. Silver. "Deep Reinforcement Learning with Double Q-Learning". In: *AAAI*. 2016. AAAI'16. Phoenix, Arizona: AAAI Press, 2016, pp. 2094–2100.
- 13
- 14 [HH06] C. Holmes and L. Held. "Bayesian auxiliary variable models for binary and multinomial regression". In: *Bayesian Analysis* 1.1 (2006), pp. 145–168.
- 15 [HHC12] J. Hu, P. Hu, and H. S. Chang. "A Stochastic Approximation Framework for a Class of Randomized Optimization Algorithms". In: *IEEE Trans. Automatic Control* 57.1 (2012).
- 16
- 17 [HH09] A. Hyvärinen, J. Hurri, and P. Hoyer. *Natural Image Statistics: a probabilistic approach to early computational vision*. Springer, 2009.
- 18
- 19 [HHK19] M. Häusmann, F. A. Hamprecht, and M. Kandemir. "Sampling-Free Variational Inference of Bayesian Neural Networks by Variance Backpropagation". In: *UAI*. 2019.
- 20 [HHLB11] F. Hutter, H. H. Hoos, and K. Leyton-Brown. "Sequential Model-Based Optimization for General Algorithm Configuration". In: *Intl. Conf. on Learning and Intelligent Optimization (LION)*. 2011, pp. 507–523.
- 21
- 22 [HHLMF18] M. Havasi, J. M. Hernández-Lobato, and J. J. Murillo-Fuentes. "Inference in Deep Gaussian Processes using Stochastic Gradient Hamiltonian Monte Carlo". In: (June 2018). arXiv: 1806.05490 [*stat.ML*].
- 23
- 24 [Hig+17] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *ICLR*. 2017.
- 25
- 26 [Hin02] G. E. Hinton. "Training products of experts by minimizing contrastive divergence". en. In: *Neural Computation* 14.8 (Aug. 2002), pp. 1771–1800.
- 27
- 28 [Hin10] G. Hinton. *A Practical Guide to Training Restricted Boltzmann Machines*. Tech. rep. U. Toronto, 2010.
- 29
- 30 [Hin14] G. Hinton. *Lecture 6e on neural networks (RMSprop: Divide the gradient by a running average of its recent magnitude)*. 2014.
- 31
- 32 [Hin+95] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. "The "wake-sleep" algorithm for unsupervised neural networks". en. In: *Science* 268.5214 (May 1995), pp. 1158–1161.
- 33
- 34 [HIY19] K. Hayashi, M. Imaizumi, and Y. Yoshida. "On Random Subsampling of Gaussian Process Regression: A Graphon-Based Analysis". In: (Jan. 2019). arXiv: 1901.09541 [*stat.ML*].
- 35
- 36 [HJ20] J. Huang and N. Jiang. "From Importance Sampling to Doubly Robust Policy Gradient". In: *ICML*. 2020.
- 37
- 38 [HJA20] J. Ho, A. Jain, and P. Abbeel. "Denoising Diffusion Probabilistic Models". In: *NIPS*. 2020.
- 39 [HJT18] M. D. Hoffman, M. J. Johnson, and D. Tran. "Autoconj: Recognizing and Exploiting Conjugacy Without a Domain-Specific Language". In: *NIPS*. 2018.
- 40 [HKZ12] D. Hsu, S. Kakade, and T. Zhang. "A spectral algorithm for learning hidden Markov models". In: *J. of Computer and System Sciences* 78.5 (2012), pp. 1460–1480.
- 41
- 42 [HL04] D. R. Hunter and K. Lange. "A Tutorial on MM Algorithms". In: *The American Statistician* 58 (2004), pp. 30–37.
- 43
- 44 [HL+16a] J. Hernandez-Lobato, Y. Li, M. Rowland, T. Bui, D. Hernandez-Lobato, and R. Turner. "Black-Box Alpha Divergence Minimization". en. In: *ICML*. June 2016, pp. 1511–1520.
- 45
- 46 [HL+16b] M. Hernandez-Lobato, M. A. Gelbart, R. P. Adams, M. W. Hoffman, and Z. Ghahramani. "A General Framework for Constrained Bayesian Optimization using Information-based Search". In: *JMLR* (2016).
- 47
- 48 [HL20] X. Hu and J. Lei. "A Distribution-Free Test of Covariate Shift Using Conformal Prediction". In: (Oct. 2020). arXiv: 2010.07147 [*stat.ME*].
- 49
- 50
- 51
- 52
- 53
- 54
- 55
- 56
- 57
- 58
- 59
- 60
- 61
- 62
- 63
- 64
- 65
- 66
- 67
- 68
- 69
- 70
- 71
- 72
- 73
- 74
- 75
- 76
- 77
- 78
- 79
- 80
- 81
- 82
- 83
- 84
- 85
- 86
- 87
- 88
- 89
- 90
- 91
- 92
- 93
- 94
- 95
- 96
- 97
- 98
- 99
- 100
- 101
- 102
- 103
- 104
- 105
- 106
- 107
- 108
- 109
- 110
- 111
- 112
- 113
- 114
- 115
- 116
- 117
- 118
- 119
- 120
- 121
- 122
- 123
- 124
- 125
- 126
- 127
- 128
- 129
- 130
- 131
- 132
- 133
- 134
- 135
- 136
- 137
- 138
- 139
- 140
- 141
- 142
- 143
- 144
- 145
- 146
- 147
- 148
- 149
- 150
- 151
- 152
- 153
- 154
- 155
- 156
- 157
- 158
- 159
- 160
- 161
- 162
- 163
- 164
- 165
- 166
- 167
- 168
- 169
- 170
- 171
- 172
- 173
- 174
- 175
- 176
- 177
- 178
- 179
- 180
- 181
- 182
- 183
- 184
- 185
- 186
- 187
- 188
- 189
- 190
- 191
- 192
- 193
- 194
- 195
- 196
- 197
- 198
- 199
- 200
- 201
- 202
- 203
- 204
- 205
- 206
- 207
- 208
- 209
- 210
- 211
- 212
- 213
- 214
- 215
- 216
- 217
- 218
- 219
- 220
- 221
- 222
- 223
- 224
- 225
- 226
- 227
- 228
- 229
- 230
- 231
- 232
- 233
- 234
- 235
- 236
- 237
- 238
- 239
- 240
- 241
- 242
- 243
- 244
- 245
- 246
- 247
- 248
- 249
- 250
- 251
- 252
- 253
- 254
- 255
- 256
- 257
- 258
- 259
- 260
- 261
- 262
- 263
- 264
- 265
- 266
- 267
- 268
- 269
- 270
- 271
- 272
- 273
- 274
- 275
- 276
- 277
- 278
- 279
- 280
- 281
- 282
- 283
- 284
- 285
- 286
- 287
- 288
- 289
- 290
- 291
- 292
- 293
- 294
- 295
- 296
- 297
- 298
- 299
- 300
- 301
- 302
- 303
- 304
- 305
- 306
- 307
- 308
- 309
- 310
- 311
- 312
- 313
- 314
- 315
- 316
- 317
- 318
- 319
- 320
- 321
- 322
- 323
- 324
- 325
- 326
- 327
- 328
- 329
- 330
- 331
- 332
- 333
- 334
- 335
- 336
- 337
- 338
- 339
- 340
- 341
- 342
- 343
- 344
- 345
- 346
- 347
- 348
- 349
- 350
- 351
- 352
- 353
- 354
- 355
- 356
- 357
- 358
- 359
- 360
- 361
- 362
- 363
- 364
- 365
- 366
- 367
- 368
- 369
- 370
- 371
- 372
- 373
- 374
- 375
- 376
- 377
- 378
- 379
- 380
- 381
- 382
- 383
- 384
- 385
- 386
- 387
- 388
- 389
- 390
- 391
- 392
- 393
- 394
- 395
- 396
- 397
- 398
- 399
- 400
- 401
- 402
- 403
- 404
- 405
- 406
- 407
- 408
- 409
- 410
- 411
- 412
- 413
- 414
- 415
- 416
- 417
- 418
- 419
- 420
- 421
- 422
- 423
- 424
- 425
- 426
- 427
- 428
- 429
- 430
- 431
- 432
- 433
- 434
- 435
- 436
- 437
- 438
- 439
- 440
- 441
- 442
- 443
- 444
- 445
- 446
- 447
- 448
- 449
- 450
- 451
- 452
- 453
- 454
- 455
- 456
- 457
- 458
- 459
- 460
- 461
- 462
- 463
- 464
- 465
- 466
- 467
- 468
- 469
- 470
- 471
- 472
- 473
- 474
- 475
- 476
- 477
- 478
- 479
- 480
- 481
- 482
- 483
- 484
- 485
- 486
- 487
- 488
- 489
- 490
- 491
- 492
- 493
- 494
- 495
- 496
- 497
- 498
- 499
- 500
- 501
- 502
- 503
- 504
- 505
- 506
- 507
- 508
- 509
- 510
- 511
- 512
- 513
- 514
- 515
- 516
- 517
- 518
- 519
- 520
- 521
- 522
- 523
- 524
- 525
- 526
- 527
- 528
- 529
- 530
- 531
- 532
- 533
- 534
- 535
- 536
- 537
- 538
- 539
- 540
- 541
- 542
- 543
- 544
- 545
- 546
- 547
- 548
- 549
- 550
- 551
- 552
- 553
- 554
- 555
- 556
- 557
- 558
- 559
- 560
- 561
- 562
- 563
- 564
- 565
- 566
- 567
- 568
- 569
- 570
- 571
- 572
- 573
- 574
- 575
- 576
- 577
- 578
- 579
- 580
- 581
- 582
- 583
- 584
- 585
- 586
- 587
- 588
- 589
- 590
- 591
- 592
- 593
- 594
- 595
- 596
- 597
- 598
- 599
- 600
- 601
- 602
- 603
- 604
- 605
- 606
- 607
- 608
- 609
- 610
- 611
- 612
- 613
- 614
- 615
- 616
- 617
- 618
- 619
- 620
- 621
- 622
- 623
- 624
- 625
- 626
- 627
- 628
- 629
- 630
- 631
- 632
- 633
- 634
- 635
- 636
- 637
- 638
- 639
- 640
- 641
- 642
- 643
- 644
- 645
- 646
- 647
- 648
- 649
- 650
- 651
- 652
- 653
- 654
- 655
- 656
- 657
- 658
- 659
- 660
- 661
- 662
- 663
- 664
- 665
- 666
- 667
- 668
- 669
- 670
- 671
- 672
- 673
- 674
- 675
- 676
- 677
- 678
- 679
- 680
- 681
- 682
- 683
- 684
- 685
- 686
- 687
- 688
- 689
- 690
- 691
- 692
- 693
- 694
- 695
- 696
- 697
- 698
- 699
- 700
- 701
- 702
- 703
- 704
- 705
- 706
- 707
- 708
- 709
- 710
- 711
- 712
- 713
- 714
- 715
- 716
- 717
- 718
- 719
- 720
- 721
- 722
- 723
- 724
- 725
- 726
- 727
- 728
- 729
- 730
- 731
- 732
- 733
- 734
- 735
- 736
- 737
- 738
- 739
- 740
- 741
- 742
- 743
- 744
- 745
- 746
- 747
- 748
- 749
- 750
- 751
- 752
- 753
- 754
- 755
- 756
- 757
- 758
- 759
- 760
- 761
- 762
- 763
- 764
- 765
- 766
- 767
- 768
- 769
- 770
- 771
- 772
- 773
- 774
- 775
- 776
- 777
- 778
- 779
- 780
- 781
- 782
- 783
- 784
- 785
- 786
- 787
- 788
- 789
- 790
- 791
- 792
- 793
- 794
- 795
- 796
- 797
- 798
- 799
- 800
- 801
- 802
- 803
- 804
- 805
- 806
- 807
- 808
- 809
- 810
- 811
- 812
- 813
- 814
- 815
- 816
- 817
- 818
- 819
- 820
- 821
- 822
- 823
- 824
- 825
- 826
- 827
- 828
- 829
- 830
- 831
- 832
- 833
- 834
- 835
- 836
- 837
- 838
- 839
- 840
- 841
- 842
- 843
- 844
- 845
- 846
- 847
- 848
- 849
- 850
- 851
- 852
- 853
- 854
- 855
- 856
- 857
- 858
- 859
- 860
- 861
- 862
- 863
- 864
- 865
- 866
- 867
- 868
- 869
- 870
- 871
- 872
- 873
- 874
- 875
- 876
- 877
- 878
- 879
- 880
- 881
- 882
- 883
- 884
- 885
- 88

- [Hoh+20] F. Hohman, M. Conlen, J. Heer, and D. H. P. Chau. "Communicating with interactive articles". In: *Distill* 5.9 (2020), e28.
- [Hol86] P. W. Holland. "Statistics and Causal Inference". In: *JASA* 81.396 (1986), pp. 945–960.
- [Hon+10] A. Honkela, T. Raiko, M. Kuusela, M. Tornio, and J. Karhunen. "Approximate Riemannian Conjugate Gradient Learning for Fixed-Form Variational Bayes". In: *JMLR* 11.Nov (2010), pp. 3235–3268.
- [Hor+05] E. Horvitz, J. Apacible, R. Sarin, and L. Liao. "Prediction, Expectation, and Surprise: Methods, Designs, and Study of a Deployed Traffic Forecasting Service". In: *UAI*. 2005.
- [Hor61] P. Horst. "Generalized canonical correlations and their applications to experimental data". en. In: *J. Clin. Psychol.* 17 (Oct. 1961), pp. 331–347.
- [Hos+20] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. "Meta-Learning in Neural Networks: A Survey". In: (Apr. 2020). arXiv: 2004.05439 [cs.LG].
- [HOT06] G. Hinton, S. Osindero, and Y. Teh. "A fast learning algorithm for deep belief nets". In: *Neural Computation* 18 (2006), pp. 1527–1554.
- [Hot36] H. Hotelling. "Relations Between Two Sets of Variates". In: *Biometrika* 28.3/4 (1936), pp. 321–377.
- [HOW11] P. Hall, J. T. Ormerod, and M. P. Wand. "Theory of Gaussian Variational Approximation for a Generalised Linear Mixed Model". In: *Statistica Sinica* 21 (2011), pp. 269–389.
- [HP10] J. Hacker and P. Pierson. *Winner-Take-All Politics: How Washington Made the Rich Richer — and Turned Its Back on the Middle Class*. Simon & Schuster, 2010.
- [HR20a] M. Hernan and J. Robins. *Causal Inference: What If*. CRC Press, 2020.
- [HR20b] M. Hernán and J. Robins. *Causal Inference: What If*. Boca Raton: Chapman & Hall/CRC., 2020.
- [HS06] G. Hinton and R. Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *Science* 313.5786 (2006), pp. 504–507.
- [HS09] M. Heaton and J. Scott. *Bayesian computation and the linear model*. Tech. rep. Duke, 2009.
- [HS12] P. Hennig and C. Schuler. "Entropy search for information-efficient global optimization". In: *JMLR* 13 (2012), pp. 1809–1837.
- [HS13] J. Y. Hsu and D. S. Small. "Calibrating Sensitivity Analyses to Observed Covariates in Observational Studies". In: *Biometrics* 69.4 (2013), pp. 803–811. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/biom.12101>.
- [HS18] D. Ha and J. Schmidhuber. "World Models". In: *NIPS*. 2018.
- [HS88] D. S. Hochbaum and D. B. Shmoys. "A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach". In: *SICOMP*. 1988.
- [HS97] S. Hochreiter and J. Schmidhuber. "Flat minima". en. In: *Neural Comput.* 9.1 (1997), pp. 1–42.
- [HSG06] J. D. Hol, T. B. Schön, and F. Gustafsson. "On Resampling Algorithms for Particle Filters". In: *2006 IEEE Nonlinear Statistical Signal Processing Workshop*. Sept. 2006, pp. 79–82.
- [HSGF21] S. Hassan, S. Särkkä, and Á. F. García-Fernández. "Temporal Parallelization of Inference in Hidden Markov Models". In: *IEEE Trans. Signal Processing* 69 (Feb. 2021), pp. 4875–4887.
- [Hu+18] Y.-C. Hsu, Y.-C. Liu, A. Ramasamy, and Z. Kira. "Re-evaluating Continual Learning Scenarios: A Categorization and Case for Strong Baselines". In: *NIPS Continual Learning Workshop*. Oct. 2018.
- [HT01] G. E. Hinton and Y. Teh. "Discovering multiple constraints that are frequently approximately satisfied". In: *UAI*. 2001.
- [HT09] H. Hoefling and R. Tibshirani. "Estimation of Sparse Binary Pairwise Markov Networks using Pseudo-likelihoods". In: *JMLR* 10 (2009).
- [HT15] J. H. Huggins and J. B. Tenenbaum. "Risk and regret of hierarchical Bayesian learners". In: *ICML*. May 2015.
- [HT17] T. J. Hastie and R. J. Tibshirani. *Generalized additive models*. Routledge, 2017.
- [HTF09] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. 2nd edition. Springer, 2009.
- [HTW15] T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. CRC Press, 2015.
- [Hu+00] M. Hu, C. Ingram, M. Sirski, C. Pal, S. Swamy, and C. Patten. *A Hierarchical HMM Implementation for Vertebrate Gene Splice Site Prediction*. Tech. rep. Dept. Computer Science, Univ. Waterloo, 2000.
- [Hu+12] J. Hu, Y. Wang, E. Zhou, M. C. Fu, and S. I. Marcus. "A Survey of Some Model-Based Methods for Global Optimization". en. In: *Optimization, Control, and Applications of Stochastic Systems*. Systems & Control: Foundations & Applications. Birkhäuser, Boston, 2012, pp. 157–179.
- [Hu+17] W. Hu, C. J. Li, L. Li, and J.-G. Liu. "On the diffusion approximation of nonconvex stochastic gradient descent". In: (May 2017). arXiv: 1705.07562 [stat.ML].
- [Hu+18] W. Hu, G. Niu, I. Sato, and M. Sugiyama. "Does Distributionally Robust Supervised Learning Give Robust Classifiers?". In: *ICML*. 2018.
- [Hu+17a] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. Hopcroft, and K. Weinberger. "Snapshot ensembles: train 1, get M for free". In: *ICLR*. 2017.
- [Hu+17b] Y. Huang, Y. Zhang, N. Li, Z. Wu, and J. A. Chambers. "A Novel Robust Student's t-Based Kalman Filter". In: *IEEE Trans. Aerosp. Electron. Syst.* 53.3 (June 2017), pp. 1545–1554.
- [Hu+18a] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinevlescu, and D. Eck. "Music Transformer". In: (Sept. 2018). arXiv: 1809.04281 [cs.LG].
- [Hu+18b] C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville. "Neural Autoregressive Flows". In: *ICML*. 2018.
- [Hu+19a] W. R. Huang, Z. Emam, M. Goldblum, L. Fowl, J. K. Terry, F. Huang, and T. Goldstein. "Understanding generalization through visualizations". In: *arXiv preprint arXiv:1906.03291* (2019).
- [Hu+19b] Y. Huang, Y. Zhang, Y. Zhao, and J. A. Chambers. "A Novel Robust Gaussian–Student's t Mixture Distribution Based Kalman Filter". In: *IEEE Trans. Signal Process.* 67.13 (July 2019), pp. 3606–3620.
- [Hug+19] J. H. Huggins, T. Campbell, M. Kasprzak, and T. Broderick. "Scalable Gaussian Process Inference with Finite-data Mean and Variance Guarantees". In: *AISTATS*. 2019.
- [Hug+20] J. H. Huggins, M. Kasprzak, T. Campbell, and T. Broderick. "Validated Variational Inference via Practical Posterior Error Bounds". In: *AISTATS*. Ed. by S. Chiappa and R. Calandriello. Vol. 108. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1792–1802.
- [Hus17a] F. Huszár. *Is Maximum Likelihood Useful for Representation Learning?* 2017.
- [Hus17b] F. Huszár. "Variational inference using implicit distributions". In: *arXiv preprint arXiv:1702.08235* (2017).
- [Hut89] M. F. Hutchinson. "A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines". In: *Communications in Statistics-Simulation and Computation* 18.3 (1989), pp. 1059–1076.
- [HV21] J. Helske and M. Virola. "bssm: Bayesian Inference of Non-linear and Non-Gaussian State Space Models in R". In: (Jan. 2021). arXiv: 2101.08492 [stat.CO].
- [HVD14] G. Hinton, O. Vinyals, and J. Dean. "Distilling the Knowledge in a Neural Network". In: *NIPS DL workshop*. 2014.
- [HW13] A. Huang and M. P. Wand. "Simple Marginally Noninformative Prior Distributions for Covariance Matrices". en. In: *Bayesian Analysis* 8.2 (June 2013), pp. 439–452.
- [HXW17] H. He, B. Xin, and D. Wipf. "From Bayesian Sparsity to Gated Recurrent Nets". In: *NIPS*. 2017.
- [HY01] M. Hansen and B. Yu. "Model selection and the principle of minimum description length". In: *JASA* (2001).
- [Hyv05] A. Hyvärinen. "Estimation of non-normalized statistical models by score matching". In: *JMLR* 6.Apr (2005), pp. 695–709.
- [Hyv07a] A. Hyvärinen. "Connections between score matching, contrastive divergence, and pseudolikelihood for continuous-valued variables". In: *IEEE Transactions on neural networks* 18.5 (2007), pp. 1529–1531.
- [Hyv07b] A. Hyvärinen. "Some extensions of score matching". In: *Computational statistics & data analysis* 51.5 (2007), pp. 2499–2512.
- [IB12] R. Iyer and J. Bilmes. "Algorithms for Approximate Minimization of the Difference Between Submodular Functions, with Applications". In: *Uncertainty in Artificial Intelligence (UAI)*. Catalina Island, USA: AUAI, 2012.
- [IB13] R. Iyer and J. Bilmes. "Submodular Optimization with Submodular Cover and Submodular Knapsack Constraints". In: *Neural Information Processing Society (NeurIPS, formerly NIPS)*. Lake Tahoe, CA, 2013.
- [IB15] R. K. Iyer and J. A. Bilmes. "Polyhedral aspects of Submodularity, Convexity and Concavity". In: *Arxiv, CoRR abs/1506.07329* (2015).
- [IB98] M. Isard and A. Blake. "CONDENSATION - conditional density propagation for visual tracking". In: *Intl. J. of Computer Vision* 29.1 (1998), pp. 5–18.

- 1 [IBK06] E. L. Ionides, C. Bretó, and A. A. King. “Inference  
2 for nonlinear dynamical systems”. en. In: *PNAS* 103.49 (Dec.  
3 2006), pp. 18438–18443.
- 4 [IFF00] S. Iwata, L. Fleischer, and S. Fujishige. “A combinatorial  
5 strongly polynomial algorithm for minimizing submodular  
functions”. In: *Journal of the ACM* (2000).
- 6 [IFW05] A. T. Ihler, J. W. Fischer III, and A. S. Willsky. “Loopy  
Belief Propagation: Convergence and Effects of Message Er-  
rors”. In: *JMLR* 6 (Dec. 2005), pp. 905–936.
- 7 [IJ01] H. Ishwaran and L. F. James. “Gibbs sampling methods  
for stick-breaking priors”. In: *Journal of the American Statistical  
Association* 96.453 (2001), pp. 161–173.
- 8 [IJB13] R. Iyer, S. Jegelka, and J. Bilmes. “Curvature and Optimal  
Algorithms for Learning and Minimizing Submodular Func-  
tions”. In: *NIPS*. Lake Tahoe, CA, 2013.
- 9 [IKB21] A. Immer, M. Korzepa, and M. Bauer. “Improving pre-  
dictions of Bayesian neural nets via local linearization”. In:  
*AISTATS*. Ed. by A. Banerjee and K. Fukumizu. Vol. 130. Pro-  
ceedings of Machine Learning Research. PMLR, 2021, pp. 703–  
711.
- 10 [IM17] J. Ingraham and D. Marks. “Bayesian Sparsity for In-  
tractable Undirected Models”. In: *ICML*. 2017.
- 11 [Imb03] G. Imbens. “Sensitivity to Exogeneity Assumptions in  
Program Evaluation”. In: *The American Economic Review*  
(2003).
- 12 [Imb03] G. W. Imbens. “Potential Outcome and Directed  
Acyclic Graph Approaches to Causality: Relevance for Empirical  
Practice in Economics”. In: (July 2019). arXiv: [1907.07271](https://arxiv.org/abs/1907.07271)  
[stat.ME].
- 13 [IN09] S. Iwata and K. Nagano. “Submodular function minimiza-  
tion under covering constraints”. In: *Proceedings of the 50th  
Annual IEEE Symposium on Foundations of Computer Science  
(FOCS)*. 2009, pp. 671–680.
- 14 [INK18] P. Izmailov, A. Novikov, and D. Kropotov. “Scalable  
Gaussian Processes with Billions of Inducing Inputs via Tensor  
Train Decomposition”. In: *ICML*. 2018.
- 15 [Inn20] M. Innes. “Sense &amp; Sensitivities: The Path to  
General-Purpose Algorithmic Differentiation”. In: *Proceedings  
of Machine Learning and Systems*. Ed. by I. Dhillon, D. Pa-  
pailopoulos, and V. Sze. Vol. 2. 2020, pp. 58–69.
- 16 [IO09] S. Iwata and J. B. Orlin. “A simple combinatorial al-  
gorithm for submodular function minimization”. In: *Proceedings  
of the twentieth annual ACM-SIAM symposium on Discrete  
algorithms*. SIAM, 2009, pp. 1230–1237.
- 17 [IRO0] D. R. Insua and F. Ruggeri. *Robust Bayesian Analysis*.  
Springer, 2000.
- 18 [IR10] A. Ilin and T. Raiko. “Practical Approaches to Princi-  
pal Component Analysis in the Presence of Missing Values”.  
In: *JMLR* 11 (2010), pp. 1957–2000.
- 19 [IR15] G. Imbens and D. Rubin. *Causal Inference in Statistics,  
Social and Biomedical Sciences: An Introduction*. Cambridge  
University Press, 2015.
- 20 [IS15] S. Ioffe and C. Szegedy. “Batch Normalization: Acceler-  
ating Deep Network Training by Reducing Internal Covariate  
Shift”. In: *ICML*. 2015, pp. 448–456.
- 21 [Isa03] M. Isard. “PAMPAS: Real-Valued Graphical Models for  
Computer Vision”. In: *CVPR*. Vol. 1. 2003, p. 613.
- 22 [Isl19] R. Islam, R. Seraj, S. Y. Arnon, and D. Precup. “Doubly  
Robust Off-Policy Actor-Critic Algorithms for Reinforce-  
ment Learning”. In: *NearIPS Workshop on Safety and Robust-  
ness in Decision Making*. 2019.
- 23 [Iso17] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. “Image-  
to-Image Translation with Conditional Adversarial Networks”.  
In: *CVPR*. 2017.
- 24 [IX00] K. Ito and K. Xiong. “Gaussian filters for nonlinear fil-  
tering problems”. In: *IEEE Trans. Automat. Contr.* 45.5 (May  
2000), pp. 910–927.
- 25 [Iye+21] R. Iyer, N. Khargonkar, J. Bilmes, and H. Asnani.  
“Generalized Submodular Information Measures: Theoretical  
Properties, Examples, Optimization Algorithms, and Applica-  
tions”. In: *IEEE Transactions on Information Theory* (2021).
- 26 [Izm+18] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov,  
and A. G. Wilson. “Averaging Weights Leads to Wider Optima  
and Better Generalization”. In: *UAI*. 2018.
- 27 [Izm+19] P. Izmailov, W. J. Maddox, P. Kirichenko, T. Garipov,  
D. Vetrov, and A. G. Wilson. “Subspace Inference for Bayesian  
Deep Learning”. In: *UAI*. 2019.
- 28 [Izm+21a] P. Izmailov, P. Nicholson, S. Lotfi, and A. G. Wilson.  
“Dangers of Bayesian Model Averaging under Covariate Shift”.  
In: *NIPS*. 2021.
- 29 [Izm+21b] P. Izmailov, S. Vikram, M. D. Hoffman, and A. G.  
Wilson. “What Are Bayesian Neural Network Posteriors Really  
Like?”. In: *ICML*. 2021.
- 30 [Jaa01] T. Jaakkola. “Tutorial on variational approximation  
methods”. In: *Advanced mean field methods*. Ed. by M. Opper  
and D. Saad. MIT Press, 2001.
- 31 [Jac+21] M. Jacobs, M. F. Pradier, T. H. McCoy, R. H. Perlis,  
F. Doshi-Velez, and K. Z. Gajos. “How machine-learning rec-  
ommendations influence clinician treatment selections: the ex-  
ample of antidepressant selection”. In: *Translational psychiatry*  
11.1 (2021), pp. 1–9.
- 32 [Jad+17] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul,  
J. Z. Leibo, D. Silver, and K. Kavukcuoglu. “Reinforcement  
Learning with Unsupervised Auxiliary Tasks”. In: *ICLR*. 2017.
- 33 [Jak21] K. Jakkala. “Deep Gaussian Processes: A Survey”. In:  
(June 2021). arXiv: [2106.12135](https://arxiv.org/abs/2106.12135) [cs.LG].
- 34 [Jan+17] P. A. Jang, A. Loeb, M. Davidow, and A. G. Wilson.  
“Scalable Levy Process Priors for Spectral Kernel Learning”. In:  
*Advances in Neural Information Processing Systems*. 2017.
- 35 [Jan18] E. Jang. *Normalizing Flows Tutorial*. 2018.
- 36 [Jan+19] M. Janner, J. Fu, M. Zhang, and S. Levine. “When  
to Trust Your Model: Model-Based Policy Optimization”. In:  
*NIPS*. 2019.
- 37 [Jas] Jason Antic and Jeremy Howard and Uri Manor. *Decrap-  
ification, DeOldification, and Super Resolution (Blog post)*.
- 38 [Jay03] E. T. Jaynes. *Probability theory: the logic of science*.  
Cambridge university press, 2003.
- 39 [Jay+20] S. M. Jayakumar, W. M. Czarnecki, J. Menick, J.  
Schwarz, J. Rae, S. Osindero, Y. W. Teh, T. Harley, and R. Pas-  
canu. “Multiplicative Interactions and Where to Find Them”.  
In: *ICLR*. 2020.
- 40 [JB03] A. Jakulin and I. Bratko. “Analyzing Attribute Depen-  
dencies”. In: *Proc. 7th European Conf. on Principles and Practi-  
ce of Knowledge Discovery in Databases*. 2003.
- 41 [JB16a] S. Jegelka and J. Bilmes. “Graph cuts with interacting  
edge weights: examples, approximations, and algorithms”. In:  
*Mathematical Programming* (2016), pp. 1–42.
- 42 [JB16b] R. Jonschkowski and O. Brock. “End-to-end learnable  
histogram filters”. In: *NIPS Workshop on Deep Learning for  
Action and Interaction*. 2016.
- 43 [JB65] C. Jacobi and C. W. Borchardt. “De investigando or-  
dine systematis aequationum differentialium vulgarium cuius-  
cunque”. In: *Journal für die reine und angewandte Mathematik*  
1865.64 (1865), pp. 297–320.
- 44 [Jef04] R. Jeffrey. *Subjective Probability: The Real Thing*. Cam-  
bridge, 2004.
- 45 [Jen+17] R. Jenatton, C. Archambeau, J. González, and M.  
Seeger. “Bayesian Optimization with Tree-structured Depen-  
dencies”. en. In: *ICML*. July 2017, pp. 1655–1664.
- 46 [Jeu+19] O. Jeunen, D. Mykhaylov, D. Rohde, F. Vasile, A.  
Gilotte, and M. Bompaire. “Learning from Bandit Feedback:  
An Overview of the State-of-the-art”. In: (Sept. 2019). arXiv:  
[1909.08471](https://arxiv.org/abs/1909.08471) [cs.IR].
- 47 [JG20] A. Jacovi and Y. Goldberg. “Towards faithfully inter-  
pretable NLP systems: How should we define and evaluate faith-  
fulness?”. In: *arXiv preprint arXiv:2004.03685* (2020).
- 48 [JG21] A. Jacovi and Y. Goldberg. “Aligning faithful interpre-  
tations with their social attribution”. In: *Transactions of the  
Association for Computational Linguistics* 9 (2021), pp. 294–  
310.
- 49 [JGH18] A. Jacot, F. Gabriel, and C. Hongler. “Neural Tangent  
Kernel: Convergence and Generalization in Neural Networks”.  
In: *NIPS*. 2018.
- 50 [JGP17] E. Jang, S. Gu, and B. Poole. “Categorical Reparam-  
eterization with Gumbel-Softmax”. In: *ICLR*. 2017.
- 51 [Jia+19] R. Jia, A. Raghunathan, K. Göksel, and P. Liang.  
“Certified Robustness to Adversarial Word Substitutions”. In:  
*EMNLP*. 2019.
- 52 [Jia21] H. Jiang. “Minimizing convex functions with integral  
minimizers”. In: *Proceedings of the 2021 ACM-SIAM Sym-  
posium on Discrete Algorithms (SODA)*. SIAM, 2021, pp. 976–  
985.
- 53 [Jih+12] Jihong Min, J Kim, Seunghak Shin, and I. S. Kweon.  
“Efficient Data-Driven MCMC sampling for vision-based 6D  
SLAM”. In: *ICRA*. May 2012, pp. 3025–3032.
- 54 [Jit+16] W. Jitkrittum, Z. Szabó, K. P. Chwialkowski, and  
A. Gretton. “Interpretable Distribution Features with Maxi-  
mum Testing Power”. In: *NIPS*. Curran Associates, Inc., 2016,  
pp. 181–189.
- 55 [JJ00] T. S. Jaakkola and M. I. Jordan. “Bayesian parameter  
estimation via variational methods”. In: *Statistics and Com-  
puting* 10 (2000), pp. 25–37.
- 56 [Jih94] F. V. Jensen and F. Jensen. “Optimal Junction Trees”.  
In: *UAI*. 1994.
- 57 [JKJ12] K. Jiang, B. Kulis, and M. Jordan. “Small-variance  
asymptotics for exponential family Dirichlet process mixture  
models”. In: *Advances in Neural Information Processing Sys-  
tems* 25 (2012), pp. 3158–3166.
- 58 [JKK95] C. S. Jensen, A. Kong, and U. Kjaerulff. “Blocking-  
Gibbs Sampling in Very Large Probabilistic Expert Systems”.  
In: *Int'l. J. Human-Computer Studies* (1995), pp. 647–666.

- 1
- [JL15a] V. Jalali and D. Leake. "CBR meets big data: A case study of large-scale adaptation rule generation". In: *International Conference on Case-Based Reasoning*. Springer. 2015, pp. 181–196.
- 2
- [JL15b] V. Jalali and D. B. Leake. "Enhancing case-based regression with automatically-generated ensembles of adaptations". In: *Journal of Intelligent Information Systems* 46 (2015), pp. 237–258.
- 3
- [JL16] N. Jiang and L. Li. "Doubly Robust Off-policy Evaluation for Reinforcement Learning". In: *ICML*. 2016, pp. 652–661.
- 4
- [JM00] D. Jurafsky and J. H. Martin. *Speech and language processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall, 2000.
- 5
- [JM08] D. Jurafsky and J. H. Martin. *Speech and language processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2nd edition. Prentice-Hall, 2008.
- 6
- [JM18] A. Jolicoeur-Martineau. "The relativistic discriminator: a key element missing from standard GAN". In: *arXiv preprint arXiv:1807.00794* (2018).
- 7
- [JM70] D. H. Jacobson and D. Q. Mayne. *Differential Dynamic Programming*. Elsevier Press, 1970.
- 8
- [JMW06] J. K. Johnson, D. M. Malioutov, and A. S. Willsky. "Walk-sum interpretation and analysis of Gaussian belief propagation". In: *NIPS*. 2006, pp. 579–586.
- 9
- [JNJ20] C. Jin, P. Netrapalli, and M. I. Jordan. "What is local optimality in non-convex-concave minimax optimization?" In: *Proceedings of the 37th International Conference on Machine Learning - Volume 73*. 2020.
- 10
- [JO18] M. Jankowiak and F. Obermeyer. "Pathwise Derivatives Beyond the Reparameterization Trick". In: *ICML*. 2018.
- 11
- [JOA10] T. Jaksch, R. Ortner, and P. Auer. "Near-optimal Regret Bounds for Reinforcement Learning". In: *JMLR* 11 (2010), pp. 1563–1600.
- 12
- [JOA17] P. E. Jacob, J. O'Leary, and Y. F. Atchadé. "Unbiased markov chain monte carlo with couplings". In: *arXiv preprint arXiv:1708.03625* (2017).
- 13
- [Joh12] M. J. Johnson. "A Simple Explanation of A Spectral Algorithm for Learning Hidden Markov Models". In: (Apr. 2012). arXiv: 1204.2477 [stat.ME].
- 14
- [Jon01] D. R. Jones. "A Taxonomy of Global Optimization Methods Based on Response Surfaces". In: *J. Global Optimiz.* 21.4 (Dec. 2001), pp. 345–383.
- 15
- [Jor07] M. I. Jordan. *An Introduction to Probabilistic Graphical Models*. In preparation. 2007.
- 16
- [Jor11] M. I. Jordan. "The era of Big Data". In: *ISBA Bulletin*. Vol. 18. 2011, pp. 1–3.
- 17
- [Jor+98] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. "An introduction to variational methods for graphical models". In: *Learning in Graphical Models*. Ed. by M. Jordan. MIT Press, 1998.
- 18
- [Jos20] C. Joshi. *Transformers are Graph Neural Networks*. Tech. rep. 2020.
- 19
- [Jos+22] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun. "Hands-on Bayesian Neural Networks – a Tutorial for Deep Learning Users". In: (July 2022).
- 20
- [JP95] R. Jirousek and S. Preucil. "On the effective implementation of the iterative proportional fitting procedure". In: *Computational Statistics & Data Analysis* 19 (1995), pp. 177–189.
- 21
- [JS19] C. Ji and H. Shen. "Stochastic Variational Inference via Upper Bound". In: (Dec. 2019). arXiv: 1912.00650 [cs.LG].
- 22
- [JS93] M. Jerrum and A. Sinclair. "Polynomial-time approximation algorithms for the Ising model". In: *SIAM J. on Computing* 22 (1993), pp. 1087–1116.
- 23
- [JS96] M. Jerrum and A. Sinclair. "The Markov chain Monte Carlo method: an approach to approximate counting and integration". In: *Approximation Algorithms for NP-hard problems*. Ed. by D. S. Hochbaum. PWS Publishing, 1996.
- 24
- [JSY19] P. Jaini, K. A. Selby, and Y. Yu. "Sum-of-Squares Polynomial Flow". In: *ICML*. 2019, pp. 3009–3018.
- 25
- [JU97] S. Julier and J. Uhlmann. "A New Extension of the Kalman Filter to Nonlinear Systems". In: *Proc. of AeroSense: The 11th Intl. Symp. on Aerospace/Defence Sensing, Simulation and Controls*. 1997.
- 26
- [JW14] M. Johnson and A. Willsky. "Stochastic Variational Inference for Bayesian Time Series Models". In: *ICML*. 2014, pp. 1854–1862.
- 27
- [JW19] S. Jain and B. C. Wallace. "Attention is not explanation". In: *arXiv preprint arXiv:1902.10186* (2019).
- 28
- [Kaa12] Kaare Brandt Petersen and Michael Syskind Pedersen. *The Matrix Cookbook*. 2012.
- 29
- [KAH19] F. H. Kingma, P. Abbeel, and J. Ho. "Bit-Swap: Recursive Bits-Back Coding for Lossless Compression with Hierarchical Latent Variables". In: *ICML*. 2019.
- 30
- [Kai58] H. Kaiser. "The varimax criterion for analytic rotation in factor analysis". In: *Psychometrika* 23.3 (1958).
- 31
- [Kal02] S. M. Kakade. "A Natural Policy Gradient". In: *NIPS*. 2002, pp. 1531–1538.
- 32
- [Kal06] O. Kallenberg. *Foundations of modern probability*. Springer Science & Business Media, 2006.
- 33
- [Kal+11] M. Kalakrishnan, S. Chitta, E. A. Theodorou, P. Pastor, and S. Schaal. "STOMP: Stochastic Trajectory Optimization for Motion Planning". In: *ICRA*. 2011, pp. 4569–4574.
- 34
- [Kal+18a] D. Kalashnikov et al. "QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation". In: *CORL*. 2018.
- 35
- [Kal+18b] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. Oord, S. Dieleman, and K. Kavukcuoglu. "Efficient neural audio synthesis". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2410–2419.
- 36
- [Kam16] E. Kamar. "Directions in Hybrid Intelligence: Completing AI Systems with Human Intelligence." In: *IJCAI*. 2016, pp. 4070–4073.
- 37
- [Kan+15] N. Kantas, A. Doucet, S. S. Singh, J. Maciejowski, and N. Chopin. "On Particle Methods for Parameter Estimation in State-Space Models". In: *Stat. Sci.* 30.3 (Aug. 2015), pp. 328–351.
- 38
- [Kan+20] T. Kaneko, H. Kameoka, K. Tanaka, and N. Hojo. "CycleGAN-VC3: Examining and Improving CycleGAN-VCs for Mel-spectrogram Conversion". In: *Interspeech conference proceedings* (2020).
- 39
- [Kan42] L. Kantorovich. "On the transfer of masses (in Russian)". In: *Doklady Akademii Nauk* 37.2 (1942), pp. 227–229.
- 40
- [Kar+18] T. Karras, T. Aila, S. Laine, and J. Lehtinen. "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *ICLR*. 2018.
- 41
- [Kar+20a] A.-H. Karimi, G. Barthe, B. Balle, and I. Valera. "Model-Agnostic Counterfactual Explanations for Consequential Decisions". 2020. arXiv: 1905.11190 [cs.LG].
- 42
- [Kar+20b] A.-H. Karimi, G. Barthe, B. Schölkopf, and I. Valera. "A survey of algorithmic recourse: definitions, formulations, solutions, and prospects". In: *arXiv preprint arXiv:2010.04050* (2020).
- 43
- [Kar+20c] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. "Analyzing and improving the image quality of stylegan". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8110–8119.
- 44
- [Kar+21] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila. "Alias-Free Generative Adversarial Networks". In: *arXiv preprint arXiv:2106.12423* (2021).
- 45
- [Kat05] T. Katayama. *Subspace Methods for Systems Identification*. Springer Verlag, 2005.
- 46
- [Kat+06] H. G. Katzgraber, S. Trebst, D. A. Huse, and M. Troyer. "Feedback-optimized parallel tempering Monte Carlo". In: *Journal of Statistical Mechanics: Theory and Experiment* 2006.03 (2006), P03018.
- 47
- [Kat+17] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. "Reluplex: An efficient SMT solver for verifying deep neural networks". In: *International Conference on Computer Aided Verification*. Springer. 2017, pp. 97–117.
- 48
- [Kat+19] N. Kato, H. Osone, K. Oomori, C. W. Ooi, and Y. Ochiai. "GANs-Based Clothes Design: Pattern Maker Is All You Need to Design Clothing". In: *Proceedings of the 10th Augmented Human International Conference 2019*. New York, NY, USA: Association for Computing Machinery, 2019.
- 49
- [Kau+19] V. Kaushal, R. Iyer, S. Kothawade, R. Mahadev, K. Doctor, and G. Ramakrishnan. "Learning from less data: A unified data subset selection and active learning framework for computer vision". In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019, pp. 1289–1299.
- 50
- [Kau+21] D. Kaushik, A. Setlur, E. Hovy, and Z. C. Lipton. "Explaining The Efficacy of Counterfactually Augmented Data". In: *ICLR*. 2021.
- 51
- [KB00] H. J. Kushner and A. S. Budhiraja. "A nonlinear filtering algorithm based on an approximation of the conditional distribution". In: *IEEE Trans. Automat. Contr.* 45.3 (Mar. 2000), pp. 580–585.
- 52
- [KB14a] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).
- 53
- [KB14b] K. Kirchhoff and J. Bilmes. "Submodularity for Data Selection in Machine Translation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014.
- 54
- [KB15] D. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *ICLR*. 2015.

- 1
- 2 [KB16] T. Kim and Y. Bengio. "Deep directed generative models  
with energy-based probability estimation". In: *arXiv preprint  
arXiv:1606.03439* (2016).
- 3 [KBH19] F. Kunstner, L. Balles, and P. Hennig. "Limitations of  
the Empirical Fisher Approximation". In: (May 2019). arXiv:  
1905.12558 [cs.LG].
- 4 [KCC20] V. Kumar, A. Choudhary, and E. Cho. "Data Augmen-  
tation using Pre-trained Transformer Models". In: *Proceedings  
of the 2nd Workshop on Life-long Learning for Spoken Lan-  
guage Systems*. Suzhou, China: Association for Computational  
Linguistics, Dec. 2020, pp. 18–26.
- 5 [KD18a] S. Kamthe and M. P. Deisenroth. "Data-Efficient Re-  
inforcement Learning with Probabilistic Model Predictive Con-  
trol". In: *AISTATS*. 2018.
- 6 [KD18b] D. P. Kingma and P. Dhariwal. "Glow: Generative Flow  
with Invertible  $1 \times 1$  Convolutions". In: *NIPS*. 2018.
- 7 [Kei+19a] L. Ke, M. Barnes, W. Sun, G. Lee, S. Choudhury, and  
S. Srinivasan. "Imitation Learning as  $f$ -Divergence Minimiza-  
tion". In: *arXiv preprint arXiv:1905.12888* (2019).
- 8 [Kei+19b] L. Ke, S. Choudhury, M. Barnes, W. Sun, G. Lee, and  
S. Srinivasan. "Imitation Learning as  $f$ -Divergence Minimiza-  
tion". arXiv:1905.12888. 2019.
- 9 [Kei06] F. C. Keil. "Explanation and understanding". In: *Annu.  
Rev. Psychol.* 57 (2006), pp. 227–254.
- 10 [Kel+20] J. Kelly, J. Bettencourt, M. J. Johnson, and D. Duve-  
naud. "Learning Differential Equations that are Easy to Solve". In:  
*Neural Information Processing Systems*. 2020.
- 11 [Kem+06] C. Kemp, J. Tenenbaum, T. Y. T. Griffiths and,  
and N. Ueda. "Learning systems of concepts with an infinite rela-  
tional model". In: *AAAI*. 2006.
- 12 [Kem+10] C. Kemp, J. Tenenbaum, S. Niyogi, and T. Griffiths.  
"A probabilistic model of theory formation". In: *Cognition* 114  
(2010), pp. 165–196.
- 13 [Ken16] E. H. Kennedy. "Semiparametric theory and empirical  
processes in causal inference". In: *Statistical causal inferences  
and their applications in public health research*. ICSA Book Ser.  
Stat. Springer, [Cham], 2016, pp. 141–167.
- 14 [Ken17] E. H. Kennedy. *Semiparametric theory*. 2017. arXiv:  
1709.06418 [stat.ME].
- 15 [Ken20] E. H. Kennedy. *Optimal doubly robust estimation of  
heterogeneous causal effects*. 2020. arXiv: 2004.14497 [math.ST].
- 16 [Kes+17] N. S. Keskar, D. Mudigere, J. Nocedal, M. Simeyan-  
skiy, and P. T. P. Tang. "On Large-Batch Training for Deep  
Learning: Generalization Gap and Sharp Minima". In: *ICLR*.  
2017.
- 17 [KF09a] D. Koller and N. Friedman. *Probabilistic Graphical  
Models: Principles and Techniques*. MIT Press, 2009.
- 18 [KF09b] D. Krishnan and R. Fergus. "Fast Image Deconvolution  
using Hyper-Laplacian Priors". In: *NIPS*. 2009, pp. 1033–1041.
- 19 [KFL01] F. Kschischang, B. Frey, and H.-A. Loeliger. "Factor  
Graphs and the Sum-Product Algorithm". In: *IEEE Trans Info.  
Theory* (2001).
- 20 [KG05] A. Krause and C. Guestrin. "Near-optimal Nonmyopic  
Value of Information in Graphical Models". In: *Proc. of the 21st  
Annual Conf. on Uncertainty in Artificial Intelligence (UAI  
2005)*. AUAI Press, 2005, pp. 324–331.
- 21 [KG17] A. Kendall and Y. Gal. "What Uncertainties Do We  
Need in Bayesian Deep Learning for Computer Vision?" In:  
*NIPS*. Curran Associates, Inc., 2017, pp. 5574–5584.
- 22 [KGW11] M. Kalli, J. E. Griffin, and S. G. Walker. "Slice  
sampling mixture models". In: *Statistics and computing* 21.1  
(2011), pp. 93–105.
- 23 [Kha+10] M. E. Khan, B. Marlin, G. Bouchard, and K. P. Mur-  
phy. "Variational bounds for mixed-data factor analysis". In:  
*NIPS*. 2010.
- 24 [Kha12] M. E. Khan. "Variational Bounds for Learning and In-  
ference with Discrete Data in Latent Gaussian Models". PhD  
thesis, UBC, 2012.
- 25 [Kha+18] M. E. Khan, D. Nielsen, V. Tangkarratt, W. Lin, Y.  
Gal, and A. Srivastava. "Fast and Scalable Bayesian Deep  
Learning by Weight-Perturbation in Adam". In: *ICML*. 2018.
- 26 [Kha20] M. E. Khan. *Deep learning with Bayesian principles*.  
NeurIPS tutorial. 2020.
- 27 [Kha+21] S. Khan, M. Nasir, M. Hayat, S. W. Zamir, F. S.  
Khan, and M. Shah. "Transformers in Vision: A Survey". In:  
*ACM Computing Surveys December* (Jan. 2021).
- 28 [KHJ20] A. Kristiadi, M. Hein, and P. Hennig. "Being Bayesian,  
Even Just a Bit, Fixes Overconfidence in ReLU Networks". In:  
*ICML*. 2020.
- 29 [KHL20] D. Kaushik, E. Hovy, and Z. C. Lipton. *Learning  
the Difference that Makes a Difference with Counterfactually-  
Augmented Data*. 2020. arXiv: 1909.12434 [cs.CL].
- 30 [Kil+20] K. Killamsetty, D. Sivasubramanian, G. Ramakrish-  
nan, and R. Iyer. "GLISTER: Generalization based Data Subset  
Selection for Efficient and Robust Learning". In: *arXiv preprint  
arXiv:2012.10630* (2020).
- 31 [Kim+18a] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler,  
F. Viégas, and R. Sayres. "Interpretability Beyond Feature At-  
tribution: Quantitative Testing with Concept Activation Vec-  
tors (tcaV)". In: *ICML*. 2018.
- 32 [Kim+18b] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler,  
F. Viégas, et al. "Interpretability beyond feature at-  
tribution: Quantitative testing with concept activation vec-  
tors (tcaV)". In: *International conference on machine learning*.  
PMLR, 2018, pp. 2668–2677.
- 33 [Kim+18c] Y. Kim, S. Wiseman, A. C. Miller, D. Sontag, and  
A. M. Rush. "Semi-Amortized Variational Autoencoders". In:  
*ICML*. 2018.
- 34 [Kim+19] S. Kim, S.-G. Lee, J. Song, J. Kim,, and S. Yoon.  
"FlowWaveNet: A Generative Flow for Raw Audio". In: *Pro-  
ceedings of the 36th International Conference on Machine Learn-  
ing*. 2019, pp. 3370–3378.
- 35 [Kin+14] D. P. Kingma, D. J. Rezende, S. Mohamed, and  
M. Welling. "Semi-Supervised Learning with Deep Generative  
Models". In: *NIPS*. 2014.
- 36 [Kin+16] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen,  
I. Sutskever, and M. Welling. "Improved Variational Inference  
with Inverse Autoregressive Flow". In: *NIPS*. 2016.
- 37 [Kin+19] P.-J. Kindermans, S. Hooker, J. Adebayo, M. Alber,  
K. T. Schütt, S. Dähne, D. Erhan, and B. Kim. "The (un) re-  
liability of saliency methods". In: *Explainable AI: Interpre-  
ting, Explaining and Visualizing Deep Learning*. Springer, 2019,  
pp. 267–280.
- 38 [Kir+17] J. Kirkpatrick et al. "Overcoming catastrophic forget-  
ing in neural networks". en. In: *PNAS* 114.13 (2017), pp. 3521–  
3526.
- 39 [Kir+21] A. Kirsch, J. M. J. van Amersfoort, P. H. S. Torr, and  
Y. Gal. "On pitfalls in OoD detection: Predictive entropy con-  
sidered harmful". In: *ICML Workshop on Uncertainty in Deep  
Learning*. 2021.
- 40 [Kit04] G. Kitagawa. "The two-filter formula for smoothing and  
an implementation of the Gaussian-sum smoother". In: *Annals  
of the Institute of Statistical Mathematics* 46.4 (2004), pp. 605–  
623.
- 41 [Kiw20] P. Kirichenko, P. Izmailov, and A. G. Wilson. "Why  
Normalizing Flows Fail to Detect Out-of-Distribution Data". In:  
(June 2020). arXiv: 2006.08545 [stat.ML].
- 42 [KJ12a] J. Z. Kolter and T. S. Jaakkola. "Approximate Infer-  
ence in Additive Factorial HMMs with Application to Energy  
Disaggregation". In: *AISTATS*. 2012.
- 43 [KJ12b] B. Kulis and M. I. Jordan. "Revisiting K-Means: New  
Algorithms via Bayesian Nonparametrics". In: *Proceedings of  
the 29th International Conference on International Conference  
on Machine Learning, ICML'12*. Edinburgh, Scotland: Omnipress,  
2012, 1131–1138.
- 44 [Kja90] U. Kjaerulff. *Triangulation of graphs – algorithms giving  
all partial state space*. Tech. rep. R-90-09. Dept. of Math.  
and Comp. Sci., Aalborg Univ., Denmark, 1990.
- 45 [Kja92] U. Kjaerulff. "Optimal decomposition of probabilistic  
networks by simulated annealing". In: *Statistics and Comput-  
ing*. Vol. 2. 1992, pp. 7–17.
- 46 [KJD18] J. Knoblauch, J. Jewson, and T. Damoulas. "Doubly  
Robust Bayesian Inference for Non-Stationary Streaming Data  
with  $\beta$ -Divergences". In: *NIPS*. June 2018.
- 47 [KJD19] J. Knoblauch, J. Jewson, and T. Damoulas. "General-  
ized Variational Inference: Three arguments for deriving new  
Posteriors". In: (Apr. 2019). arXiv: 1904.02063 [stat.ML].
- 48 [KJD21] J. Knoblauch, J. Jewson, and T. Damoulas. "An  
Optimization-centric View on Bayes' Rule: Reviewing and Gen-  
eralizing Variational Inference". In: *JMLR* (2021).
- 49 [KJM19] N. M. Krieger, F. D. Johansson, and C. Morris. "A Sur-  
vey on Graph Kernels". In: (Mar. 2019). arXiv: 1903.11835 [cs.GC].
- 50 [KJV83] S. Kirkpatrick, C. G. Jr., and M. Vecchi. "Optimiza-  
tion by simulated annealing". In: *Science* 220 (1983), pp. 671–  
680.
- 51 [KK11] P. Krähenbühl and V. Koltun. "Efficient Inference in  
Fully Connected CRFs with Gaussian Edge Potentials". In:  
*NIPS*. 2011.
- 52 [KKH20] I. Khemakhem, D. P. Kingma, and A. Hyvärinen.  
"Variational Autoencoders and Nonlinear ICA: A Unifying  
Framework". In: *AISTATS*. 2020.
- 53 [KKL20] N. Kitae, L. Kaiser, and A. Levskaya. "Reformer: The  
Efficient Transformer". In: *8th International Conference on  
Learning Representations, ICLR 2020, Addis Ababa, Ethiopia,  
April 26-30, 2020*. OpenReview.net, 2020.
- 54 [KKR95] K. Kanazawa, D. Koller, and S. Russell. "Stochastic  
Simulation Algorithms for Dynamic Probabilistic Networks".  
In: *UAI*. 1995.
- 55 [KKS20] F. Kunstner, R. Kumar, and M. Schmidt.  
"Homeomorphic-Invariance of EM: Non-Asymptotic Conver-  
gence in KL Divergence for Exponential Families via Mirror  
Descent". In: (Nov. 2020). arXiv: 2011.01170 [cs.LG].

- [KKT03] D. Kempe, J. Kleinberg, and É. Tardos. "Maximizing the spread of influence through a social network". In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2003, pp. 137–146.
- [KL02] S. Kakade and J. Langford. "Approximately Optimal Approximate Reinforcement Learning". In: *ICML*. ICML '02, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 267–274.
- [KL09] H. Kawakatsu and A. Largey. "EM algorithms for ordered probit models with endogenous regressors". In: *The Econometrics Journal* 12.1 (2009), pp. 164–186.
- [KL10] D. P. Kingma and Y. LeCun. "Regularized estimation of image statistics by score matching". In: *Advances in neural information processing systems*. 2010, pp. 1126–1134.
- [KL17a] M. E. Khan and W. Lin. "Conjugate-Computation Variational Inference : Converting Variational Inference in Non-Conjugate Models to Inferences in Conjugate Models". In: *AISTATS*. 2017.
- [KL17b] P. W. Koh and P. Liang. "Understanding black-box predictions via influence functions". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1885–1894.
- [KL21a] W. M. Kouw and M. Loog. "A review of domain adaptation without target labels". en. In: *IEEE PAMI* (Oct. 2021).
- [KL21b] W. M. Kouw and M. Loog. "A review of domain adaptation without target labels". en. In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2021).
- [KLA19] T. Karras, S. Laine, and T. Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *CVPR*. 2019.
- [Kle17] G. A. Klein. *Sources of power: How people make decisions*. MIT press, 2017.
- [KLM19] A. Kumar, P. Liang, and T. Ma. "Verified Uncertainty Calibration". In: *NIPS*. 2019.
- [KM00] J. Kwon and K. Murphy. *Modeling Freeway Traffic with Coupled HMMs*. Tech. rep. Univ. California, Berkeley, 2000.
- [KM08] U. Kjaerulff and A. Madsen. *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis*. Springer, 2008.
- [KMR16] J. Kleinberg, S. Mulaianathan, and M. Raghavan. "Inherent trade-offs in the fair determination of risk scores". In: *arXiv preprint arXiv:1609.05807* (2016).
- [KMY04] D. Kersten, P. Mamassian, and A. Yuille. "Object perception as Bayesian inference". en. In: *Annu. Rev. Psychol.* 55 (2004), pp. 271–304.
- [KN09] J. Z. Kolter and A. Y. Ng. "Near-Bayesian Exploration in Polynomial Time". In: *ICML*. 2009.
- [KN95] R. Kneser and H. Ney. "Improved backing-off for n-gram language modeling". In: *ICASSP*. Vol. 1. 1995, pp. 181–184.
- [KNT20] I. Kostrikov, O. Nachum, and J. Tompson. "Imitation Learning via Off-Policy Distribution Matching". In: *ICLR*. 2020.
- [Koe05] R. Koenker. *Quantile Regression*. en. Cambridge University Press, May 2005.
- [Koh+20] P. W. Koh, T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim, and P. Liang. "Concept bottleneck models". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 5338–5348.
- [Koo03] G. Koop. *Bayesian econometrics*. Wiley, 2003.
- [Kor+15] A. Korattikara, V. Rathod, K. Murphy, and M. Welling. "Bayesian Dark Knowledge". In: *NIPS*. 2015.
- [Kor+20] A. Korotin, V. Egiazarian, A. Asadulaev, A. Safin, and E. Burnaev. "Wasserstein-2 Generative Networks". In: *International Conference on Learning Representations*. 2020.
- [Kot+17] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown. "Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA". In: *JMLR* 18.25 (2017), pp. 1–5.
- [Kot+22] S. Kothawade, V. Kaushal, G. Ramakrishnan, J. Bilmes, and R. Iyer. "PRISM: A Rich Class of Parameterized Submodular Information Measures for Guided Subset Selection". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2022.
- [Koy+10] S. Koyama, L. C. Pérez-Bolde, C. R. Shalizi, and R. E. Kass. "Approximate Methods for State-Space Models". en. In: *JASA* 105.489 (Mar. 2010), pp. 170–180.
- [KP20] A. Kumar and B. Poole. "On Implicit Regularization in  $\beta$ -VAEs". In: *ICML*. 2020.
- [KPB19] I. Kobyzhev, S. Prince, and M. A. Brubaker. "Normalizing Flows: An Introduction and Review of Current Methods". In: (Aug. 2019). arXiv: 1908.09257 [stat.ML].
- [KPHL17] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato. "Grammar Variational Autoencoder". In: *ICML*. 2017.
- [KPT21] J. S. Kim, G. Plumb, and A. Talwalkar. "Sanity Simulations for Saliency Methods". In: *arXiv preprint arXiv:2105.06506* (2021).
- [KR21a] M. Khan and H. Rue. "The Bayesian Learning Rule". In: (2021).
- [KR21b] S. Kong and D. Ramanan. "OpenGAN: Open-Set Recognition via 'Open Data Generation'". In: *ICCV*. Apr. 2021.
- [Kra+08] A. Krause, H. Brendan McMahan, C. Guestrin, and A. Gupta. "Robust Submodular Observation Selection". In: *JMLR* 9 (2008), pp. 2761–2801.
- [Kri+05] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink. "Learning sparse Bayesian classifiers: multi-class formulation, fast algorithms, and generalization bounds". In: *IEEE Transaction on Pattern Analysis and Machine Intelligence* (2005).
- [KRL08] K. Kavukcuoglu, M. Ranzato, and Y. LeCun. "Fast Inference in Sparse Coding Algorithms with Applications to Object Recognition". In: *NIPS workshop on optimization in machine learning*. 2008.
- [KRS14] B. Kim, C. Rudin, and J. A. Shah. "The bayesian case model: A generative approach for case-based reasoning and prototype classification". In: *Advances in neural information processing systems*. 2014, pp. 1952–1960.
- [Kru13] J. K. Kruschke. "Bayesian estimation supersedes the t test". In: *J. Experimental Psychology: General* 142.2 (2013), pp. 573–603.
- [KS06] L. Kocsis and C. Szepesvári. "Bandit Based Monte-Carlo Planning". In: *ECCML*. 2006, pp. 282–293.
- [KS07] J. D. Y. Kang and J. L. Schafer. "Demystifying double robustness: a comparison of alternative strategies for estimating a population mean from incomplete data". In: *Statist. Sci.* 22.4 (2007), pp. 523–539.
- [KS15] H. Kaya and A. A. Salah. "Adaptive Mixtures of Factor Analyzers". In: (July 2015). arXiv: 1507.02801 [stat.ML].
- [KSB21] B. Kompa, J. Snoek, and A. Beam. "Empirical Frequentist Coverage of Deep Learning Uncertainty Quantification Procedures". In: *Entropy* 23.12 (2021).
- [KSC98] S. Kim, N. Shephard, and S. Chib. "Stochastic Volatility: Likelihood Inference and Comparison with ARCH Models". In: *Review of Economic Studies* 65.3 (1998), pp. 361–393.
- [KSH12] A. Krizhevsky, I. Sutskever, and G. Hinton. "Imagenet classification with deep convolutional neural networks". In: *NIPS*. 2012.
- [KSL21] S. Kim, Q. Song, and F. Liang. "Stochastic gradient Langevin dynamics with adaptive drifts". In: *J. Stat. Comput. Simul.* (July 2021), pp. 1–19.
- [KSN99] N. L. Kleinman, J. C. Spall, and D. Q. Naiman. "Simulation-Based Optimization with Stochastic Approximation Using Common Random Numbers". In: *Manage. Sci.* 45.11 (1999), pp. 1570–1578.
- [KSS17] R. G. Krishnan, U. Shalit, and D. Sontag. "Structured Inference Networks for Nonlinear State Space Models". In: *AAAI*. 2017.
- [KT11a] A. Kulesza and B. Taskar. "k-DPPs: Fixed-size determinantal point processes". In: *ICML*. 2011.
- [KT11b] A. Kulesza and B. Taskar. "Learning Determinantal Point Processes". In: *UAI*. 2011.
- [KT+12] A. Kulesza, B. Taskar, et al. "Determinantal Point Processes for Machine Learning". In: *Foundations and Trends in Machine Learning* 5.2–3 (2012), pp. 123–286.
- [KTB11] D. P. Kroese, T. Taimre, and Z. I. Botev. *Handbook of Monte Carlo Methods*. en. 1 edition. Wiley, Mar. 2011.
- [KTX20] R. Kohavi, D. Tang, and Y. Xu. *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*. en. 1st ed. Cambridge University Press, Apr. 2020.
- [KU21] S. Khan and J. Ugander. *Adaptive normalization for IPW estimation*. 2021. arXiv: 2106.07695 [stat.ME].
- [Kua+09] P. Kuan, G. Pan, J. A. Thomson, R. Stewart, and S. Keles. *A hierarchical semi-Markov model for detecting enrichment with application to ChIP-Seq experiments*. Tech. rep. U. Wisconsin, 2009.
- [Kub04] M. Kubale. *Graph colorings*. Vol. 352. American Mathematical Society, 2004.
- [Kuc+16] A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei. "Automatic Differentiation Variational Inference". In: *JMLR* (2016).
- [Kuh55] H. W. Kuhn. "The Hungarian method for the assignment problem". In: *Naval Research Logistics Quarterly* 2 (1955), pp. 83–97.
- [Kuh13] T. Kulesza, S. Stumpf, M. Burnett, S. Yang, I. Kwan, and W.-K. Wong. "Too much, too little, or just right? Ways explanations impact end users' mental models". In: *2013 IEEE Symposium on visual languages and human centric computing*. IEEE. 2013, pp. 3–10.

- 1
- 2 [Kul+19] M. Kull, M. Perello-Nieto, M. Kängsepp, T. S. Filho, H. Song, and P. Flach. "Beyond temperature scaling: Obtaining well-calibrated multiclass probabilities with Dirichlet calibration". In: *NIPS*. 2019.
- 3
- 4 [Kum+19a] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine. "Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction". In: *NeurIPS*. 2019, pp. 11761–11771.
- 5
- 6 [Kum+19b] M. Kumar, M. Babaeizadeh, D. Erhan, C. Finn, S. Levine, L. Dinh, and D. P. Kingma. "VideoFlow: A flow-based generative model for video". In: *ICML Workshop on Invertible Neural Networks and Normalizing Flows* (2019).
- 7
- 8 [Kum+19c] R. Kumar, S. Ozair, A. Goyal, A. Courville, and Y. Bengio. "Maximum entropy generators for energy-based models". In: *arXiv preprint arXiv:1901.08508* (2019).
- 9 [Kün+19] S. R. Künelz, J. S. Sekhon, P. J. Bickel, and B. Yu. "Metalearners for estimating heterogeneous treatment effects using machine learning". In: *Proceedings of the National Academy of Sciences* 116.10 (2019), pp. 4156–4165. eprint: <https://www.pnas.org/content/116/10/4156.full.pdf>.
- 10
- 11 [Kur+19] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel. "Model-Ensemble Trust-Region Policy Optimization". In: *ICLR*. 2019.
- 12
- 13 [Kur+20] R. Kurle, B. Cseke, A. Klushyn, P. van der Smagt, and S. Günnemann. "Continual Learning with Bayesian Neural Networks for Non-Stationary Data". In: *ICLR*. 2020.
- 14
- 15 [Kus+18] M. J. Kusner, J. R. Loftus, C. Russell, and R. Silva. *Counterfactual Fairness*. 2018. arXiv: [1703.06856 \[stat.ML\]](1703.06856).
- 16 [Kus64] H. J. Kushner. "A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise". In: *J. Basic Eng* 86.1 (Mar 1964), pp. 97–106.
- 17 [VKV10] A. Klami, S. Virtanen, and S. Kaski. "Bayesian exponential family projections for coupled data sources". In: *UAI*. 2010.
- 18
- 19 [KW14] D. P. Kingma and M. Welling. "Auto-encoding variational Bayes". In: *ICLR*. 2014.
- 20 [KW18] A. S. I. Kim and M. P. Wand. "On expectation propagation for generalised, linear and mixed models". In: *Aust. N. Z. J. Stat.* 60.1 (Mar 2018), pp. 75–102.
- 21
- 22 [KW19a] D. P. Kingma and M. Welling. "An Introduction to Variational Autoencoders". In: *Foundations and Trends in Machine Learning* 12.4 (2019), pp. 307–392.
- 23 [KW19b] M. J. Kochenderfer and T. A. Wheeler. *Algorithms for Optimization*. The MIT Press, Mar. 2019.
- 24 [KW70] G. S. Kimeldorf and G. Wahba. "A Correspondence Between Bayesian Estimation on Stochastic Processes and Smoothing by Splines". In: *Ann. Math. Stat.* 41.2 (Apr 1970), pp. 495–502.
- 25
- 26 [KW96] R. E. Kass and L. Wasserman. "The Selection of Prior Distributions by Formal Rules". In: *JASA* 91.435 (1996), pp. 1343–1370.
- 27
- 28 [KWW06] K. Kurihara, M. Welling, and N. Vlassis. "Accelerated variational DP mixture models". In: *NIPS*. 2006.
- 29 [KWW22] M. J. Kochenderfer, T. A. Wheeler, and K. Wray. *Algorithms for Decision Making*. The MIT Press, 2022.
- 30
- 31 [KY94] J. J. Kosowsky and A. L. Yuille. "The invisible hand algorithm: Solving the assignment problem with statistical physics". In: *Neural networks* 7.3 (1994), pp. 477–490.
- 32 [Kyn+19] T. Kykkäniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila. "Improved Precision and Recall Metric for Assessing Generative Models". In: *NeurIPS*. 2019.
- 33
- 34 [KZ02] V. Kolmogorov and R. Zabih. "What energy functions can be minimized via graph cuts?". In: *Computer Vision—ECCV 2002* (2002), pp. 183–208.
- 35 [LA87] P. J. M. Laarhoven and E. H. L. Aarts, eds. *Simulated Annealing: Theory and Applications*. Norwell, MA, USA: Kluwer Academic Publishers, 1987.
- 36
- 37 [Lab18] R. Labbe. *Kalman and Bayesian Filters in Python*. 2018.
- 38 [Lad96] F. Lad. *Operational Subjective Statistical Methods: A Mathematical, Philosophical, and Historical Introduction*. en. 1st ed. Wiley-Interscience, Sept. 1996.
- 39
- 40 [Lag+19] I. Lage, E. Chen, J. He, M. Narayanan, B. Kim, S. J. Gershman, and F. Doshi-Velez. "Human evaluation of models built for interpretability". In: *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*. Vol. 7. 1. 2019, pp. 59–67.
- 41
- 42 [Lak+17] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. "Building Machines That Learn and Think Like People". en. In: *Behav. Brain Sci.* (2017), pp. 1–101.
- 43
- 44 [Lal+21] A. Lal, M. W. Lockhart, Y. Xu, and Z. Zu. "How Much Should We Trust Instrumental Variable Estimates in Political Science? Practical Advice based on Over 60 Replicated Studies". In: (2021).
- 45
- 46
- 47
- [Lam92] D. Lambert. "Zero-Inflated Poisson Regression, with an Application to Defects in Manufacturing". In: *Technometrics* 34.1 (1992), pp. 1–14.
- [Lan+12] H. Langseth, T. D. Nielsen, R. Rumí', and A. Salmerón. "Mixtures of truncated basis functions". In: *Int. J. Approx. Reason.* 53.2 (Feb. 2012), pp. 212–227.
- [Lao+20] J. Lao, C. Suter, I. Langmore, C. Chimisov, A. Saxena, P. Sountsov, D. Moore, R. A. Saurous, M. D. Hoffman, and J. V. Dillon. "tfp.mcmc: Modern Markov Chain Monte Carlo Tools Built for Modern Hardware". In: *PROBPROG*. 2020.
- [Lar+16] A. B. L. Larsen, S. K. Sonderby, H. Larochelle, and O. Winther. "Autoencoding beyond pixels using a learned similarity metric". In: *International conference on machine learning*. PMLR. 2016, pp. 1558–1566.
- [Las08] K. B. Laskey. "MEBN: A language for first-order Bayesian knowledge bases". In: *Artif. Intell.* 172.2 (Feb. 2008), pp. 140–178.
- [Lau92] S. L. Lauritzen. "Propagation of probabilities, means and variances in mixed graphical association models". In: *JASA* 87.420 (1992), pp. 1098–1108.
- [Lau95] S. L. Lauritzen. "The EM algorithm for graphical association models with missing data". In: *Computational Statistics and Data Analysis* 19 (1995), pp. 191–201.
- [Lau96] S. Lauritzen. *Graphical Models*. OUP, 1996.
- [Law05] N. D. Lawrence. "Probabilistic non-linear principal component analysis with Gaussian process latent variable models". In: *JMLR* 6 (2005), pp. 1783–1816.
- [Law19] N. Lawrence. *Deep Gaussian Processes (Machine learning summer school tutorial)*. 2019.
- [LB09] H. Lin and J. A. Bilmes. "How to Select a Good Training-data Subset for Transcription: Submodular Active Selection for Sequences". In: *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Brighton, UK, 2009.
- [LB10a] H. Lin and J. Bilmes. "Multi-document Summarization via Budgeted Maximization of Submodular Functions". In: *North American chapter of the Association for Computational Linguistics/Human Language Technology Conference (NAACL-HLT-2010)*. Los Angeles, CA, 2010.
- [LB10b] H. Lin and J. A. Bilmes. "An Application of the Submodular Principal Partition to Training Data Subset Selection". In: *Neural Information Processing Society (NeurIPS, formerly NIPS) Workshop: NeurIPS (formerly NIPS) Workshop on Discrete Optimization in Machine Learning: Submodularity, Sparsity & Polyhedra (DISCML)*. Vancouver, Canada, 2010.
- [LB11] H. Lin and J. Bilmes. "A Class of Submodular Functions for Document Summarization". In: *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT-2011)*. (long paper). Portland, OR, 2011.
- [LB12] H. Lin and J. Bilmes. "Learning Mixtures of Submodular Shells with Application to Document Summarization". In: *Uncertainty in Artificial Intelligence (UAI)*, 2012.
- [LB19] A. Levraty and V. Belle. "Learning Tractable Probabilistic Models in Open Worlds". In: (Jan. 2019). arXiv: [1901.05847 \[cs.LG\]](1901.05847).
- [LBB20] Y. Liu, P.-L. Bacon, and E. Brunskill. "Understanding the Curse of Horizon in Off-Policy Evaluation via Conditional Importance Sampling". In: *ICML*. 2020.
- [LBL16] H. Lakkaraju, S. H. Bach, and J. Leskovec. "Interpretable decision sets: A joint framework for description and prediction". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1675–1684.
- [LBL17] C. Lakshminarayanan, S. Bhatnagar, and C. Szepesvári. "A Linearly Relaxed Approximate Linear Program for Markov Decision Processes". In: *IEEE Transactions on Automatic Control* 63.4 (2017), pp. 1185–1191.
- [LBW17] T. A. Le, A. G. Baydin, and F. Wood. "Inference Compilation and Universal Probabilistic Programming". In: *AISTATS*. 2017.
- [LC02] J. Langford and R. Caruana. "(Not) bounding the true error". In: *NeurIPS*. 2002.
- [LCG12] Y. Lou, R. Caruana, and J. Gehrke. "Intelligible models for classification and regression". In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2012, pp. 150–158.
- [LCR21] T. Lesort, M. Caccia, and I. Rish. "Understanding Continuous Learning Settings with Data Distribution Drift Analysis". In: (Apr. 2021). arXiv: [2104.01678 \[cs.LG\]](2104.01678).
- [LDZ11] Y. Li, H. Duan, and C. X. Zhai. "Cloudspeller: Spelling correction for search queries by using a unified hidden markov model with web-scale resources". In: *SIGIR*. 2011.
- [LDZ12] Y. Li, H. Duan, and C. Zhai. "A Generalized Hidden Markov Model with Discriminative Training for Query Spelling Correction". In: *SIGIR*. 2012, pp. 611–620.

- 1
- [Le+18] T. A. Le, M. Igl, T. Rainforth, T. Jin, and F. Wood. "Auto-Encoding Sequential Monte Carlo". In: *ICLR*. 2018.
- 2
- [L'E18] P. L'Ecuyer. "Randomized Quasi-Monte Carlo: An Introduction for Practitioners". In: *Monte Carlo and Quasi-Monte Carlo Methods*. Springer International Publishing, 2018, pp. 1–19.
- 3
- [Le+19] T. A. Le, A. R. Kosirok, N. Siddharth, Y. W. Teh, and F. Wood. "Revisiting Reweighted Wake-Sleep for Models with Stochastic Control Flow". In: *UAI*. 2019.
- 4
- [LeC+98] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. "Efficient BackProp". en. In: *Neural Networks: Tricks of the Trade*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 1998, pp. 9–50.
- 5
- [Lee06] J. de Leeuw. "Principal Component Analysis of Binary Data by Iterated Singular Value Decomposition". In: *Comput. Stat. Data Anal.* 50.1 (Jan. 2006), pp. 21–39.
- 6
- [Lee+10] J. Lee, V. Mirrokni, V. Nagarajan, and M. Sviridenko. "Maximizing Nonmonotone Submodular Functions under Matroid or Knapsack Constraints". In: *SIAM Journal on Discrete Mathematics* 23.4 (2010), pp. 2053–2078.
- 7
- [Lee+18] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. "Deep Neural Networks as Gaussian Processes". In: *ICLR*. 2018.
- 8
- [Lei+18] J. Lei, M. G'Sell, A. Rinaldo, R. J. Tibshirani, and L. Wasserman. "Distribution-Free Predictive Inference For Regression". In: *JASA* (2018).
- 9
- [Lei+20] F. Leibfried, V. Dutildoir, S. T. John, and N. Durande. "A Tutorial on Sparse Gaussian Processes and Variational Inference". In: (Dec. 2020). arXiv: 2012.13962 [cs.LG].
- 10
- [Lem09] C. Lemieux. *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer, New York, NY, 2009.
- 11
- [Léo14] C. Léonard. "A survey of the Schrödinger problem and some of its connections with optimal transport". In: *Discrete & Continuous Dynamical Systems* 34.4 (2014), p. 1533.
- 12
- [Let+15a] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan. "Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model". In: *The Annals of Applied Statistics* 9.3 (2015).
- 13
- [Let+15b] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan. "Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model". In: *The Annals of Applied Statistics* 9.3 (2015), pp. 1350–1371.
- 14
- [Lev18] S. Levine. "Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review". In: (May 2018). arXiv: 1805.0909 [cs.LG].
- 15
- [Lev+20] S. Levine, A. Kumar, G. Tucker, and J. Fu. *Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems*. arXiv:2005.01643. 2020.
- 16
- [LF+21] L. Le Folgoc, V. Baltatzis, S. Desai, A. Devaraj, S. Ellis, O. E. Martinez Manzanera, A. Nair, H. Qiu, J. Schnabel, and B. Glocker. "Is MC Dropout Bayesian?" In: (Oct. 2021). arXiv: 2110.04286 [cs.LG].
- 17
- [LFG21] F. Lin, X. Fang, and Z. Gao. "Distributionally Robust Optimization: A review on theory and applications". In: *Numer. Algebra Control Optim.* 12.1 (Nov. 2021), pp. 159–212.
- 18
- [LGMT11] F. Le Gland, V. Monbet, and V.-D. Tran. "Large Sample Asymptotics for the Ensemble Kalman Filter". In: *Oxford Handbook of Nonlinear Filtering*. Ed. by D Crisan And. 2011.
- 19
- [LHLT15] Y. Li, J. M. Hernandez-Lobato, and R. E. Turner. "Stochastic Expectation Propagation". In: *NIPS*. 2015.
- 20
- [LHR20] A. Lucic, H. Haned, and M. de Rijke. "Why does my model fail? contrastive local explanations for retail forecasting". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 2020, pp. 90–98.
- 21
- [Li+10] L. Li, W. Chu, J. Langford, and R. E. Schapire. "A contextual-bandit approach to personalized news article recommendation". In: *WWW*. 2010.
- 22
- [Li+16] C. Li, C. Chen, D. Carlson, and L. Carin. "Preconditioned Stochastic Gradient Langevin Dynamics for Deep Neural Networks". In: *AAAI*. 2016.
- 23
- [Li+17a] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Poczos. "Mmd gan: Towards deeper understanding of moment matching network". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2203–2213.
- 24
- [Li+17b] L. Li, K. Jamieson, G. De Salvo, A. Rostamizadeh, and A. Talwalkar. "Hyperband: bandit-based configuration evaluation for hyperparameter optimization". In: *ICLR*. 2017.
- 25
- [Li+17c] O. Li, H. Liu, C. Chen, and C. Rudin. "Deep Learning for Case-Based Reasoning through Prototypes: A Neural Network that Explains Its Predictions". 2017. arXiv: 1710.04806 [cs.AI].
- 26
- [Li+17d] T.-C. Li, J.-Y. Su, W. Liu, and J. M. Corchado. "Approximate Gaussian conjugacy: parametric recursive filtering under nonlinearity, multimodality, uncertainty, and constraint, and beyond". In: *Frontiers of Information Technology & Electronic Engineering* 18.12 (Dec. 2017), pp. 1913–1939.
- 27
- [Li+18] X. Li, C. Li, J. Chi, J. Ouyang, and W. Wang. "Black-box Expectation Propagation for Bayesian Models". In: *ICDM*. Proceedings. Society for Industrial and Applied Mathematics, May 2018, pp. 603–611.
- 28
- [Li+18] Y. Li. "Deep Reinforcement Learning". In: (Oct. 2018). arXiv: 1810.06339 [cs.LG].
- 29
- [Li+19] J. Li, S. Qu, X. Li, J. Szurley, J. Z. Kolter, and F. Metze. "Adversarial Music: Real world Audio Adversary against Wake-word Detection System". In: *NIPS*. Curran Associates, Inc., 2019, pp. 11908–11918.
- 30
- [Li+20a] C. Li, X. Gao, Y. Li, B. Peng, X. Li, Y. Zhang, and J. Gao. "Optimus: Organizing Sentences via Pre-trained Modeling of a Latent Space". In: *EMNLP*. Apr. 2020.
- 31
- [Li+20b] R. Li, S. Pei, B. Chen, Y. Song, T. Zhang, W. Yang, and J. Shaman. "Substantial undocumented infection facilitates the rapid dissemination of novel coronavirus (SARS-CoV2)". en. In: *Science* (Mar. 2020).
- 32
- [Lia+07] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. "Learning and Inferring Transportation Routines". In: *Artificial Intelligence* 171.5 (2007), pp. 311–331.
- 33
- [Lia+08] F. Liang, R. Paulo, G. Molina, M. Clyde, and J. Berger. "Mixtures of g-priors for Bayesian Variable Selection". In: *JASA* 103.481 (2008), pp. 410–423.
- 34
- [Lia+19] V. Liao, R. Ballamy, M. Muller, and H. Candello. "Human-AI Collaboration: Towards Socially-Guided Machine Learning". In: *CHI Workshop on Human-Centered Machine Learning Perspectives*. 2019.
- 35
- [Lil+16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. "Continuous control with deep reinforcement learning". In: *ICLR*. 2016.
- 36
- [Lin13a] W. Lin. "Agnostic notes on regression adjustments to experimental data: Reexamining Freedman's critique". In: *The Annals of Applied Statistics* 7.1 (2013), pp. 295–318.
- 37
- [Lin13b] D. A. Linzer. "Dynamic Bayesian Forecasting of Presidential Elections in the States". In: *JASA* 108.501 (Mar. 2013), pp. 124–134.
- 38
- [Lin+17] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun. "Adversarial ranking for language generation". In: *Advances in Neural Information Processing Systems*. 2017, pp. 3155–3165.
- 39
- [Lin+20] H. Lin, H. Chen, T. Zhang, C. Laroche, and K. Choromanski. "Demystifying Orthogonal Monte Carlo and Beyond". In: (May 2020). arXiv: 2005.13590 [cs.LG].
- 40
- [Lin+21] T. Lin, Y. Wang, X. Liu, and X. Qiu. "A Survey of Transformers". In: (June 2021). arXiv: 2106.04554 [cs.LG].
- 41
- [Lin88a] B. Lindsay. "Composite Likelihood Methods". In: *Contemporary Mathematics* 80.1 (1988), pp. 221–239.
- 42
- [Lin88b] R. Linsker. "Self-organization in a perceptual network". In: *Computer* 21.3 (Mar. 1988), pp. 105–117.
- 43
- [Lin92] L.-J. Lin. "Self-Improving Reactive Agents Based on Reinforcement Learning, Planning, and Teaching". In: *Mach. Learn.* 8.3–4 (May 1992), pp. 293–321.
- 44
- [Lip18] Z. C. Lipton. "The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery." In: *Queue* 16.3 (2018), pp. 31–57.
- 45
- [Liu01] J. Liu. *Monte Carlo Strategies in Scientific Computation*. Springer, 2001.
- 46
- [Liu+15] Z. Liu, P. Luo, X. Wang, and X. Tang. "Deep Learning Face Attributes in the Wild". In: *ICCV*. 2015.
- 47
- [Liu+18a] L. Liu, X. Liu, C.-J. Hsieh, and D. Tao. "Stochastic Second-order Methods for Non-convex Optimization with Inexact Hessian and Gradient". In: (Sept. 2018). arXiv: 1809.09853 [stat.ML].
- 48
- [Liu+18b] Q. Liu, L. Li, Z. Tang, and D. Zhou. "Breaking the Curse of Horizon: Infinite-horizon Off-policy Estimation". In: *NeurIPS*. Curran Associates Inc., 2018, pp. 5361–5371.
- 49
- [Liu+19a] H. Liu, Y.-S. Ong, Z. Yu, J. Cai, and X. Shen. "Scalable Gaussian Process Classification with Additive Noise for Various Likelihoods". In: (Sept. 2019). arXiv: 1909.06541 [stat.ML].
- 50
- [Liu+19b] R. Liu, J. Regier, N. Tripuraneni, M. I. Jordan, and J. McAuliffe. "Rao-Blackwellized Stochastic Gradients for Discrete Distributions". In: *ICML*. 2019.
- 51
- [Liu+20a] F. Liu, W. Xu, J. Lu, G. Zhang, A. Gretton, and D. J. Sutherland. "Learning Deep Kernels for Non-Parametric Two-Sample Tests". In: *ICML*. Feb. 2020.
- 52
- [Liu+20b] F. Liu, W. Xu, J. Lu, G. Zhang, A. Gretton, and D. J. Sutherland. "Learning deep kernels for non-parametric two-sample tests". In: *International Conference on Machine Learning*. PMLR, 2020, pp. 6316–6326.
- 53
- [Liu+20c] H. Liu, Y.-S. Ong, X. Shen, and J. Cai. "When Gaussian Process Meets Big Data: A Review of Scalable GPs". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.1 (2020).

- 1
- 2 [Liu+20d] J. Z. Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax-  
Weiss, and B. Lakshminarayanan. "Simple and Principled Un-  
certainty Estimation with Deterministic Deep Learning via Dis-  
tance Awareness". In: *NIPS*. 2020.
- 3 [Liu+21a] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and  
G. Neubig. "Pre-train, Prompt, and Predict: A Systematic Sur-  
vey of Prompting Methods in Natural Language Processing". In:  
(July 2021). arXiv: [2107.13586 \[cs.CL\]](https://arxiv.org/abs/2107.13586).
- 4 [Liu+21b] W. Liu, X. Wang, J. D. Owens, and Y. Li. "Energy-  
based Out-of-distribution Detection". In: *NIPS*. 2021.
- 5 [Liu+22] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Dar-  
rell, and S. Xie. "A ConvNet for the 2020s". In: (Jan. 2022).  
arXiv: [2201.03545 \[cs.CV\]](https://arxiv.org/abs/2201.03545).
- 6 [LJ08] P. Liang and M. I. Jordan. "An Asymptotic Analysis of  
Generative, Discriminative, and Pseudolikelihood Estimators".  
In: *International Conference on Machine Learning (ICML)*.  
2008.
- 7 [LJS14] F. Lindsten, M. I. Jordan, and T. B. Schön. "Par-  
ticle Gibbs with Ancestor Sampling". In: *JMLR* 15.63 (2014),  
pp. 2145–2184.
- 8 [Lju87] L. Ljung. *System Identification: Theory for the User*.  
Prentice Hall, 1987.
- 9 [LK07] P. Liang and D. Klein. *Structured Bayesian Nonpara-  
metric Models with Variational Inference*. ACL Tutorial. 2007.
- 10 [LK09] P. Liang and D. Klein. "Online EM for Unsupervised  
Models". In: *NAACL*. 2009.
- 11 [LJK09] D. Lewandowski, D. Kurowicka, and H. Joe. "Gen-  
erating random correlation matrices based on vines and ex-  
tended onion method". In: *J. Multivar. Anal.* 100.9 (Oct. 2009),  
pp. 1989–2001.
- 12 [LL17] S. M. Lundberg and S.-I. Lee. "A unified approach to in-  
terpreting model predictions". In: *NIPS*. 2017, pp. 4765–4774.
- 13 [LLC20] M. Locher, K. B. Laskey, and P. C. G. Costa. "Design  
patterns for modeling first-order expressive Bayesian networks".  
In: *Knowl. Eng. Rev.* 35 (2020).
- 14 [LLJ16] Q. Liu, J. Lee, and M. Jordan. "A kernelized Stein dis-  
crepancy for goodness-of-fit tests". In: *International conference  
on machine learning*. 2016, pp. 276–284.
- 15 [LLN06] B. Lehmann, D. Lehmann, and N. Nisan. "Combinato-  
rial auctions with decreasing marginal utilities". In: *Games and  
Economic Behavior* 55.2 (2006), pp. 270–296.
- 16 [Llo+14] J. R. Lloyd, D. Duvenaud, R. Grosse, J. B. Tenenbaum,  
and Z. Ghahramani. "Automatic Construction and Natural-  
Language Description of Nonparametric Regression Models".  
In: *AAAI*. 2014.
- 17 [LLT89] K. Lange, R. Little, and J. Taylor. "Robust Statistical  
Modeling Using the T Distribution". In: *JASA* 84.408 (1989),  
pp. 881–896.
- 18 [LM11] H. Larochelle and I. Murray. "The neural autoregressive  
distribution estimator". In: *AISTATS*. Vol. 15. 2011, pp. 29–37.
- 19 [LM20] M. L. Leavitt and A. Morcos. "Towards falsifiable in-  
terpretability research". In: *arXiv preprint arXiv:2010.12016*  
(2020).
- 20 [LMP01] J. Lafferty, A. McCallum, and F. Pereira. "Conditional  
Random Fields: Probabilistic Models for Segmenting and Label-  
ing Sequence Data". In: *ICML*. 2001.
- 21 [LMR15] F. Lavancier, J. Møller, and E. Rubak. "Determinant-  
al point process models and statistical inference". In: *Journal  
of the Royal Statistical Society: Series B (Statistical Method-  
ology)* 77.4 (2015), pp. 853–877.
- 22 [LM16] B. Leimkuhler, C. Matthews, and G. Stoltz. "The  
computation of averages from equilibrium and nonequilibrium  
Langvin molecular dynamics". In: *IMA J. Numer. Anal.* 36.1  
(Jan. 2016), pp. 13–79.
- 23 [LN01] S. Lauritzen and D. Nilsson. "Representing and solving  
decision problems with limited information". In: *Management  
Science* 47 (2001), pp. 1238–1251.
- 24 [LN19] H. Lin and V. Ng. "Abstractive summarization: A survey  
of the state of the art". In: *Proceedings of the AAAI Conference  
on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 9815–9822.
- 25 [LN96] o. Lee and J. A. Nelder. "Hierarchical Generalized Lin-  
ear Models". In: *J. of Royal Stat. Soc. Series B* 58.4 (1996),  
pp. 619–678.
- 26 [Loc+18] F. Locatello, S. Bauer, M. Lucic, G. Rätsch, S. Gelly,  
B. Schölkopf, and O. Bachem. "Challenging Common Assump-  
tions in the Unsupervised Learning of Disentangled Repre-  
sentations". In: (Nov. 2018). arXiv: [1811.12359 \[cs.LG\]](https://arxiv.org/abs/1811.12359).
- 27 [Lod+02] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristian-  
ini, and C. Watkins. "Text classification using string kernels".  
In: *J. Mach. Learn. Res.* (Mar. 2002).
- 28 [Loe04] H Loeliger. "An introduction to factor graphs". In:  
*IEEE Signal Process. Magazine* 21.1 (Jan. 2004), pp. 28–41.
- 29 [Loe+07] H Loeliger, J Dauwels, J Hu, S Korl, L Ping, and F. R.  
Kschischang. "The Factor Graph Approach to Model-Based Sig-  
nal Processing". In: *Proc. IEEE* 95.6 (June 2007), pp. 1295–  
1322.
- 30 [Lon+18] M. Long, Z. Cao, J. Wang, and M. I. Jordan. "Condi-  
tional adversarial domain adaptation". In: *Neural Information  
Processing Systems* (2018).
- 31 [Lov+20] T. Lovett, M. Briers, M. Charalambides, R. Jersakova,  
J. Lomax, and C. Holmes. "Inferring proximity from Blue-  
tooth Low Energy RSSI with Unscented Kalman Smoothers".  
In: (July 2020). arXiv: [2007.05057 \[eess.SP\]](https://arxiv.org/abs/2007.05057).
- 32 [Lov83] L. Lovász. "Submodular functions and convexity". In:  
*Mathematical programming: the state of the art*. Springer, 1983,  
pp. 235–257.
- 33 [LP01] U. Lerner and R. Parr. "Inference in Hybrid Networks:  
Theoretical Limits and Practical Algorithms". In: *UAI*. 2001.
- 34 [LP03] M. G. Lagoudakis and R. Parr. "Least-Squares Policy  
Iteration". In: *JMLR* 4 (2003), pp. 1107–1149.
- 35 [LP06] N. Lartillot and H. Philippe. "Computing Bayes factors  
using thermodynamic integration". en. In: *Systematic Biology*  
55.2 (Apr. 2006), pp. 195–207.
- 36 [LPB17] B. Lakshminarayanan, A. Pritzel, and C. Blundell.  
"Simple and Scalable Predictive Uncertainty Estimation using  
Deep Ensembles". In: *NIPS*. 2017.
- 37 [LPO17] D. Lopez-Paz and M. Oquab. "Revisiting classifier two-  
sample tests". In: *International Conference on Learning Rep-  
resentations*. 2017.
- 38 [LPR17] D. Lopez-Paz and M. Ranzato. "Gradient Episodic  
Memory for Continual Learning". In: *NIPS*. June 2017.
- 39 [LR18] T. Liang and A. Rakitin. "Just Interpolate: Kernel  
"Ridgeless" Regression Can Generalize". In: (Aug. 2018). arXiv:  
[1808.00387 \[math.ST\]](https://arxiv.org/abs/1808.00387).
- 40 [LR85] T. L. Lai and H. Robbins. "Asymptotically efficient  
adaptive allocation rules". en. In: *Adv. Appl. Math.* (Mar.  
1985).
- 41 [LR87] R. J. Little and D. B. Rubin. *Statistical Analysis with  
Missing Data*. New York: Wiley and Son, 1987.
- 42 [LR95] C. Liu and D. Rubin. "ML Estimation of the T distribu-  
tion using EM and its extensions, ECM and ECME". In: *Sta-  
tistica Sinica* 5 (1995), pp. 19–39.
- 43 [LRC19] Y. Li, B. I. P. Rubinstein, and T. Cohn. "Truth In-  
ference at Scale: A Bayesian Model for Adjudicating Highly  
Redundant Crowd Annotations". In: *WWW*. Feb. 2019.
- 44 [LS01] D. Lee and S. Seung. "Algorithms for non-negative ma-  
trix factorization". In: *NIPS*. 2001.
- 45 [LS19] T. Lattimore and C. Szepesvari. *Bandit Algorithms*.  
Cambridge, 2019.
- 46 [LS79] P. W. Lewis and G. S. Shedler. "Simulation of nonho-  
mogeneous Poisson processes by thinning". In: *Naval research  
logistics quarterly* 26.3 (1979), pp. 403–413.
- 47 [LS88] S. L. Lauritzen and D. J. Spiegelhalter. "Local computa-  
tions with probabilities on graphical structures and their appli-  
cations to expert systems". In: *J. of Royal Stat. Soc. Series B*  
B 50 (1988), pp. 127–224.
- 48 [LS98] V. Lepar and P. P. Shenoy. "A Comparison of Lauritzen-  
Spiegelhalter, Hugin and Shenoy-Shafer Architectures for Com-  
puting Marginals of Probability Distributions". In: *UAI*. Ed. by  
G. Cooper and S. Moral. Morgan Kaufmann, 1998, pp. 328–337.
- 49 [LS99] D. D. Lee and H. S. Seung. "Learning the parts of objects  
by non-negative matrix factorization". In: *Nature* 401.6755  
(1999), pp. 788–791.
- 50 [LST21] N. Loo, S. Swaroop, and R. E. Turner. "Generalized  
Variational Continual Learning". In: *ICLR*. 2021.
- 51 [LST90] J. K. Lenstra, D. B. Shmoys, and É. Tardos. "Approxima-  
tion algorithms for scheduling unrelated parallel machines".  
In: *Mathematical programming*. 1990.
- 52 [LSV09] J. Lee, M. Sviridenko, and J. Vondrák. "Submodular  
maximization over multiple matroids via generalized exchange  
properties". In: *Approximation, Randomization, and Com-  
binatorial Optimization. Algorithms and Techniques* (2009),  
pp. 244–257.
- 53 [LSW15] Y. T. Lee, A. Sidford, and S. C.-w. Wong. "A faster cut-  
ting plane method and its implications for combinatorial and  
convex optimization". In: *2015 IEEE 56th Annual Symposium  
on Foundations of Computer Science*. IEEE. 2015, pp. 1049–  
1065.
- 54 [LSZ15] Y. Li, K. Swersky, and R. Zemel. "Generative Moment  
Matching Networks". In: *ICML*. 2015.
- 55 [LT16] M.-Y. Liu and O. Tuzel. "Coupled Generative Adversar-  
ial Networks". In: *NIPS*. 2016, pp. 469–477.
- 56 [LTW15] Q. Li, C. Tai, and E. Weinan. "Stochastic modified  
equations and adaptive stochastic gradient algorithms". In:  
*ICML*. Nov. 2015.
- 57 [Lu+20] J. Lu, P. Gong, J. Ye, and C. Zhang. "Learning from  
Very Few Samples: A Survey". In: (Sept. 2020). arXiv:  
[2009.02653 \[cs.LG\]](https://arxiv.org/abs/2009.02653).

- [Lu+21] X. Lu, I. Osband, B. Van Roy, and Z. Wen. “Evaluating Probabilistic Inference in Deep Learning: Beyond Marginal Predictions”. In: (July 2021). arXiv: 2107.09224 [cs.LG].
- [Lu+22] X. Lu, I. Osband, B. Van Roy, and Z. Wen. “From Predictions to Decisions: The Importance of Joint Predictive Distributions”. In: (Feb. 2022). arXiv: 2107.09224 [cs.LG].
- [Luc+18] A. Lucas, M. Iliadis, R. Molina, and A. K. Katsaggelos. “Using Deep Neural Networks for Inverse Problems in Imaging: Beyond Analytical Methods”. In: *IEEE Signal Process. Mag.* 35.1 (Jan. 2018), pp. 20–36.
- [Luc+19] J. Lucas, G. Tucker, R. Grosse, and M. Norouzi. “Don’t blame the ELBO! A linear VAE perspective on posterior collapse”. In: *NIPS*. 2019.
- [Luh58] H. P. Luhn. “The automatic creation of literature abstracts”. In: *IBM Journal of research and development* 2.2 (1958), pp. 159–165.
- [Lun+20] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. “From local explanations to global understanding with explainable AI for trees”. In: *Nature machine intelligence* 2.1 (2020), pp. 56–67.
- [Luo+19] Y. Luo, H. Xu, Y. Li, Y. Tian, T. Darrell, and T. Ma. “Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees”. In: *ICLR*. 2019.
- [Lut16] J. Lutinen. “BayesPy: variational Bayesian inference in Python”. In: *JMLR* (2016).
- [LUW17] C. Louizos, K. Ullrich, and M. Welling. “Bayesian Compression for Deep Learning”. In: *NIPS*. 2017.
- [LV06] F. Liese and I. Vajda. “On divergences and informations in statistics and information theory”. In: *IEEE Transactions on Information Theory* 52.10 (2006), pp. 4394–4412.
- [LW04] H. Lopes and M. West. “Bayesian model assessment in factor analysis”. In: *Statistica Sinica* 14 (2004), pp. 41–67.
- [LW16] C. Louizos and M. Welling. “Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors”. In: *ICML*. 2016.
- [LW17] C. Louizos and M. Welling. “Multiplicative Normalizing Flows for Variational Bayesian Neural Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. 2017, pp. 2218–2227.
- [LWS18] Z. C. Lipton, Y.-X. Wang, and A. Smola. “Detecting and Correcting for Label Shift with Black Box Predictors”. In: *ICML*. 2018.
- [LY17] J. H. Lim and J. C. Ye. “Geometric gan”. In: *arXiv preprint arXiv:1705.02894* (2017).
- [Lyu11] S. Lyu. “Unifying non-maximum likelihood learning objectives with minimum KL contraction”. In: *Advances in Neural Information Processing Systems*. 2011, pp. 64–72.
- [Lyu12] S. Lyu. “Interpretation and generalization of score matching”. In: *arXiv preprint arXiv:1205.2629* (2012).
- [LZ20] B. Lim and S. Zohren. “Time Series Forecasting With Deep Learning: A Survey”. In: (Apr. 2020). arXiv: 2004.13408 [stat.ML].
- [MA10] I. Murray and R. P. Adams. “Slice sampling covariance hyperparameters of latent Gaussian models”. In: *NIPS*. 2010, pp. 1732–1740.
- [Maa+16] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. “Auxiliary Deep Generative Models”. In: *ICML*. 2016.
- [Maa+19] L. Maaløe, M. Fraccaro, V. Levine, and O. Winther. “BIVA: A Very Deep Hierarchical Latent Variables for Generative Modeling”. In: *NIPS*. 2019.
- [Mac03] D. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.
- [Mac+11] J. H. Macke, L. Büsing, J. P. Cunningham, B. M. Y. Ece, K. V. Shenoy, and M. Sahani. “Empirical models of spiking in neural populations”. In: *NIPS*. 2011.
- [Mac+15] D. Maclaurin, D. Duvenaud, M. Johnson, and R. P. Adams. *Autograd: Reverse-mode differentiation of native Python*. 2015.
- [Mac+19] D. Maclaurin, A. Radul, M. J. Johnson, and D. Vytiniotis. “Dex: array programming with typed indices”. In: *NeurIPS workshop: Program Transformations for Machine Learning* (2019).
- [Mac75] O. Macchi. “The coincidence approach to stochastic point processes”. In: *Advances in Applied Probability* 7.1 (1975), pp. 83–122.
- [Mac92a] D. MacKay. “The evidence framework applied to classification networks”. In: *Neural Computation* 4.5 (1992), pp. 720–736.
- [Mac92b] D. J. C. MacKay. “A Practical Bayesian Framework for Backpropagation Networks”. In: *Neural Comput.* 4.3 (May 1992), pp. 448–472.
- [Mac95] D. MacKay. “Probable networks and plausible predictions — a review of practical Bayesian methods for supervised neural networks”. In: *Network: Computation in Neural Systems* 6.3 (1995), pp. 469–505.
- [Mac98] D. MacKay. “Introduction to Gaussian Processes”. In: *Neural Networks and Machine Learning*. Ed. by C. Bishop. 1998.
- [Mac99a] S. N. MacEachern. “Dependent nonparametric processes”. In: *ASA, proceedings of the section on Bayesian statistical science*. Vol. 1. Alexandria, Virginia. Virginia: American Statistical Association; 1999. 1999, pp. 50–55.
- [Mac99b] D. MacKay. “Comparison of approximate methods for handling hyperparameters”. In: *Neural Computation* 11.5 (1999), pp. 1035–1068.
- [Mad+17] C. J. Maddison, D. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. W. Teh. “Filtering Variational Objectives”. In: *NIPS*. 2017.
- [Mad+18] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. “Towards Deep Learning Models Resistant to Adversarial Attacks”. In: *ICLR*. 2018.
- [Mad+19] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson. “A Simple Baseline for Bayesian Uncertainty in Deep Learning”. In: *NIPS*. Curran Associates, Inc., 2019, pp. 13153–13164.
- [MAD20] W. R. Morningstar, A. A. Alemi, and J. V. Dillon. “PAC<sup>m</sup>-Bayes: Narrowing the Empirical Risk Gap in the Misspecified Bayesian Regime”. In: (Oct. 2020). arXiv: 2010.09629 [cs.LG].
- [Mah07] R. P. S. Mahler. *Statistical Multisource-Multitarget Information Fusion*. Norwood, MA, USA: Artech House, Inc., 2007.
- [Mah08] H. Mahmoud. *Polya Urn Models*. en. 1 edition. Chapman and Hall/CRC, June 2008.
- [Mah13] R. Mahler. “Statistics 102 for Multisource-Multitarget Detection and Tracking”. In: *IEEE J. Sel. Top. Signal Process.* 7.3 (June 2013), pp. 376–389.
- [Mai+10] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. “Online learning for matrix factorization and sparse coding”. In: *JMLR* 11 (2010), pp. 19–60.
- [Mai13] J. Mairal. “Stochastic Majorization-minimization Algorithms for Large-scale Optimization”. In: *NIPS*. 2013, pp. 2283–2291.
- [Mai15] J. Mairal. “Incremental Majorization-Minimization Optimization with Application to Large-Scale Machine Learning”. In: *SIAM J. Optim.* 25.2 (Jan. 2015), pp. 829–855.
- [Mai+22] Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner. “Online continual learning in image classification: An empirical survey”. In: *Neurocomputing* 469 (Jan. 2022), pp. 28–51.
- [Mak+15a] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. “Adversarial Autoencoders”. In: (Nov. 2015). arXiv: 1511.05644 [cs.LG].
- [Mak+15b] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. “Adversarial autoencoders”. In: *arXiv preprint arXiv:1511.05644* (2015).
- [Mak+20] A. Makkluva, A. Taghvaei, S. Oh, and J. Lee. “Optimal transport mapping via input convex neural networks”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 6672–6681.
- [Mal+17] D. M. Malioutov, K. R. Varshney, A. Emad, and S. Dash. “Learning interpretable classification rules with boolean compressed sensing”. In: *Transparent Data Mining for Big and Small Data*. Springer, 2017, pp. 95–121.
- [Man] B. Mann. *How many times should you shuffle a deck of cards?* Tech. rep. Dartmouth.
- [Man+07] V. Mansinghka, D. Roy, R. Rifkin, and J. Tenenbaum. “AClass: An online algorithm for generative classification”. In: *AISTATS*. 2007.
- [Man+19] D. J. Mankowitz, N. Levine, R. Jeong, Y. Shi, J. Kay, A. Abdolmaleki, J. T. Springenberg, T. Mann, T. Hester, and M. Riedmiller. “Robust Reinforcement Learning for Continuous Control with Model Misspecification”. In: (June 2019). arXiv: 1906.07516 [cs.LG].
- [Man90] C. F. Manski. “Nonparametric Bounds on Treatment Effects”. In: *The American Economic Review* 80.2 (1990), pp. 319–323.
- [Mao+17] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley. “Least Squares Generative Adversarial Networks”. In: *ICCV*. 2017.
- [Mar03] B. Marlin. “Modeling User Rating Profiles for Collaborative Filtering”. In: *NIPS*. 2003.
- [Mar+06] A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, and R. F. abd A. Califano. “ARACNE: An Algorithm for the Reconstruction of Gene Regulatory Networks in a Mammalian Cellular Context”. In: *BMC Bioinformatics* 7 (2006).

- 1
- 2 [Mar08] B. Marlin, "Missing Data Problems in Machine Learning". PhD thesis. U. Toronto, 2008.
- 3 [Mar+10] B. M. Marlin, K. Swersky, B. Chen, and N. de Freitas, "Inductive Principles for Restricted Boltzmann Machine Learning". In: *AISTATS*. 2010.
- 4 [Mar10a] J. Martens, "Deep learning via Hessian-free optimization". In: *ICML*. 2010.
- 5 [Mar10b] J. Martens, "Learning the Linear Dynamical System with ASOS". In: *ICML*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 743–750.
- 6 [Mar16] J. Martens, "Second-order optimization for neural networks". PhD thesis. Toronto, 2016.
- 7 [Mar18] O. Martin, *Bayesian analysis with Python*. Packt, 2018.
- 8 [Mar20] J. Martens, "New insights and perspectives on the natural gradient method". In: *JMLR* (2020).
- 9 [Mar21] J. Marino, "Predictive Coding, Variational Autoencoders, and Biological Connections". en. In: *Neural Comput.* 34.1 (Dec. 2021), pp. 1–44.
- 10 [Mas+20] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer, "Class-incremental learning: survey and performance evaluation on image classification". In: (Oct. 2020). arXiv: 2010.15277 [cs.LG].
- 11 [Mat14] C. Mattfeld, "Implementing spectral methods for hidden Markov models with real-valued emissions". MA thesis. ETH Zurich, Apr. 2014.
- 12 [Mat+16] A. Matthews, J. Hensman, R. Turner, and Z. Ghahramani, "On Sparse Variational Methods and the Kullback-Leibler Divergence between Stochastic Processes". en. In: *AISTATS*. May 2016, pp. 231–239.
- 13 [Mav16] Mavrogiatou, L. and Vyshevskiy, "Sequential Importance Sampling for Online Bayesian Changepoint Detection". In: *22nd International Conference on Computational Statistics*. 2016.
- 14 [MAV17] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational Dropout Sparsifies Deep Neural Networks". In: *ICML*. 2017.
- 15 [May79] P. Maybeck, *Stochastic models, estimation, and control*. Academic Press, 1979.
- 16 [Maz+22] P. Mazzaglia, T. Verbelen, O. Catal, and B. Dhoedt, "The Free Energy Principle for Perception and Action: A Deep Learning Perspective". en. In: *Entropy* 24.2 (Feb. 2022).
- 17 [MAZA18] X. B. P. and Marcin Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization". In: *ICRA*. 2018, pp. 1–8.
- 18 [MB16] Y. Miao and P. Blunsom, "Language as a Latent Variable: Discrete Generative Models for Sentence Compression". In: *EMNLP*. 2016.
- 19 [MB18] A. Mensch and M. Blondel, "Differentiable Dynamic Programming for Structured Prediction and Attention". In: *ICML*. 2018.
- 20 [MB21] M. Y. Michelis and Q. Becker, "On Linear Interpolation in the Latent Space of Deep Generative Models". In: *ICLR Workshop on Geometrical and Topological Representation Learning*. May 2021.
- 21 [MB88] T. Mitchell and J. Beauchamp, "Bayesian Variable Selection in Linear Regression". In: *JASA* 83 (1988), pp. 1023–1036.
- 22 [MBJ06] J. McAuliffe, D. Blei, and M. Jordan, "Nonparametric empirical Bayes for the Dirichlet process mixture model". In: *Statistics and Computing* 16.1 (2006), pp. 5–14.
- 23 [MBJ20] T. M. Moerland, J. Broekens, and C. M. Jonker, "Model-based Reinforcement Learning: A Survey". In: (June 2020). arXiv: 2006.16712 [cs.LG].
- 24 [MBK20] X. Meng, R. Bachmann, and M. E. Khan, "Training Binary Neural Networks using the Bayesian Learning Rule". In: *ICML*. Feb. 2020.
- 25 [MBL20] B. Mirzasoleiman, J. Bilmes, and J. Leskovec, "Core-sets for data-efficient training of machine learning models". In: *International Conference on Machine Learning*. PMLR, 2020, pp. 6950–6960.
- 26 [MBW20] W. J. Maddox, G. Benton, and A. G. Wilson, "Rethinking Parameter Counting in Deep Models: Effective Dimensionality Revisited". In: *arXiv preprint arXiv:2003.02139* (2020).
- 27 [MC19] P. Moreno Comellas, "Vision as inverse graphics for detailed scene understanding". en. PhD thesis. July 2019.
- 28 [McA99] D. A. McAllester, "PAC-Bayesian model averaging". In: *Proceedings of the twelfth annual conference on computational learning theory*. 1999.
- 29 [McC03] A. McCray, "An upper level ontology for the biomedical domain". In: *Comparative and Functional Genomics* 4 (2003), pp. 80–84.
- 30 [McE20] R. McElreath, *Statistical Rethinking: A Bayesian Course with Examples in R and Stan (2nd edition)*. en. Chapman and Hall/CRC, 2020.
- 31 [McG54] W. McGill, "Multivariate information transmission". In: *Psychometrika* 19 (1954), pp. 97–116.
- 32 [McM+13] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, J. Nie, T. Phillips, E. Davydov, D. Golovin, et al., "Ad click prediction: a view from the trenches". In: *KDD*. 2013, pp. 1222–1230.
- 33 [Md+19] C. de Masson d'Autume, M. Rosca, J. Rae, and S. Mohamed, "Training language GANs from Scratch". In: (May 2019). arXiv: 1905.09922 [cs.CL].
- 34 [MD97] X. L. Meng and D. van Dyk, "The EM algorithm — an old folk song sung to a fast new tune (with Discussion)". In: *J. Royal Stat. Soc. B* 59 (1997), pp. 511–567.
- 35 [MDA15] D. MacLaurin, D. Duvenaud, and R. P. Adams, "Gradient-based Hyperparameter Optimization through Reversible Learning". In: *ICML*. 2015.
- 36 [MDM19] S. Mahloujifar, D. I. Diochnos, and M. Mahmoody, "The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 4536–4543.
- 37 [MDR94] S. Muggleton and L. De Raedt, "Inductive logic programming: Theory and methods". In: *The Journal of Logic Programming* 19 (1994), pp. 629–679.
- 38 [ME17] H. Mei and J. M. Eisner, "The neural hawkes process: A neurally self-modulating multivariate point process". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6745–6764.
- 39 [Med+21] M. A. Medina, J. L. M. Olea, C. Rush, and A. Velez, "On the Robustness to Misspecification of  $\alpha$ -Posteriori Their Variational Approximations". In: (Apr. 2021). arXiv: 2104.08324 [stat.ML].
- 40 [Mee+18] J.-W. van de Meent, B. Paige, H. Yang, and F. Wood, "An introduction to probabilistic programming". Foundations and Trends in Machine Learning, 2018.
- 41 [Mei18a] N. Meinshausen, "Causality from a distributional robustness point of view". In: *2018 IEEE Data Science Workshop (DSW)*. June 2018, pp. 6–10.
- 42 [Mei18b] N. Meinshausen, "CAUSALITY FROM A DISTRIBUTIONAL ROBUSTNESS POINT OF VIEW". In: *2018 IEEE Data Science Workshop (DSW)*. 2018, pp. 6–10.
- 43 [Mer] *Definition of interpret*. 2022. URL: <https://www.merriam-webster.com/dictionary/interpret>.
- 44 [Mer+00] R. van der Merwe, A. Doucet, N. de Freitas, and E. Wan, "The Unscented Particle Filter". In: *NIPS-13*. 2000.
- 45 [Met16] C. Metz, *In Two Moves, AlphaGo and Lee Sedol Redefine the Future*. 2016. URL: <https://www.wired.com/2016/03/two-moves-alpha-go-lee-sedol-redefined-future/> (visited on 01/07/2022).
- 46 [Met+16] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled Generative Adversarial Networks". In: (2016).
- 47 [Met+17] L. Metz, J. Ibarz, N. Jaity, and J. Davidson, "Discrete Sequential Prediction of Continuous Actions for Deep RL". In: (May 2017). arXiv: 1705.05035 [cs.LG].
- 48 [Met+53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of state calculations by fast computing machines". In: *J. of Chemical Physics* 21 (1953), pp. 1087–1092.
- 49 [Mey+18] F. Meyer, T. Kropfleiter, J. Williams, R. Lau, F. Hiawatsh, P. Braca, and M. Win, "Message passing algorithms for scalable multitarget tracking". In: *Proc. IEEE* 106.2 (2018).
- 50 [Mey+21] R. A. Meyer, C. Musco, C. Musco, and D. P. Woodruff, "Hutch++: Optimal Stochastic Trace Estimation". In: *SIAM Symposium on Simplicity in Algorithms (SOSA21)*. 2021.
- 51 [Mey22] S. Meyn, *Control Systems and Reinforcement Learning*. Cambridge, 2022.
- 52 [MG15] J. Martens and R. Grosse, "Optimizing Neural Networks with Kronecker-factored Approximate Curvature". In: *ICML*. 2015.
- 53 [MGM06] I. Murray, Z. Ghahramani, and D. J. C. MacKay, "MCMC for doubly-intractable distributions". In: *Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*. AUAI Press, 2006, pp. 359–366.
- 54 [MGN18a] L. Mescheder, A. Geiger, and S. Nowozin, "Which Training Methods for GANs do actually Converge?". In: *ICML*. 2018.
- 55 [MGN18b] L. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for GANs do actually converge?". In: *International conference on machine learning*. PMLR, 2018, pp. 3481–3490.
- 56 [MGR18] H. Mania, A. Guy, and B. Recht, "Simple random search of static linear policies is competitive for reinforcement learning". In: *NIPS*. Ed. by S. Bengio, H. Wallach, H. Larochelle,

- 1 K Grauman, N Cesa-Bianchi, and R Garnett. Curran Associates, Inc., 2018, pp. 1800–1809.
- 2 [MH12] R. Mazumder and T. Hastie. *The Graphical Lasso: New Insights and Alternatives*. Tech. rep. Stanford Dept. of Statistics, 2012.
- 3 [MH14] J. W. Miller and M. T. Harrison. “Inconsistency of Pitman-Yor process mixtures for the number of components”. In: *JMLR* 15.1 (2014), pp. 3333–3370.
- 4 [MH20] I. Mordatch and J. Hamrick. *ICML tutorial on model-based methods in reinforcement learning*. <https://sites.google.com/corp/view/mbrl-tutorial>. 2020.
- 5 [MHB17] S. Mandt, M. D. Hoffman, and D. M. Blei. “Stochastic Gradient Descent As Approximate Bayesian Inference”. In: *JMLR* 18.1 (Jan. 2017), pp. 4873–4907.
- 6 [MHH14] F. Meyer, O. Hlinka, and F. Hlawatsch. “Sigma point belief propagation”. In: *IEEE Signal Processing Letters* 21.2 (2014), pp. 145–149.
- 7 [MHN13] A. L. Maas, A. Y. Hannun, and A. Y. Ng. “Rectifier Nonlinearities Improve Neural Network Acoustic Models”. In: *ICML*, Vol. 28. 2013.
- 8 [MHS03] J. R. Movellan, J. Hershey, and J. Susskind. “Large-scale convolutional HMMs for real-time video tracking”. 2003.
- 9 [Mid+19] L. Middleton, G. Deligiannidis, A. Doucet, and P. E. Jacob. “Unbiased Smoothing using Particle Independent Metropolis-Hastings”. In: *AISTATS*, Vol. 89. Proceedings of Machine Learning Research. PMLR, 2019, pp. 2378–2387.
- 10 [Mik+13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. “Distributed representations of words and phrases and their compositionality”. In: *NIPS*. 2013, pp. 3111–3119.
- 11 [Mil+05] B. Milch, B. Marthi, S. Russell, D. Sontag, D. Ong, and A. Kolobov. “BLOG: Probabilistic Models with Unknown Objects”. In: *IJCAI*. 2005.
- 12 [Mil19] T. Miller. “Explanation in artificial intelligence: Insights from the social sciences”. In: *Artificial intelligence* 267 (2019), pp. 1–38.
- 13 [Mil+20] B. Millidge, A. Tschantz, A. K. Seth, and C. L. Buckley. “On the Relationship Between Active Inference and Control as Inference”. In: *International Workshop on Active Inference*. June 2020.
- 14 [Mil21] B. Millidge. “Applications of the Free Energy Principle to Machine Learning and Neuroscience”. PhD thesis. U. Edinburgh, June 2021.
- 15 [Mil+21] B. Millidge, A. Tschantz, A. Seth, and C. Buckley. “Neural Kalman Filtering”. In: (Feb. 2021). arXiv: [2102.10021](https://arxiv.org/abs/2102.10021) [cs.NE].
- 16 [Mil+21a] J. P. Miller, R. Taori, A. Raghunathan, S. Sagawa, P. W. Koh, V. Shankar, P. Liang, Y. Carmon, and L. Schmidt. “Accuracy on the Line: On the Strong Correlation Between Out-of-Distribution and In-Distribution Generalization”. In: *ICML*. 2021, pp. 7721–7735.
- 17 [Mil+21b] J. P. Miller, R. Taori, A. Raghunathan, S. Sagawa, P. W. Koh, V. Shankar, P. Liang, Y. Carmon, and L. Schmidt. “Accuracy on the Line: On the Strong Correlation Between Out-of-Distribution and In-Distribution Generalization”. In: *ICML*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 7721–7735.
- 18 [Min00a] T. Minka. *Bayesian linear regression*. Tech. rep. MIT, 2000.
- 19 [Min00b] T. Minka. *Bayesian model averaging is not model combination*. Tech. rep. MIT Media Lab, 2000.
- 20 [Min00c] T. Minka. *Estimating a Dirichlet distribution*. Tech. rep. MIT, 2000.
- 21 [Min01a] T. Minka. “A family of algorithms for approximate Bayesian inference”. PhD thesis. MIT, 2001.
- 22 [Min01b] T. Minka. “Expectation Propagation for approximate Bayesian inference”. In: *UAI*. 2001.
- 23 [Min04] T. Minka. *Power EP*. Tech. rep. MSR-TR-2004-149. 2004.
- 24 [Min05] T. Minka. *Divergence measures and message passing*. Tech. rep. MSR Cambridge, 2005.
- 25 [Min+18] T. Minka, J. Winn, J. Guiver, Y. Zaykov, D. Fabian, and J. Bronskill. *Infer.NET 0.9*. Microsoft Research Cambridge, 2018.
- 26 [Min78] M. Minoux. “Accelerated greedy algorithms for maximizing submodular set functions”. In: *Optimization Techniques*. Ed. by J. Stoer. Vol. 7. Lecture Notes in Control and Information Sciences. 10.1007/BFb0006528. Springer Berlin / Heidelberg, 1978, pp. 234–243.
- 27 [Mis+18] A. Mishkin, F. Kunstner, D. Nielsen, M. Schmidt, and M. E. Khan. “SLANG: Fast Structured Covariance Approximations for Bayesian Deep Learning with Natural Gradient”. In: *NIPS*. Curran Associates, Inc., 2018, pp. 6245–6255.
- 28 [Mit+19] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru. “Model cards for model reporting”. In: *Proceedings of the conference on fairness, accountability, and transparency*. 2019, pp. 220–229.
- 29 [Mit+20] J. Mitrovic, B. McWilliams, J. Walker, L. Buesing, and C. Blundell. *Representation Learning via Invariant Causal Mechanisms*. 2020. arXiv: [2010.07922](https://arxiv.org/abs/2010.07922) [cs.LG].
- 30 [Miy+18a] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. “Spectral Normalization for Generative Adversarial Networks”. In: *ICLR*. 2018.
- 31 [Miy+18b] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. “Spectral Normalization for Generative Adversarial Networks”. In: *ICLR*. 2018.
- 32 [Miy+18c] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. “Spectral Normalization for Generative Adversarial Networks”. In: *International Conference on Learning Representations*. 2018.
- 33 [MJ97] M. Meila and M. Jordan. *Triangulation by continuous embedding*. Tech. rep. 1605. MIT AI Lab, 1997.
- 34 [MK05] J. Mooij and H. Kappen. “Sufficient conditions for convergence of loopy belief propagation”. In: *UAI*. 2005.
- 35 [MK18] T. Miyato and M. Koyama. “cGANs with Projection Discriminator”. In: *International Conference on Learning Representations*. 2018.
- 36 [MK19] J. Menick and N. Kalchbrenner. “Generating high fidelity images with subscale pixel networks and multidimensional upscaling”. In: *ICLR*. 2019.
- 37 [MK97] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, 1997.
- 38 [MKG21] W. J. Ma, K. Kording, and D. Goldreich. *Bayesian models of perception and action*. MIT Press, 2021.
- 39 [MKH19] R. Müller, S. Kornblith, and G. E. Hinton. “When does label smoothing help?”. In: *NIPS*. 2019, pp. 4694–4703.
- 40 [MLK11] O. Martin, R. Kumar, and J. Lao. *Bayesian Modeling and Computation in Python*. CRC Press, 2011.
- 41 [MKS21] K. Murphy, A. Kumar, and S. Serghiou. “Risk score learning for COVID-19 contact tracing apps”. In: *Machine Learning for Healthcare*. Apr. 2021.
- 42 [ML02] T. Minka and J. Lafferty. “Expectation-propagation for the Generative Aspect Model”. In: *UAI*. Morgan Kaufmann Publishers Inc., 2002, pp. 352–359.
- 43 [ML16] S. Mohamed and B. Lakshminarayanan. “Learning in Implicit Generative Models”. In: (2016). arXiv: [1610.03483](https://arxiv.org/abs/1610.03483) [stat.ML].
- 44 [MLN19] P. Michel, O. Levy, and G. Neubig. “Are Sixteen Heads Really Better than One?”. In: *NIPS*. 2019.
- 45 [MLW19] V. Masrani, T. A. Le, and F. Wood. “The Thermodynamic Variational Objective”. In: *NIPS*. Curran Associates, Inc., 2019, pp. 11521–11530.
- 46 [MM01] T. K. Marks and J. R. Movellan. *Diffusion networks, products of experts, and factor analysis*. Tech. rep. University of California San Diego, 2001.
- 47 [MM90] D. Q. Mayne and H. Michalska. “Receding horizon control of nonlinear systems”. In: *IEEE Trans. Automat. Contr.* 35.7 (July 1990), pp. 814–824.
- 48 [MMC98] R. J. McEliece, D. J. C. MacKay, and J. F. Cheng. “Turbo decoding as an instance of Pearl’s ‘belief propagation algorithm’”. In: *IEEE J. on Selected Areas in Comm.* 16.2 (1998), pp. 140–152.
- 49 [MMP13] L. Malagò, M. Matteucci, and G. Pistoni. “Natural gradient, fitness modelling and model selection: A unifying perspective”. In: *IEEE Congress on Evolutionary Computation*. 2013.
- 50 [MMP87] J. Marroquin, S. Mitter, and T. Poggio. “Probabilistic solution of ill-posed problems in computational vision”. In: *JASA* 82.297 (1987), pp. 76–89.
- 51 [MMT17] C. J. Maddison, A. Mnih, and Y. W. Teh. “The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables”. In: *ICLR*. 2017.
- 52 [MN89] P. McCullagh and J. Nelder. *Generalized linear models*. 2nd edition. Chapman and Hall, 1989.
- 53 [MNG17a] L. Mescheder, S. Nowozin, and A. Geiger. “Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks”. In: *International Conference on Machine Learning*. PMLR, 2017, pp. 2391–2400.
- 54 [MNG17b] L. Mescheder, S. Nowozin, and A. Geiger. “The numerics of gans”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 1825–1835.
- 55 [Mni+15] V. Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (2015), pp. 529–533.
- 56 [Mni+16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. “Asynchronous Methods for Deep Reinforcement Learning”. In: *ICML*. 2016.
- 57 [MO14] M. Mirza and S. Osindero. “Conditional generative adversarial nets”. In: *arXiv preprint arXiv:1411.1784* (2014).
- 58 [Mob16] H. Mobahi. “Closed Form for Some Gaussian Convolutions”. In: (Feb. 2016). arXiv: [1602.05610](https://arxiv.org/abs/1602.05610) [math.CA].

- 1
- 2 [Moc+96] J. Mockus, W. Eddy, A. Mockus, L. Mockus, and G. Reklaitis. *Bayesian Heuristic Approach to Discrete and Global Optimization: Algorithms, Visualization, Software, and Applications*. Kluwer, 1996.
- 3
- 4 [Möh+19a] S. Mohamed, M. Rosca, M. Figurnov, and A. Mnih. "Monte Carlo Gradient Estimation in Machine Learning". In: (June 2019). arXiv: 1906.10652 [stat.ML].
- 5
- 6 [Moh+19b] H. Mohammadi, P. Challenor, M. Goodfellow, and D. Williamson. "Emulating computer models with step-discontinuous outputs using Gaussian processes". In: (Mar. 2019). arXiv: 1903.02071 [stat.ME].
- 7
- 8 [Mon81] G. Monge. "Mémoire sur la théorie des déblais et des remblais". In: *Histoire de l'Académie Royale des Sciences* (1781), pp. 666–704.
- 9
- 10 [Mor+21] W. Morningstar, C. Ham, A. Gallagher, B. Lakshminarayanan, A. Alemi, and J. Dillon. "Density of States Estimation for Out of Distribution Detection". In: *AISTATS*. Ed. by A. Banerjee and K. Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, 2021, pp. 3232–3240.
- 11
- 12 [Mor63] T. Morimoto. "Markov Processes and the H-Theorem". In: *J. Phys. Soc. Jpn.* 18.3 (Mar. 1963), pp. 328–331.
- 13 [MOT15] A. Mordvintsev, C. Olah, and M. Tyka. *Inceptionism: Going Deeper into Neural Networks*. <https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html>. Accessed: NA-NA-NA. 2015.
- 14
- 15 [Mov08] J. R. Movellan. "A minimum velocity approach to learning". In: *unpublished draft, Jan* (2008).
- 16 [MP01] K. Murphy and M. Paskin. "Linear time inference in hierarchical HMMs". In: *NIPS*. 2001.
- 17 [MP21] D. Mazza and M. Pagani. "Automatic Differentiation in PCF". In: *Proc. ACM Program. Lang.* 5.POPL (Jan. 2021).
- 18
- 19 [MP95] D. MacKay and L. Peto. "A hierarchical dirichlet language model". In: *Natural Language Engineering* 1.3 (1995), pp. 289–307.
- 20
- 21 [MPS18] O. Mangoubi, N. S. Pillai, and A. Smith. "Does Hamiltonian Monte Carlo mix faster than a random walk on multimodal densities?". In: (Aug. 2018). arXiv: 1808.03230 [math.PR].
- 22 [MR09] A. Melkumyan and F. Ramos. "A Sparse Covariance Function for Exact Gaussian Process Inference in Large Datasets". In: *IJCAI*. 2009, pp. 1936–1942.
- 23
- 24 [MR10] B. Milch and S. Russell. "Extending Bayesian Networks to the Open-Universe Case". In: *Heuristics, Probability and Causality: A Tribute to Judea Pearl*. Ed. by R. Dechter, H. Geffner, and J. Y. Halpern. College Publications, 2010.
- 25
- 26 [MRW19] B. Mittelstadt, C. Russell, and S. Wachter. "Explaining explanations in AI". In: *Proceedings of the conference on fairness, accountability, and transparency*. 2019, pp. 279–288.
- 27 [MS96] V Matveev V and R Shrock. "Complex-temperature singularities in Potts models on the square lattice". en. In: *Phys. Rev. E Stat. Phys. Plasmas Fluids Relat. Interdiscip. Topics* 54.6 (Dec. 1996), pp. 6174–6185.
- 28
- 29 [MS99] C. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, 1999.
- 30 [MSA18] S. Makridakis, E. Spiliotis, and V. Assimakopoulos. "The M4 Competition: Results, findings, conclusion and way forward". In: *Int. J. Forecast.* 34.4 (Oct. 2018), pp. 802–808.
- 31
- 32 [MSB21] B. Millidge, A. Seth, and C. L. Buckley. "Predictive Coding: a Theoretical and Experimental Review". In: (July 2021). arXiv: 2107.12979 [cs.AI].
- 33
- 34 [MT12] A. Mnih and Y. W. Teh. "A fast and simple algorithm for training neural probabilistic language models". In: *ICML*. 2012, pp. 419–426.
- 35 [MTB20] B. Millidge, A. Tschantz, and C. L. Buckley. "Predictive Coding Approximates Backprop along Arbitrary Computation Graphs". In: *arXiv preprint arXiv:2006.04182* (2020).
- 36
- 37 [MTM14] C. J. Maddison, D. Tarlow, and T. Minka. "A\* Sampling". In: *NIPS*. 2014.
- 38 [Mua+17] K. Muandet, K. Fukumizu, B. Sriperumbudur, and B. Schölkopf. "Kernel Mean Embedding of Distributions: A Review and Beyond". In: *Foundations and Trends* 10.1–2 (2017), pp. 1–141.
- 39
- 40 [Mua+20] K. Muandet, A. Mehrjou, S. K. Lee, and A. Raj. *Dual Instrumental Variable Regression*. 2020.
- 41 [Muk+18] S. Mukherjee, D. Shankar, A. Ghosh, N. Tahawadekar, P. Kompaali, S. Sarawagi, and K. Chaudhury. "ARMDN: Associative and Recurrent Mixture Density Networks for eRetail Demand Forecasting". In: (Mar. 2018). arXiv: 1803.03800 [cs.LG].
- 42
- 43 [Müü+19a] T. Müller, B. McWilliams, F. Rousselle, M. Gross, and J. Novák. "Neural Importance Sampling". In: *SIGGRAPH*. 2019.
- 44
- 45 [Müü+19b] T. Müller, B. McWilliams, F. Rousselle, M. Gross, and J. Novák. "Neural importance sampling". In: *ACM Transactions on Graphics* 38.5 (2019), p. 145.
- 46
- 47
- 48
- 49
- 50
- 51
- 52
- 53
- 54
- 55
- 56
- 57
- 58
- 59
- 60
- 61
- 62
- 63
- 64
- 65
- 66
- 67
- 68
- 69
- 70
- 71
- 72
- 73
- 74
- 75
- 76
- 77
- 78
- 79
- 80
- 81
- 82
- 83
- 84
- 85
- 86
- 87
- 88
- 89
- 90
- 91
- 92
- 93
- 94
- 95
- 96
- 97
- 98
- 99
- 100
- 101
- 102
- 103
- 104
- 105
- 106
- 107
- 108
- 109
- 110
- 111
- 112
- 113
- 114
- 115
- 116
- 117
- 118
- 119
- 120
- 121
- 122
- 123
- 124
- 125
- 126
- 127
- 128
- 129
- 130
- 131
- 132
- 133
- 134
- 135
- 136
- 137
- 138
- 139
- 140
- 141
- 142
- 143
- 144
- 145
- 146
- 147
- 148
- 149
- 150
- 151
- 152
- 153
- 154
- 155
- 156
- 157
- 158
- 159
- 160
- 161
- 162
- 163
- 164
- 165
- 166
- 167
- 168
- 169
- 170
- 171
- 172
- 173
- 174
- 175
- 176
- 177
- 178
- 179
- 180
- 181
- 182
- 183
- 184
- 185
- 186
- 187
- 188
- 189
- 190
- 191
- 192
- 193
- 194
- 195
- 196
- 197
- 198
- 199
- 200
- 201
- 202
- 203
- 204
- 205
- 206
- 207
- 208
- 209
- 210
- 211
- 212
- 213
- 214
- 215
- 216
- 217
- 218
- 219
- 220
- 221
- 222
- 223
- 224
- 225
- 226
- 227
- 228
- 229
- 230
- 231
- 232
- 233
- 234
- 235
- 236
- 237
- 238
- 239
- 240
- 241
- 242
- 243
- 244
- 245
- 246
- 247
- 248
- 249
- 250
- 251
- 252
- 253
- 254
- 255
- 256
- 257
- 258
- 259
- 260
- 261
- 262
- 263
- 264
- 265
- 266
- 267
- 268
- 269
- 270
- 271
- 272
- 273
- 274
- 275
- 276
- 277
- 278
- 279
- 280
- 281
- 282
- 283
- 284
- 285
- 286
- 287
- 288
- 289
- 290
- 291
- 292
- 293
- 294
- 295
- 296
- 297
- 298
- 299
- 300
- 301
- 302
- 303
- 304
- 305
- 306
- 307
- 308
- 309
- 310
- 311
- 312
- 313
- 314
- 315
- 316
- 317
- 318
- 319
- 320
- 321
- 322
- 323
- 324
- 325
- 326
- 327
- 328
- 329
- 330
- 331
- 332
- 333
- 334
- 335
- 336
- 337
- 338
- 339
- 340
- 341
- 342
- 343
- 344
- 345
- 346
- 347
- 348
- 349
- 350
- 351
- 352
- 353
- 354
- 355
- 356
- 357
- 358
- 359
- 360
- 361
- 362
- 363
- 364
- 365
- 366
- 367
- 368
- 369
- 370
- 371
- 372
- 373
- 374
- 375
- 376
- 377
- 378
- 379
- 380
- 381
- 382
- 383
- 384
- 385
- 386
- 387
- 388
- 389
- 390
- 391
- 392
- 393
- 394
- 395
- 396
- 397
- 398
- 399
- 400
- 401
- 402
- 403
- 404
- 405
- 406
- 407
- 408
- 409
- 410
- 411
- 412
- 413
- 414
- 415
- 416
- 417
- 418
- 419
- 420
- 421
- 422
- 423
- 424
- 425
- 426
- 427
- 428
- 429
- 430
- 431
- 432
- 433
- 434
- 435
- 436
- 437
- 438
- 439
- 440
- 441
- 442
- 443
- 444
- 445
- 446
- 447
- 448
- 449
- 450
- 451
- 452
- 453
- 454
- 455
- 456
- 457
- 458
- 459
- 460
- 461
- 462
- 463
- 464
- 465
- 466
- 467
- 468
- 469
- 470
- 471
- 472
- 473
- 474
- 475
- 476
- 477
- 478
- 479
- 480
- 481
- 482
- 483
- 484
- 485
- 486
- 487
- 488
- 489
- 490
- 491
- 492
- 493
- 494
- 495
- 496
- 497
- 498
- 499
- 500
- 501
- 502
- 503
- 504
- 505
- 506
- 507
- 508
- 509
- 510
- 511
- 512
- 513
- 514
- 515
- 516
- 517
- 518
- 519
- 520
- 521
- 522
- 523
- 524
- 525
- 526
- 527
- 528
- 529
- 530
- 531
- 532
- 533
- 534
- 535
- 536
- 537
- 538
- 539
- 540
- 541
- 542
- 543
- 544
- 545
- 546
- 547
- 548
- 549
- 550
- 551
- 552
- 553
- 554
- 555
- 556
- 557
- 558
- 559
- 560
- 561
- 562
- 563
- 564
- 565
- 566
- 567
- 568
- 569
- 570
- 571
- 572
- 573
- 574
- 575
- 576
- 577
- 578
- 579
- 580
- 581
- 582
- 583
- 584
- 585
- 586
- 587
- 588
- 589
- 590
- 591
- 592
- 593
- 594
- 595
- 596
- 597
- 598
- 599
- 600
- 601
- 602
- 603
- 604
- 605
- 606
- 607
- 608
- 609
- 610
- 611
- 612
- 613
- 614
- 615
- 616
- 617
- 618
- 619
- 620
- 621
- 622
- 623
- 624
- 625
- 626
- 627
- 628
- 629
- 630
- 631
- 632
- 633
- 634
- 635
- 636
- 637
- 638
- 639
- 640
- 641
- 642
- 643
- 644
- 645
- 646
- 647
- 648
- 649
- 650
- 651
- 652
- 653
- 654
- 655
- 656
- 657
- 658
- 659
- 660
- 661
- 662
- 663
- 664
- 665
- 666
- 667
- 668
- 669
- 670
- 671
- 672
- 673
- 674
- 675
- 676
- 677
- 678
- 679
- 680
- 681
- 682
- 683
- 684
- 685
- 686
- 687
- 688
- 689
- 690
- 691
- 692
- 693
- 694
- 695
- 696
- 697
- 698
- 699
- 700
- 701
- 702
- 703
- 704
- 705
- 706
- 707
- 708
- 709
- 710
- 711
- 712
- 713
- 714
- 715
- 716
- 717
- 718
- 719
- 720
- 721
- 722
- 723
- 724
- 725
- 726
- 727
- 728
- 729
- 730
- 731
- 732
- 733
- 734
- 735
- 736
- 737
- 738
- 739
- 740
- 741
- 742
- 743
- 744
- 745
- 746
- 747
- 748
- 749
- 750
- 751
- 752
- 753
- 754
- 755
- 756
- 757
- 758
- 759
- 760
- 761
- 762
- 763
- 764
- 765
- 766
- 767
- 768
- 769
- 770
- 771
- 772
- 773
- 774
- 775
- 776
- 777
- 778
- 779
- 780
- 781
- 782
- 783
- 784
- 785
- 786
- 787
- 788
- 789
- 790
- 791
- 792
- 793
- 794
- 795
- 796
- 797
- 798
- 799
- 800
- 801
- 802
- 803
- 804
- 805
- 806
- 807
- 808
- 809
- 810
- 811
- 812
- 813
- 814
- 815
- 816
- 817
- 818
- 819
- 820
- 821
- 822
- 823
- 824
- 825
- 826
- 827
- 828
- 829
- 830
- 831
- 832
- 833
- 834
- 835
- 836
- 837
- 838
- 839
- 840
- 841
- 842
- 843
- 844
- 845
- 846
- 847
- 848
- 849
- 850
- 851
- 852
- 853
- 854
- 855
- 856
- 857
- 858
- 859
- 860
- 861
- 862
- 863
- 864
- 865
- 866
- 867
- 868
- 869
- 870
- 871
- 872
- 873
- 874
- 875
- 876
- 877
- 878
-

- [NCT16a] S. Nowozin, B. Cseke, and R. Tomioka. "f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization". In: *NIPS*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., 2016, pp. 271–279.
- [NCT16b] S. Nowozin, B. Cseke, and R. Tomioka. "f-gan: Training generative neural samplers using variational divergence minimization". In: *NIPS*. 2016, pp. 271–279.
- [NCT16c] S. Nowozin, B. Cseke, and R. Tomioka. "f-gan: Training generative neural samplers using variational divergence minimization". In: *Advances in neural information processing systems*. 2016, pp. 271–279.
- [ND20] O. Nachum and B. Dai. "Reinforcement Learning via Fenchel-Rockafellar Duality". In: (Jan. 2020). arXiv: 2001.01866 [cs, LG].
- [NDL20] A. Nishimura, D. Dunson, and J. Lu. "Discontinuous Hamiltonian Monte Carlo for discrete parameters and discontinuous likelihoods". In: *Biometrika* (2020).
- [Nea00] R. Neal. "Markov Chain Sampling Methods for Dirichlet Process Mixture Models". In: *JCGS* 9.2 (2000), pp. 249–265.
- [Nea01] R. M. Neal. "Annealed Importance Sampling". In: *Statistics and Computing* 11 (2001), pp. 125–139.
- [Nea03] R. Neal. "Slice sampling". In: *Annals of Statistics* 31.3 (2003), pp. 7–5767.
- [Nea+08] R. Neal et al. "Computing likelihood functions for high-energy physics experiments when distributions are defined by simulators with nuisance parameters". In: (2008).
- [Nea10] R. Neal. "MCMC using Hamiltonian Dynamics". In: *Handbook of Markov Chain Monte Carlo*. Ed. by S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. Chapman & Hall, 2010.
- [Nea+11] R. M. Neal et al. "MCMC using Hamiltonian dynamics". In: *Handbook of markov chain monte carlo* 2.11 (2011), p. 2.
- [Nea12] R. C. Neath. "On Convergence Properties of the Monte Carlo EM Algorithm". In: *arXiv [math.ST]* (June 2012).
- [Nea20] B. Neal. *Introduction to Causal Inference from a Machine Learning Perspective*. 2020.
- [Nea92] R. Neal. "Connectionist learning of belief networks". In: *Artificial Intelligence* 56 (1992), pp. 71–113.
- [Nea93] R. M. Neal. *Probabilistic Inference using Markov Chain Monte Carlo Methods*. Tech. rep. CRG-TR-93-1. 144pp. Dept. of Computer Science, University of Toronto, 1993.
- [Nea95] R. M. Neal. "Bayesian Learning for Neural Networks". PhD thesis, University of Toronto, 1995.
- [Nea96] R. Neal. *Bayesian learning for neural networks*. Springer, 1996.
- [Nef+02] A. Neffan, L. Liang, X. Pi, X. Liu, and K. Murphy. "Dynamic Bayesian Networks for Audio-Visual Speech Recognition". In: *J. Applied Signal Processing* (2002).
- [Neg+21] J. Negrea, J. Yang, H. Feng, D. M. Roy, and J. H. Huggins. "Statistical inference with stochastic gradient algorithms". In: (2021).
- [Neil+18] D. Neil, J. Briody, A. Lacoste, A. Sim, P. Creed, and A. Saffari. "Interpretable graph convolutional neural networks for inference on noisy knowledge graphs". In: *arXiv preprint arXiv:1812.00279* (2018).
- [Nel21] Nelson Elhage and Neel Nanda and Catherine Olsson and Tom Henighan and Nicholas Joseph and Ben Mann and Amanda Askell and Yuntao Bai and Anna Chen and Tom Conerly and Nova DasSarma and Dawn Drain and Deep Ganguli and Zac Hatfield-Dodds and Danny Hernandez and Andy Jones and Jackson Kernion and Liane Lovitt and Kamal Ndousse and Dario Amodei and Tom Brown and Jack Clark and Jared Kaplan and Sam McCandlish and Chris Olah. *A Mathematical Framework for Transformer Circuits*. Tech. rep. Anthropic, 2021.
- [Neu11] G. Neumann. "Variational Inference for Policy Search in Changing Situations". In: *ICML*. 2011, pp. 817–824.
- [Ney+17] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. "Exploring generalization in deep learning". In: *NIPS*. 2017.
- [NG01] D. Nilsson and J. Goldberger. "Sequentially finding the N-Best List in Hidden Markov Models". In: *IJCAI*. 2001, pp. 1280–1285.
- [Ngi+11] J. Ngiam, Z. Chen, P. W. Koh, and A. Y. Ng. "Learning deep energy models". In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 1105–1112.
- [Ngu+16] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. *Synthesizing the preferred inputs for neurons in neural networks via deep generator networks*. 2016. arXiv: 1605.09304 [cs, NE].
- [Ngu+18a] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner. "Variational Continual Learning". In: *ICLR*. 2018.
- [Ngu+18b] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner. "Variational Continual Learning". In: *ICLR*. 2018.
- [Ngu+19] T. T. Nguyen, Q. V. H. Nguyen, D. T. Nguyen, D. T. Nguyen, Thien Huynh-The, S. Nahavandi, T. T. Nguyen, Q.-V. Pham, and C. M. Nguyen. "Deep Learning for Deepfakes Creation and Detection: A Survey". In: (Sept. 2019). arXiv: 1909.11573 [cs, CV].
- [Ngu+21] T. Nguyen, R. Novak, L. Xiao, and J. Lee. "Dataset Distillation with Infinitely Wide Convolutional Networks". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. 2021.
- [NHL98a] R. M. Neal and G. E. Hinton. "A new view of the EM algorithm that justifies incremental and other variants". In: *Learning in Graphical Models*. Ed. by M. Jordan. MIT Press, 1998.
- [NHL98b] R. M. Neal and G. E. Hinton. "A View of the EM Algorithm that Justifies Incremental, Sparse, and other Variants". In: *Learning in Graphical Models*. Ed. by M. I. Jordan. Dordrecht: Springer Netherlands, 1998, pp. 355–368.
- [NHL98c] E. Nalisnick, J. M. Hernández-Lobato, and P. Smyth. "Dropout as a Structured Shrinkage Prior". In: *ICML*. 1998.
- [NHR99] A. Ng, D. Harada, and S. Russell. "Policy invariance under reward transformations: Theory and application to reward shaping". In: *ICML*. 1999.
- [NI92] H. Nagamochi and T. Ibaraki. "Computing edge-connectivity of multigraphs and capacitated graphs". In: *SIAM J. Discrete Math.* 5 (1992), pp. 54–66.
- [Nic+21] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen. "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models". In: (Dec. 2021). arXiv: 2112.10741 [cs, CV].
- [Nijkamp+19] E. Nijkamp, M. Hill, S.-C. Zhu, and Y. N. Wu. "Learning non-convergent non-persistent short-run MCMC to learn energy-based model". In: *Advances in Neural Information Processing Systems*. 2019, pp. 5232–5242.
- [Nil98] D. Nilsson. "An efficient algorithm for finding the M most probable configurations in a probabilistic expert system". In: *Statistics and Computing* 8 (1998), pp. 159–173.
- [Nix+19] J. Nixon, M. Dusenberry, L. Zhang, G. Jerfel, and D. Tran. "Measuring Calibration in Deep Learning". In: (Apr. 2019). arXiv: 1904.01685 [cs, LG].
- [NJ00] A. Y. Ng and M. Jordan. "PEGASUS: A policy search method for large MDPs and POMDPs". In: *UAI*. 2000.
- [NJB05] M. Narasimhan, N. Jojic, and J. A. Bilmes. "Q-clustering". In: *Advances in Neural Information Processing Systems* 18 (2005), pp. 979–986.
- [NK17] V. Nagarajan and J. Z. Kolter. "Gradient descent GAN optimization is locally stable". In: *Advances in neural information processing systems*. 2017, pp. 5585–5595.
- [NKI10] K. Nagano, Y. Kawahara, and S. Iwata. "Minimum average cost clustering". In: *Advances in Neural Information Processing Systems* 23 (2010), pp. 1759–1767.
- [NKMG03] K. Nummiaro, E. Koller-Meier, and L. V. Gool. "An Adaptive Color-Based Particle Filter". In: *Image and Vision Computing* 21.1 (2003), pp. 99–110.
- [NLIS15] C. A. Naesseth, F. Lindsten, and T. B. Schön. "Nested Sequential Monte Carlo Methods". In: *ICML*. Feb. 2015.
- [NLIS19] C. A. Naesseth, F. Lindsten, and T. B. Schön. "Elements of Sequential Monte Carlo". In: *Foundations and Trends in Machine Learning* (2019).
- [NM12] A. Nenkova and K. McKeown. "A survey of text summarization techniques". In: *Mining text data*. Springer, 2012, pp. 43–76.
- [NMC05] A. Niculescu-Mizil and R. Caruana. "Predicting Good Probabilities with Supervised Learning". In: *ICML*. 2005.
- [NP19] W. Nie, N. Narodytska, and A. Patel. "RelGAN: Relational Generative Adversarial Networks for Text Generation". In: *International Conference on Learning Representations*. 2019.
- [Noc+21] L. Noci, K. Roth, G. Bachmann, S. Nowozin, and T. Hofmann. "Disentangling the Roles of Curation, Data-Augmentation and the Prior in the Cold Posterior Effect". In: *NIPS*. June 2021.
- [Noé+19] F. Noé, S. Olsson, J. Köhler, and H. Wu. "Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning". In: *Science* 365 (2019).
- [Nou+02] M. N. Nounou, B. R. Bakshi, P. K. Goel, and X. Shen. "Process modeling by Bayesian latent variable regression". In: *Am. Inst. Chemical Engineers Journal* 48.8 (Aug. 2002), pp. 1775–1793.
- [Nov+19] R. Novak, L. Xiao, J. Lee, Y. Bahri, G. Yang, J. Hron, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein. "Bayesian Deep Convolutional Networks with Many Channels are Gaussian Processes". In: *ICLR*. 2019.

- 1
- 2 [Now+14] S. Nowozin, P. V. Gehler, J. Jancsary, and C. H. Lampert, eds. *Advanced Structured Prediction*. en. The MIT Press, Dec. 2014.
- 3
- 4 [NP03] W. K. Newey and J. L. Powell. "Instrumental variable estimation of nonparametric models". In: *Econometrica* 71.5 (2003), pp. 1565–1578.
- 5 [NP12] N. Nunn and D. Puga. "Ruggedness: The blessing of bad geography in Africa". In: *Rev. Econ. Stat.* 94.1 (2012).
- 6
- 7 [NR00a] A. Ng and S. Russell. "Algorithms for inverse reinforcement learning". In: *ICML*. 2000.
- 8 [NR00b] A. Y. Ng and S. Russell. "Algorithms for Inverse Reinforcement Learning". In: *Proc. 17th International Conf. on Machine Learning*. Citeseer. 2000.
- 9 [NR08] H. Nickisch and C. E. Rasmussen. "Approximations for Binary Gaussian Process Classification". In: *JMLR* 9.Oct (2008), pp. 2035–2078.
- 10
- 11 [NR94] M. Newton and A. Raftery. "Approximate Bayesian Inference with the Weighted Likelihood Bootstrap". In: *J. of Royal Stat. Soc. Series B* 56.1 (1994), pp. 3–48.
- 12 [NS01] K. Nowicki and T. A. B. Snijders. "Estimation and Prediction for Stochastic Blockstructures". In: *Journal of the American Statistical Association* 96.455 (2001), pp. 1077–1087.
- 13
- 14 [NS17] E. Nalisnick and P. Smyth. "Variational Reference Priors". In: *ICLR Workshop*. 2017.
- 15 [NS18] E. Nalisnick and P. Smyth. "Learning Priors for Invariance". In: *AISTATS*. 2018.
- 16 [NW06] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2006.
- 17
- 18 [NW20] X Nie and S Wager. "Quasi-oracle estimation of heterogeneous treatment effects". In: *Biometrika* 108.2 (Sept. 2020), pp. 299–319. eprint: <https://academic.oup.com/biomet/article-pdf/108/2/299/3793893/asaa076.pdf>.
- 19
- 20 [NWF78] G. Nemhauser, L. Wolsey, and M. Fisher. "An analysis of approximations for maximizing submodular set functions—I". In: *Mathematical Programming* 14.1 (1978), pp. 265–294.
- 21 [NWJ09] X. Nguyen, M. J. Wainwright, and M. I. Jordan. "On Surrogate Loss Functions and f-Divergences". In: *Ann. Stat.* 37.2 (2009), pp. 876–904.
- 22
- 23 [NWJ+09] X. Nguyen, M. J. Wainwright, M. I. Jordan, et al. "On surrogate loss functions and f-divergences". In: *The Annals of Statistics* 37.2 (2009), pp. 876–904.
- 24
- 25 [NWJ10a] X. Nguyen, M. J. Wainwright, and M. I. Jordan. "Estimating Divergence Functionals and the Likelihood Ratio by Convex Risk Minimization". In: *IEEE Trans. Inf. Theory* 56.11 (Nov. 2010), pp. 5847–5861.
- 26
- 27 [NWJ10b] X. Nguyen, M. J. Wainwright, and M. I. Jordan. "Estimating divergence functionals and the likelihood ratio by convex risk minimization". In: *IEEE Transactions on Information Theory* 56.11 (2010), pp. 5847–5861.
- 28
- 29 [NY83] A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley, 1983.
- 30
- 31 [NYC15] A. Nguyen, J. Yosinski, and J. Clune. "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images". In: *CVPR*. 2015.
- 32
- 33 [OA09] M. Opper and C. Archambeau. "The variational Gaussian approximation revisited". en. In: *Neural Comput.* 21.3 (Mar. 2009), pp. 786–792.
- 34 [OA21] S. W. Ober and L. Aitchison. "Global inducing point variational posteriors for Bayesian neural networks and deep Gaussian processes". In: *ICML*. 2021.
- 35 [Obe+19] F. Obermeyer, E. Bingham, M. Jankowiak, J. Chiu, N. Pradhan, A. Rush, and N. Goodman. "Tensor Variable Elimination for Plated Factor Graphs". In: *ICML*. Feb. 2019.
- 36 [OCM21] L. A. Ortega, R. Cabafas, and A. R. Masegosa. "Diversity and Generalization in Neural Network Ensembles". In: (Oct. 2021). arXiv: [2110.13786 \[cs.LG\]](https://arxiv.org/abs/2110.13786).
- 37
- 38 [ODK96] M. Ostendorf, V. Digalakis, and O. Kimball. "From HMMs to segment models: a unified view of stochastic modeling for speech recognition". In: *IEEE Trans. on Speech and Audio Processing* 4.5 (1996), pp. 360–378.
- 39
- 40 [OF96] B. A. Olshausen and D. J. Field. "Emergence of simple cell receptive field properties by learning a sparse code for natural images". In: *Nature* 381 (1996), pp. 607–609.
- 41 [O'H78] A. O'Hagan. "Curve Fitting and Optimal Design for Prediction". In: *J. of Royal Stat. Soc. Series B* 40 (1978), pp. 1–42.
- 42
- 43 [OKK16] A. Van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. "Pixel Recurrent Neural Networks". In: *ICML*. 2016.
- 44
- 45 [Oli18] Y. Ollivier. "Online natural gradient as a Kalman filter". en. In: *Electron. J. Stat.* 12.2 (2018), pp. 2930–2961.
- 46
- 47
- [OLV18] A. van den Oord, Y. Li, and O. Vinyals. "Representation Learning with Contrastive Predictive Coding". In: (July 2018). arXiv: [1807.03748 \[cs.LG\]](https://arxiv.org/abs/1807.03748).
- [OM96] P. V. Overschee and B. D. Moor. *Subspace Identification for Linear Systems: Theory, Implementation, Applications*. Kluwer Academic Publishers, 1996.
- [OMS17] C. Olah, A. Mordvintsev, and L. Schubert. "Feature Visualization". In: *Distill* 2.11 (Nov. 2017).
- [ON09] B. O'Neill. "Exchangeability, Correlation, and Bayes' Effect". In: *Int. Stat. Rev.* 77.2 (2009), pp. 241–250.
- [ONS18] V. M.-H. Ong, D. J. Nott, and M. S. Smith. "Gaussian Variational Approximation With a Factor Covariance Structure". In: *J. Comput. Graph. Stat.* 27.3 (July 2018), pp. 465–478.
- [oor+16] A. Van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. "WaveNet: A Generative Model for Raw Audio". In: (Dec. 2016). arXiv: [1609.03499 \[cs.SD\]](https://arxiv.org/abs/1609.03499).
- [Oor+16] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. "Conditional Image Generation with PixelCNN Decoders". In: (2016). arXiv: [1606.05328 \[cs.CV\]](https://arxiv.org/abs/1606.05328).
- [Oor+18] A. van den Oord et al. "Parallel WaveNet: Fast High-Fidelity Speech Synthesis". In: *ICML*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholm Sweden: PMLR, 2018, pp. 3918–3926.
- [Oor+19] A. van den Oord, B. Poole, O. Vinyals, and A. Razavi. "Fixing Posterior Collapses with delta-VAEs". In: *ICLR*. 2019.
- [OOS17] A. Odena, C. Olah, and J. Shlens. "Conditional image synthesis with auxiliary classifier gans". In: *International conference on machine learning*. 2017, pp. 2642–2651.
- [Opt88] Optimal information processing and Bayes' theorem. In: *The American Statistician* 42.4 (1988), pp. 278–280.
- [OR20] A. Owen and D. Rudolf. "A strong law of large numbers for scrambled net integration". In: (Feb. 2020). arXiv: [2002.07859 \[math.MA\]](https://arxiv.org/abs/2002.07859).
- [Ore+20] B. N. Oreshkin, D. Carpow, N. Chapados, and Y. Bengio. "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting". In: *ICLR*. 2020.
- [Ort+19] P. A. Ortega et al. "Meta-learning of Sequential Strategies". In: (May 2019). arXiv: [1905.03030 \[cs.LG\]](https://arxiv.org/abs/1905.03030).
- [Osa+18] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters. "An Algorithmic Perspective on Imitation Learning". In: *Foundations and Trends in Robotics* 7.1–2 (2018), pp. 1–179.
- [Osa+19a] K. Osawa, S. Swaroop, A. Jain, R. Eschenhagen, R. E. Turner, R. Yokota, and M. E. Khan. "Practical Deep Learning with Bayesian Principles". In: *NIPS*. 2019.
- [Osa+19b] K. Osawa, Y. Tsuji, Y. Ueno, A. Naruse, R. Yokota, and S. Matsukura. "Large-Scale Distributed Second-Order Optimization Using Kronecker-Factored Approximate Curvature for Deep Convolutional Neural Networks". In: *CVPR*. 2019.
- [Obs16] I. Osband. "Risk versus Uncertainty in Deep Learning: Bayes, Bootstrap and the Dangers of Dropout". In: *NIPS workshop on Bayesian deep learning*. 2016.
- [Obs+21] I. Osband, Z. Wen, S. M. Asghari, V. Dwaracherla, B. Hao, M. Ibrahim, D. Lawson, X. Lu, B. O'Donoghue, and B. Van Roy. "Evaluating Predictive Distributions: Does Bayesian Deep Learning Work?". In: (Oct. 2021). arXiv: [2110.04629 \[cs.LG\]](https://arxiv.org/abs/2110.04629).
- [Ose11] I. V. Oseledets. "Tensor-Train Decomposition". In: *SIAM J. Sci. Comput.* 33.5 (Jan. 2011), pp. 2295–2317.
- [OT05] A. B. Owen and S. D. Tribble. "A quasi-Monte Carlo Metropolis algorithm". en. In: *PNAS* 102.25 (June 2005), pp. 8844–8849.
- [OTT19] M. Okada, S. Takenaka, and T. Taniguchi. "Multi-person Pose Tracking using Sequential Monte Carlo with Probabilistic Neural Pose Predictor". In: (Sept. 2019). arXiv: [1909.07031 \[cs.LG\]](https://arxiv.org/abs/1909.07031).
- [Ova+19] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek. "Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift". In: *NIPS*. 2019.
- [OVK17] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. "Neural Discrete Representation Learning". In: *NIPS*. 2017.
- [Owe13] A. B. Owen. *Monte Carlo theory, methods and examples*. 2013.
- [Owe17] A. B. Owen. "A randomized Halton algorithm in R". In: *arXiv [stat.CO]* (June 2017).
- [Oxi11] J. Oxley. *Matroid Theory: Second Edition*. Oxford University Press, 2011.

- [Pac+14] J. Pacheco, S. Zuffi, M. Black, and E. Sudderth. "Preserving Modes and Messages via Diverse Particle Selection". In: *ICML*. Jan. 2014, pp. 1152–1160.
- [Pal] *Explainable AI in Practice Falls Short of Transparency Goals*. <https://partnershiponai.org/xai-in-practice/>. Accessed: 2021-11-23.
- [Pan+10] L. Faninski, Y. Ahmadian, D. G. Ferreira, S. Koyama, K. Rahnama Rad, M. Vidne, J. Vogelstein, and W. Wu. "A new look at state-space models for neural data". In: *J. Comput. Neurosci.* 29.1-2 (Aug. 2010), pp. 107–126.
- [Pan+21] G. Pang, C. Shen, L. Cao, and A. Van Den Hengel. "Deep Learning for Anomaly Detection: A Review". In: *ACM Comput. Surv.* 54.2 (Mar. 2021), pp. 1–38.
- [Pan+22] K. Pandey, A. Mukherjee, P. Rai, and A. Kumar. "DiffuseVAE: Efficient, Controllable and High-Fidelity Generation from Low-Dimensional Latents". In: (Jan. 2022). arXiv: 2201.00308 [cs.LG].
- [Pap+17] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z Berkay Celik, and A. Swami. "Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples". In: *ACM Asia Conference on Computer and Communications Security*. 2017.
- [Pap+19] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. "Normalizing Flows for Probabilistic Modeling and Inference". In: (Dec. 2019). arXiv: 1912.02762 [stat.ML].
- [Par+19] S. Parameswaran, C. Deledalle, L. Denis, and T. Q. Nguyen. "Accelerating GMM-Based Patch Priors for Image Restoration: Three Ingredients for a 100× Speed-Up". In: *IEEE Trans. Image Process.* 28.2 (Feb. 2019), pp. 687–698.
- [Par81] G. Parisi. "Correlation functions and computer simulations". In: *Nuclear Physics B* 180.3 (1981), pp. 378–384.
- [Pas+02] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. "Identity Uncertainty and Citation Matching". In: *NIPS*. 2002.
- [Pas03] M. Paskin. "Thin Junction Tree Filters for Simultaneous Localization and Mapping". In: *IJCAI*. 2003.
- [Pas+21a] A. Paszke, D. Johnson, D. Duvenaud, D. Vytiniotis, A. Radul, M. Johnson, J. Ragan-Kelley, and D. MacLaurin. "Getting to the Point: Index Sets and Parallelism-Preserving Autodiff for Pointful Array Programming". In: *Proc. ACM Program. Lang.* 5.ICFP (Aug. 2021).
- [Pas+21b] A. Paszke, M. J. Johnson, R. Frostig, and D. MacLaurin. "Parallelism-Preserving Automatic Differentiation for Second-Order Array Languages". In: *Proceedings of the 9th ACM SIGPLAN International Workshop on Functional High-Performance and Numerical Computing*. FHPNC 2021, Virtual, Republic of Korea: Association for Computing Machinery, 2021, 13–23.
- [PB14] R. Pascanu and Y. Bengio. "Revisiting Natural Gradient for Deep Networks". In: *ICLR*. 2014.
- [PBM16a] J. Peters, P. Bühlmann, and N. Meinshausen. "Causal inference by using invariant prediction: identification and confidence intervals". In: *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 78.5 (2016), pp. 947–1012.
- [PBM16b] J. Peters, P. Bühlmann, and N. Meinshausen. "Causal inference using invariant prediction: identification and confidence intervals". In: *J. of Royal Stat. Soc. Series B* 78.5 (2016), pp. 947–1012.
- [PC08] T. Park and G. Casella. "The Bayesian Lasso". In: *JASA* 103.482 (2008), pp. 681–686.
- [PC09] J. Paisley and L. Carin. "Nonparametric Factor Analysis with Beta Process Priors". In: *ICML*. 2009.
- [PC12] N. Pinto and D. D. Cox. "High-throughput-derived biologically-inspired features for unconstrained face recognition". In: *Image Vis. Comput.* 30.3 (Mar. 2012), pp. 159–168.
- [PC21] G. Pleiss and J. P. Cunningham. "The Limitations of Large Width in Neural Networks: A Deep Gaussian Process Perspective". In: *NIPS*. June 2021.
- [PD03] J. D. Park and A. Darwiche. "A Differential Semantics for Jointree Algorithms". In: *NIPS*. MIT Press, 2003, pp. 801–808.
- [PD11] H. Poon and P. Domingos. "Sum-Product Networks: A New Deep Architecture". In: *UAI*. Java code at <http://alchemy.cs.washington.edu/spn/>. Short intro at <http://lessoned.blogspot.com/2011/10/intro-to-sum-product-networks.html>. 2011.
- [PdC20] F.-P. Paty, A. d'Aspremont, and M. Cuturi. "Regularity as Regularization: Smooth and Strongly Convex Brenier Potentials in Optimal Transport". In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by S. Chiappa and R. Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1222–1232.
- [PDL+12] M. Parry, A. P. Dawid, S. Lauritzen, et al. "Proper local scoring rules". In: *The Annals of Statistics* 40.1 (2012), pp. 561–592.
- [PE16] V. Papyan and M. Elad. "Multi-Scale Patch-Based Image Restoration". In: *IEEE Trans. Image Process.* 25.1 (Jan. 2016), pp. 249–261.
- [Pea09a] J. Pearl. *Causality*. 2nd. Cambridge University Press, 2009.
- [Pea09b] J. Pearl. *Causality: Models, Reasoning and Inference (Second Edition)*. Cambridge Univ. Press, 2009.
- [Pea09c] J. Pearl. "Causal inference in statistics: An overview". In: *Stat. Surv.* 3.0 (2009), pp. 96–146.
- [Pea12] J. Pearl. "The Causal Foundations of Structural Equation Modeling". In: *Handbook of structural equation modeling*. Ed. by R. H. Hoyle. Vol. 68. 2012.
- [Pea19] J. Pearl. "The Seven Tools of Causal Inference, with Reflections on Machine Learning". In: *Comm. of the ACM* 62.3 (Mar. 2019), pp. 54–60.
- [Pea36] K. Pearson. "Method of moments and method of maximum likelihood". In: *Biometrika* 28.1/2 (1936), pp. 34–59.
- [Pea84] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1984.
- [Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
- [Pea94] B. A. Pearlmutter. "Fast Exact Multiplication by the Hessian". In: *Neural Comput.* 6.1 (Jan. 1994), pp. 147–160.
- [Peh+20] R. Peherz, S. Lang, A. Vergari, K. Stelzner, A. Molina, M. Trapp, G. Van den Broeck, K. Kersting, and Z. Ghahramani. "Ensum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits". In: (Apr. 2020). arXiv: 2004.06231 [cs.LG].
- [Pen13] J. Pena. "Reading dependencies from covariance graphs". In: *Intl. J. of Approximate Reasoning* 54.1 (2013).
- [Per+18] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville. "FiLM: Visual Reasoning with a General Conditioning Layer". In: *AAAI*. 2018.
- [Pes+21] H. Pesonen et al. "ABC of the Future". In: (Dec. 2021). arXiv: 2112.12841 [stat.AP].
- [Pey20] G. Peyre. "Course notes on Optimization for Machine Learning". 2020.
- [PF03] G. V. Puskorius and L. A. Feldkamp. "Parameter-based Kalman filter training: Theory and implementation". In: *Kalman Filtering and Neural Networks*. New York, USA: John Wiley & Sons, Inc., 2003, pp. 23–67.
- [PF91] G. V. Puskorius and L. A. Feldkamp. "Decoupled extended Kalman filter training of feedforward layered networks". In: *International Joint Conference on Neural Networks*. Vol. i. July 1991, 771–777 vol. 1.
- [PFW21] R. Prado, M. Ferreira, and M. West. *Time Series: Modelling, Computation and Inference (2nd ed.)*. CRC Press, 2021.
- [PG98] M. Popescu and P. D. Gader. "Image content retrieval from image databases using feature integration by Choquet integral". In: *Storage and Retrieval for Image and Video Databases VII*. Ed. by M. Yeung, B.-L. Yeo, and C. A. Bouman. Vol. 3656. International Society for Optics and Photonics. SPIE, 1998, pp. 552–560.
- [PGJ16] J. Pearl, M. Glymour, and N. Jewell. *Causal inference in statistics: a primer*. Wiley, 2016.
- [PHD19] M. F. Pradier, M. C. Hughes, and F. Doshi-Velez. "Challenges in Computing and Optimizing Upper Bounds of Marginal Likelihood based on Chi-Square Divergences". In: *AABI Symposium*. 2019.
- [Phu+18] M. Phuong, M. Welling, N. Kushman, R. Tomioka, and N. Nowozin. "The Mutual Autoencoder: Controlling Information in Latent Code Representations". In: *Arxiv* (2018).
- [Pir+13] M. Pirotta, M. Restelli, A. Pecorino, and D. Calandriello. "Safe Policy Iteration". In: *ICML*. 3. 2013, pp. 307–317.
- [PJS17] J. Peters, D. Janzing, and B. Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms (Adaptive Computation and Machine Learning series)*. The MIT Press, Nov. 2017.
- [PKP21] A. PLAAT, W. KOSTERS, and M. PREUSS. "High-Accuracy Model-Based Reinforcement Learning, a Survey". In: (July 2021). arXiv: 2107.08241 [cs.LG].
- [PL03] M. A. Paskin and G. D. Lawrence. *Junction Tree Algorithms for Solving Sparse Linear Systems*. Tech. rep. UCB/CSD-03-1271. UC Berkeley, 2003.
- [Pla00] J. Platt. "Probabilities for SV machines". In: *Advances in Large Margin Classifiers*. Ed. by A. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans. MIT Press, 2000.
- [Ple+18] G. Pleiss, J. R. Gardner, K. Q. Weinberger, and A. G. Wilson. "Constant-Time Predictive Distributions for Gaussian Processes". In: *International Conference on Machine Learning*. 2018.

- 1
- 2 [Plu+20] G. Plum, M. Al-Shedivat, Á. A. Cabrera, A. Perer, E. Xing, and A. Talwalkar. "Regularizing black-box models for improved interpretability". In: *Advances in Neural Information Processing Systems* 33 (2020).
- 3
- 4 [PM18a] N. Papernot and P. McDaniel. *Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning*. 2018. arXiv: 1803.04765 [cs.LG].
- 5 [PM18b] J. Pearl and D. Mackenzie. *The book of why: the new science of cause and effect*. 2018.
- 6 [PMT18] G. Plum, D. Molitor, and A. Talwalkar. "Supervised Local Modeling for Interpretability". In: *CoRR* abs/1807.02910 (2018). arXiv: 1807.02910.
- 7
- 8 [Pol+19] A. A. Pol, V. Berger, G. Cerminala, C. Germain, and M. Pierini. "Anomaly Detection With Conditional Variational Autoencoders". In: *IEEE International Conference on Machine Learning and Applications*. 2019.
- 9
- 10 [Pom89] D. Pomerleau. "ALVINN: An Autonomous Land Vehicle in a Neural Network". In: *NIPS*. 1989, pp. 305–313.
- 11
- 12 [Poo+19] B. Poole, S. Ozair, A. van den Oord, A. A. Alemi, and G. Tucker. "On variational lower bounds of mutual information". In: *ICML*. 2019.
- 13 [Pou04] M. Pourahmadi. *Simultaneous Modelling of Covariance Matrices: GLM, Bayesian and Nonparametric Perspectives*. Tech. rep. Northern Illinois University, 2004.
- 14
- 15 [Pou+20] F. Pourpanah, M. Abdar, Y. Luo, X. Zhou, R. Wang, C. P. Lim, and X.-Z. Wang. "A Review of Generalized Zero-Shot Learning Methods". In: (Nov. 2020). arXiv: 2011.08641 [cs.CV].
- 16
- 17 [Poy+20] R. Poyiadzi, K. Sokol, R. Santos-Rodriguez, T. De Bie, and P. Flach. "FACE: Feasible and actionable counterfactual explanations". In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 2020, pp. 344–350.
- 18
- 19 [PPC09] G. Petris, S. Petrone, and P. Campagnoli. *Dynamic linear models with R*. Springer, 2009.
- 20 [PPG91] C. S. Pomerleau, O. F. Pomerleau, and A. W. Garcia. "Biobehavioral research on nicotine use in women". In: *British Journal of Addiction* 86.5 (1991), pp. 527–531.
- 21 [PPM17] G. Papamakarios, T. Pavlakou, and I. Murray. "Masked Autoregressive Flow for Density Estimation". In: *NIPS*. 2017.
- 22
- 23 [PPS18] T. Pierrot, N. Perrin, and O. Sigaud. "First-order and second-order variants of the gradient descent in a unified framework". In: (Oct. 2018). arXiv: 1810.08102 [cs.LG].
- 24
- 25 [PR03] O. Papaspiliopoulos and G. O. Roberts. "Non-Centered Parameterisations for Hierarchical Models and Data Augmentation". In: *Bayesian Statistics* 7 (2003), pp. 307–326.
- 26
- 27 [Pra+18] S. Prabhunov, Y. Tsvetkov, R. Salakhutdinov, and A. W. Black. "Style Transfer Through Back-Translation". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 866–876.
- 28
- 29 [Pre05] S. J. Press. *Applied multivariate analysis, using Bayesian and frequentist methods of inference*. Second edition. Dover, 2005.
- 30 [Pre+17] O. Press, A. Bar, B. Bogin, J. Berant, and L. Wolf. "Language generation with recurrent generative adversarial networks without pre-training". In: *arXiv preprint arXiv:1706.01399* (2017).
- 31
- 32 [Pre+88] W. Press, W. Vetterling, S. Teukolosky, and B. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Second. Cambridge University Press, 1988.
- 33
- 34 [Pri58] R. Price. "A useful theorem for nonlinear devices having Gaussian inputs". In: *IRE Trans. Info. Theory* 4.2 (1958), pp. 69–72.
- 35 [PS07] J. Peters and S. Schaal. "Reinforcement Learning by Reward-Weighted Regression for Operational Space Control". In: *ICML*. 2007, pp. 745–750.
- 36
- 37 [PS08a] B. A. Pearlmutter and J. M. Siskind. "Reverse-Mode AD in a Functional Framework: Lambda the Ultimate Back-propagator". In: *ACM Trans. Program. Lang. Syst.* 30.2 (Mar. 2008).
- 38
- 39 [PS08b] J. Peters and S. Schaal. "Reinforcement Learning of Motor Skills with Policy Gradients". In: *Neural Networks* 21.4 (2008), pp. 682–697.
- 40
- 41 [PS12] N. G. Polson and J. G. Scott. "On the Half-Cauchy Prior for a Global Scale Parameter". en. In: *Bayesian Anal.* 7.4 (Dec. 2012), pp. 887–902.
- 42
- 43 [PS17] N. G. Polson and V. Sokolov. "Deep Learning: A Bayesian Perspective". en. In: *Bayesian Anal.* 12.4 (Dec. 2017), pp. 1275–1304.
- 44
- 45 [PSCD20] I. Paris, R. Sánchez-Caice, and F. J. Díez. "Sum-product networks: A survey". In: (Apr. 2020). arXiv: 2004.01167 [cs.LG].
- 46
- 47
- [PSD00] J. K. Pritchard, M. Stephens, and P. Donnelly. "Inference of population structure using multilocus genotype data". In: *Genetics* 155.2 (June 2000), pp. 945–959.
- [PSM19] G. Papamakarios, D. Sterratt, and I. Murray. "Sequential Neural Likelihood: Fast Likelihood-free Inference with Autoregressive Flows". In: *AISTATS*. 2019.
- [PSS00] D. Precup, R. S. Sutton, and S. P. Singh. "Eligibility Traces for Off-Policy Policy Evaluation". In: *ICML*. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 759–766.
- [PT13] S. Patterson and Y. W. Teh. "Stochastic Gradient Riemannian Langevin Dynamics on the Probability Simplex". In: *NIPS*. 2013.
- [PT87] C. Papadimitriou and J. Tsitsiklis. "The complexity of Markov decision processes". In: *Mathematics of Operations Research* 12.3 (1987), pp. 441–450.
- [PT94] P. Paatero and U. Tapper. "Positive Matrix Factorization: A Non-negative Factor Model with Optimal Utilization of Error Estimates of Data Values". In: *Environmetrics* 5 (1994), pp. 111–126.
- [PTD20] A. Prabhu, P. H. S. Torr, and P. K. Dokania. "GDumb: A simple approach that questions our progress in continual learning". In: *ECCV*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 524–540.
- [Put94] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.
- [PVC19] R. Prenger, R. Valle, and B. Catanzaro. "WaveGLOW: A Flow-based generative network for speech synthesis". In: *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2019, pp. 3617–3621.
- [PVP18] A. Pesaranghader, H. Viktor, and E. Paquet. "McDiarmid Drift Detection Methods for Evolving Data Streams". In: *ICJNN*. 2018.
- [PW05] S. Parise and M. Welling. "Learning in Markov Random Fields: An Empirical Study". In: *Joint Statistical Meeting*. 2005.
- [PW16] B. Paige and F. Wood. "Inference Networks for Sequential Monte Carlo in Graphical Models". In: *ICML*. Feb. 2016.
- [PY97] J. Pitman and M. Yor. "The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator". In: *The Annals of Probability* (1997), pp. 855–900.
- [QC+06] J. Quiñonero-Candela, C. E. Rasmussen, F. Sinz, O. Bouquet, and B. Schölkopf. "Evaluating Predictive Uncertainty Challenge". In: *Machine Learning Challenges: Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 1–27.
- [QC+08] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, eds. *Dataset Shift in Machine Learning (Neural Information Processing series)*. en. Illustrated edition. The MIT Press, Dec. 2008.
- [QCR05] J. Quiñonero-Candela and C. Rasmussen. "A unifying view of sparse approximate Gaussian process regression". In: *JMLR* 6.3 (2005), pp. 1939–1959.
- [Qin+20] C. Qin, Y. Wu, J. T. Springenberg, A. Brock, J. Donahue, T. P. Lillicrap, and P. Kohli. "Training Generative Adversarial Networks by Solving Ordinary Differential Equations". In: *arXiv preprint arXiv:2010.15040* (2020).
- [Qu+21] H. Qu, H. Rahmani, L. Xu, B. Williams, and J. Liu. "Recent Advances of Continual Learning in Computer Vision: An Overview". In: (Sept. 2021). arXiv: 2109.11369 [cs.CV].
- [Qua+07] A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell. "Hidden conditional random fields". In: *IEEE PAMI* 29.10 (2007), pp. 1848–1852.
- [Que98] M. Queyranne. "Minimizing symmetric submodular functions". In: *Math. Programming* 82 (1998), pp. 3–12.
- [QZW19] Y. Qiu, L. Zhang, and X. Wang. "Unbiased Contrastive Divergence Algorithm for Training Energy-Based Latent Variable Models". In: *International Conference on Learning Representations*. 2019.
- [RA13] O. Rippel and R. P. Adams. "High-dimensional probability estimation with deep density models". In: *ArXiv Preprint arXiv:1302.5125* (2013).
- [Rab89] L. R. Rabiner. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition". In: *Proc. of the IEEE* 77.2 (1989), pp. 257–286.
- [Rad+18] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. *Improving Language Understanding by Generative Pre-Training*. Tech. rep. OpenAI, 2018.
- [Rad+19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. *Language Models are Unsupervised Multitask Learners*. Tech. rep. OpenAI, 2019.
- [Raf+19] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: (Oct. 2019). arXiv: 1910.10683 [cs.LG].

- 1 [RAG04] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House Radar Library, 2004.
- 2 [Rai+18] T. Rainforth, A. R. Kosiorek, T. A. Le, C. J. Maddison, M. Igl, F. Wood, and Y. W. Teh. "Tighter Variational Bounds are Not Necessarily Better". In: *ICML* 2018.
- 3 [Rai+20] T. Rainforth, A. Golinski, F. Wood, and S. Zaidi. "Target-Aware Bayesian Inference: How to Beat Optimal Conventional Estimators". In: *JMLR* 21.88 (2020), pp. 1–54.
- 4 [Rai68] H. Raiffa. *Decision Analysis*. Addison Wesley, 1968.
- 5 [Rak+08] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. "SimpleMKL". In: *JMLR* 9 (2008), pp. 2491–2521.
- 6 [Ram+21] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. "Zero-Shot Text-to-Image Generation". In: (Feb. 2021). arXiv: 2102.12092 [cs.CV].
- 7 [Ran16] R. Ranganath. "Hierarchical Variational Models". In: *ICML* 2016.
- 8 [Ran+18] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. "Deep State Space Models for Time Series Forecasting". In: *NIPS*. Curran Associates, Inc., 2018, pp. 7796–7805.
- 9 [Rao10] A. V. Rao. "A Survey of Numerical Methods for Optimal Control". In: *Adv. Astronaut. Sci.* 135.1 (Jan. 2010).
- 10 [Rao99] R. P. Rao. "An optimal estimation approach to visual perception and learning". en. In: *Vision Res.* 39.11 (June 1999), pp. 1963–1989.
- 11 [Ras00] C. Rasmussen. "The Infinite Gaussian Mixture Model". In: *NIPS*. 2000.
- 12 [Rat+09] M. Ratray, O. Stegle, K. Sharp, and J. Winn. "Inference algorithms and learning theory for Bayesian sparse factor analysis". In: *Proc. Intl. Workshop on Statistical-Mechanical Informatics*, 2009.
- 13 [Ray+18] S. Ravuri, S. Mohamed, M. Rosca, and O. Vinyals. "Learning Implicit Generative Models with the Method of Learned Moments". In: *International Conference on Machine Learning*, 2018, pp. 4314–4323.
- 14 [RB16] G. P. Rigby BR. "The Efficacy of Equine-Assisted Activities and Therapies on Improving Physical Function." In: *J Altern Complement Med.* (2016).
- 15 [RB99] R. P. Rao and D. H. Ballard. "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects". en. In: *Nat. Neurosci.* 2.1 (Jan. 1999), pp. 79–87.
- 16 [RBB18a] H. Ritter, A. Botev, and D. Barber. "A Scalable Laplace Approximation for Neural Networks". In: *ICLR* 2018.
- 17 [RBB18b] H. Ritter, A. Botev, and D. Barber. "Online Structured Laplace Approximations for Overcoming Catastrophic Forgetting". In: *NIPS*. Curran Associates, Inc., 2018, pp. 3738–3748.
- 18 [RBS16] L. J. Ratliff, S. A. Burden, and S. S. Sastry. "On the characterization of local Nash equilibria in continuous games". In: *IEEE transactions on automatic control* 61.8 (2016), pp. 2301–2307.
- 19 [RC04] C. Robert and G. Casella. *Monte Carlo Statistical Methods*. 2nd edition. Springer, 2004.
- 20 [RC+18] M. Rojas-Carulla, B. Schölkopf, R. Turner, and J. Peters. "Invariant Models for Causal Transfer Learning". In: *Journal of Machine Learning Research* 19.36 (2018), pp. 1–34.
- 21 [RD06] M. Richardson and P. Domingos. "Markov logic networks". In: *Machine Learning* 62 (2006), pp. 107–136.
- 22 [RDL21] O. Rybkin, K. Daniilidis, and S. Levine. "Simple and Effective VAE Training with Calibrated Decoders". In: *ICML*. 2021.
- 23 [RDLV18] A. S. Ross and F. Doshi-Velez. "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients". In: *Thirty-second AAAI conference on artificial intelligence*. 2018.
- 24 [Rec19] B. Recht. "A Tour of Reinforcement Learning: The View from Continuous Control". In: *Annual Review of Control, Robotics, and Autonomous Systems* 2 (2019), pp. 253–279.
- 25 [Ree+17] S. Reed, A. van den Oord, N. Kalchbrenner, S. G. Colmenarejo, Z. Wang, D. Belov, and N. de Freitas. "Parallel Multiscale Autoregressive Density Estimation". In: (2017). arXiv: 1703.03664 [cs.CV].
- 26 [Rei+10] J. Reisinger, A. Waters, B. Silverthorn, and R. Mooney. "Spherical topic models". In: *ICML*. 2010.
- 27 [Rei13] S. Reich. "A Nonparametric Ensemble Transform Method for Bayesian Inference". In: *SIAM J. Sci. Comput.* 35.4 (Jan. 2013), A2013–A2024.
- 28 [Rei16] P. C. Reiss. "Just How Sensitive are Instrumental Variable Estimates?" In: *Foundations and Trends in Accounting* 10.2-4 (2016).
- 29 [Ren+19] J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. A. DePristo, J. V. Dillon, and B. Lakshminarayanan. "Likelihood Ratios for Out-of-Distribution Detection". In: *NIPS*. 2019.
- 30 [Rén61] A. Rényi. "On Measures of Entropy and Information". en. In: *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. The Regents of the University of California, 1961.
- 31 [RF96] J. J. K. Ruanaidh and W. J. Fitzgerald. *Numerical Bayesian Methods Applied to Signal Processing*. en. 1996th ed. Springer, Feb. 1996.
- 32 [RG17] M Roth and F Gustafsson. "Computation and visualization of posterior densities in scalar nonlinear and non-Gaussian Bayesian filtering and smoothing problems". In: *ICASSP*. Mar. 2017, pp. 4686–4690.
- 33 [RGB11] S. Ross, G. J. Gordon, and D. Bagnell. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning". In: *AISTATS*. 2011, pp. 627–635.
- 34 [RGR14] R. Ranganath, S. Gerrish, and D. M. Blei. "Black Box Variational Inference". In: *AISTATS*. 2014.
- 35 [RGL19] S. Rabanser, S. Günnemann, and Z. C. Lipton. "Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift". In: *NIPS*. 2019.
- 36 [RH05] H. Rue and L. Held. *Gaussian Markov Random Fields: Theory and Applications*. Vol. 104. Monographs on Statistics and Applied Probability. London: Chapman & Hall, 2005.
- 37 [RHDV17] A. S. Ross, M. C. Hughes, and F. Doshi-Velez. "Right for the right reasons: Training differentiable models by constraining their explanations". In: *IJCAI* (2017).
- 38 [RHG16a] D. Ritchie, P. Horsfall, and N. D. Goodman. "Deep Amortized Inference for Probabilistic Programs". In: (Oct. 2016). arXiv: 1610.05735 [cs.AI].
- 39 [RHG16b] M. Roth, G. Hendeby, and F. Gustafsson. "Nonlinear Kalman Filters Explained: A Tutorial on Moment Computations and Sigma Point Methods". In: *J. Advanced Information Fusion* (2016).
- 40 [RHW86a] D. Rumelhart, G. Hinton, and R. Williams. "Learning internal representations by error propagation". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Ed. by D. Rumelhart, J. McClelland, and the PDD Research Group. MIT Press, 1986.
- 41 [RHW86b] D. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536.
- 42 [Ric03] T. Richardson. "Markov properties for acyclic directed mixed graphs". In: *Scandinavian J. of Statistics* 30 (2003), pp. 145–157.
- 43 [Ric95] J. Rice. *Mathematical statistics and data analysis*. 2nd edition. Duxbury, 1995.
- 44 [Rip05] B. D. Ripley. *Spatial statistics*. Vol. 575. John Wiley & Sons, 2005.
- 45 [Rip77] B. D. Ripley. "Modelling spatial patterns". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.2 (1977), pp. 172–192.
- 46 [Ris+08] I. Rish, G. Grabarnik, G. Cecchi, F. Pereira, and G. Gordon. "Closed-form supervised dimensionality reduction with generalized linear models". In: *ICML*. 2008.
- 47 [Riv87] R. L. Rivest. "Learning decision lists". In: *Machine learning* 2.3 (1987), pp. 229–246.
- 48 [RK04] R. Rifkin and A. Klautau. "In defense of One-Vs-All classification". In: *JMLR* 5 (2004), pp. 101–141.
- 49 [RL17] S. Ravi and H. Larochelle. "Optimization as a Model for Few-Shot Learning". In: *ICLR*. 2017.
- 50 [RM15a] G. Raskutti and S. Mukherjee. "The information geometry of mirror descent". In: *IEEE Trans. Info. Theory* 61.3 (2015), pp. 1451–1457.
- 51 [RM15b] D. J. Rezende and S. Mohamed. "Variational Inference with Normalizing Flows". In: *ICML*. 2015.
- 52 [RMB08] N. L. Roux, P.-A. Manzagol, and Y. Bengio. "Top-moumoute Online Natural Gradient Algorithm". In: *NIPS*. 2008, pp. 849–856.
- 53 [RM09] H. Rue, S. Martino, and N. Chopin. "Approximate Bayesian Inference for Latent Gaussian Models Using Integrated Nested Laplace Approximations". In: *J. of Royal Stat. Soc. Series B* 71 (2009), pp. 319–392.
- 54 [RMC15] A. Radford, L. Metz, and S. Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *arXiv* (2015).
- 55 [RM16] A. Radford, L. Metz, and S. Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *ICLR*. 2016.
- 56 [RMW14a] D. Rezende, S. Mohamed, and D. Wierstra. "Stochastic backpropagation and approximate inference in deep generative models". In: *ICML*. 2014.

- 1
- 2 [RMW14b] D. J. Rezende, S. Mohamed, and D. Wierstra. "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *ICML*. Ed. by E. P. Xing and T. Jebara. Vol. 32. Proceedings of Machine Learning Research. Beijing, China: PMLR, 2014, pp. 1278–1286.
- 3
- 4 [RN02] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. 2nd edition. Prentice Hall, 2002.
- 5
- 6 [RN10] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. 3rd edition. Prentice Hall, 2010.
- 7
- 8 [RN19] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. 4th edition. Prentice Hall, 2019.
- 9
- 10 [RN94] G. A. Rummery and M. Niranjan. *On-Line Q-Learning Using Connectionist Systems*. Tech. rep. Cambridge Univ. Engineering Dept., 1994.
- 11
- 12 [RN95] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: Prentice Hall, 1995.
- 13
- 14 [Rob07] C. P. Robert. *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*. en. 2nd edition. Springer Verlag, New York, 2007.
- 15
- 16 [Rob+13] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aligrain. "Gaussian processes for time-series modelling". en. In: *Philos. Trans. A Math. Phys. Eng. Sci.* 371.1984 (Feb. 2013), p. 20110550.
- 17
- 18 [Rob+18] C. P. Robert, V. Elvira, N. Tawn, and C. Wu. "Accelerating MCMC Algorithms". In: (Apr. 2018). arXiv: 1804.02719 [stat.CO].
- 19
- 20 [Rob86] J. Robins. "A new approach to causal inference in mortality studies with a sustained exposure period—application to control of the healthy worker survivor effect". In: *Mathematical Modelling* 7.9 (1986), pp. 1393–1512.
- 21
- 22 [Rob95a] C. Robert. "Simulation of truncated normal distributions". In: *Statistics and computing* 5 (1995), pp. 121–125.
- 23
- 24 [Rob95b] A. Robins. "Catastrophic Forgetting, Rehearsal and Pseudorehearsal". In: *Conn. Sci.* 7.2 (June 1995), pp. 123–146.
- 25
- 26 [Rod14] J. Rodu. "Spectral estimation of hidden Markov models". PhD thesis. U. Penn, 2014.
- 27
- 28 [RÖG13] M. Roth, E. Özkan, and F. Gustafsson. "A Student's t filter for multi-tailed process and measurement noise". In: *ICASSP*. May 2013, pp. 5770–5774.
- 29
- 30 [Roh21] D. Rohde. "Causal Inference, is just Inference: A beautifully simple idea that not everyone accepts". In: *I (Still) Can't Believe It's Not Better! NeurIPS 2021 Workshop*. Sept. 2021.
- 31
- 32 [Ros10] P. Rosenbaum. *Design of Observational Studies*. 2010.
- 33
- 34 [Ros+21] M. Rosca, Y. Wu, B. Dherin, and D. G. Barrett. "Discretization Drift in Two-Player Games". In: (2021).
- 35
- 36 [Ros22] C. Ross. *AI gone astray: How subtle shifts in patient data send popular algorithms reeling, undermining patient safety*. en. <https://www.statnews.com/2022/02/28/sepsis-hospital-algorithms-data-shift/>. Accessed: 2022-3-2. Feb. 2022.
- 37
- 38 [Rot+17] M. Roth, G. Hendey, C. Fritzsche, and F. Gustafsson. "The Ensemble Kalman filter: a signal processing perspective". In: *EURASIP J. Adv. Signal Processing* 2017.1 (Aug. 2017), p. 56.
- 39
- 40 [Rot96] D. Roth. "On the hardness of approximate reasoning". In: *Artificial Intelligence* 82.1-2 (1996), pp. 273–302.
- 41
- 42 [ROV19] A. Razavi, A. van den Oord, and O. Vinyals. "Generating diverse high resolution images with VA-VAE-2". In: *NIPS*. 2019.
- 43
- 44 [Row97] S. Roweis. "EM algorithms for PCA and SPCA". In: *NIPS*. 1997.
- 45
- 46 [RPC19] Y. Romano, E. Patterson, and E. J. Candès. "Conformalized Quantile Regression". In: *NIPS*. May 2019.
- 47
- 48 [RPH21] A. Robey, G. J. Pappas, and H. Hassani. *Model-Based Domain Generalization*. 2021. arXiv: 2102.11436 [stat.ML].
- 49
- 50 [RRO1a] A. Rao and K. Rose. "Deterministically Annealed Design of Hidden Markov Model Speech Recognizers". In: *IEEE Trans. on Speech and Audio Proc.* 9.2 (2001), pp. 111–126.
- 51
- 52 [RR01b] G. Roberts and J. Rosenthal. "Optimal scaling for various Metropolis-Hastings algorithms". In: *Statistical Science* 16 (2001), pp. 351–367.
- 53
- 54 [RR08] A. Rahimi and B. Recht. "Random Features for Large-Scale Kernel Machines". In: *NIPS*. Curran Associates, Inc., 2008, pp. 1177–1184.
- 55
- 56 [RR09] A. Rahimi and B. Recht. "Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning". In: *NIPS*. Curran Associates, Inc., 2009, pp. 1313–1320.
- 57
- 58 [RR11] T. S. Richardson and J. M. Robins. "Single World Intervention Graphs: A Primer". In: *Second UAI workshop on causal structure learning*. 2011.
- 59
- 60
- 61
- 62
- 63
- 64
- 65
- 66
- 67
- 68
- 69
- 70
- 71
- 72
- 73
- 74
- 75
- 76
- 77
- 78
- 79
- 80
- 81
- 82
- 83
- 84
- 85
- 86
- 87
- 88
- 89
- 90
- 91
- 92
- 93
- 94
- 95
- 96
- 97
- 98
- 99
- 100
- 101
- 102
- 103
- 104
- 105
- 106
- 107
- 108
- 109
- 110
- 111
- 112
- 113
- 114
- 115
- 116
- 117
- 118
- 119
- 120
- 121
- 122
- 123
- 124
- 125
- 126
- 127
- 128
- 129
- 130
- 131
- 132
- 133
- 134
- 135
- 136
- 137
- 138
- 139
- 140
- 141
- 142
- 143
- 144
- 145
- 146
- 147
- 148
- 149
- 150
- 151
- 152
- 153
- 154
- 155
- 156
- 157
- 158
- 159
- 160
- 161
- 162
- 163
- 164
- 165
- 166
- 167
- 168
- 169
- 170
- 171
- 172
- 173
- 174
- 175
- 176
- 177
- 178
- 179
- 180
- 181
- 182
- 183
- 184
- 185
- 186
- 187
- 188
- 189
- 190
- 191
- 192
- 193
- 194
- 195
- 196
- 197
- 198
- 199
- 200
- 201
- 202
- 203
- 204
- 205
- 206
- 207
- 208
- 209
- 210
- 211
- 212
- 213
- 214
- 215
- 216
- 217
- 218
- 219
- 220
- 221
- 222
- 223
- 224
- 225
- 226
- 227
- 228
- 229
- 230
- 231
- 232
- 233
- 234
- 235
- 236
- 237
- 238
- 239
- 240
- 241
- 242
- 243
- 244
- 245
- 246
- 247
- 248
- 249
- 250
- 251
- 252
- 253
- 254
- 255
- 256
- 257
- 258
- 259
- 260
- 261
- 262
- 263
- 264
- 265
- 266
- 267
- 268
- 269
- 270
- 271
- 272
- 273
- 274
- 275
- 276
- 277
- 278
- 279
- 280
- 281
- 282
- 283
- 284
- 285
- 286
- 287
- 288
- 289
- 290
- 291
- 292
- 293
- 294
- 295
- 296
- 297
- 298
- 299
- 300
- 301
- 302
- 303
- 304
- 305
- 306
- 307
- 308
- 309
- 310
- 311
- 312
- 313
- 314
- 315
- 316
- 317
- 318
- 319
- 320
- 321
- 322
- 323
- 324
- 325
- 326
- 327
- 328
- 329
- 330
- 331
- 332
- 333
- 334
- 335
- 336
- 337
- 338
- 339
- 340
- 341
- 342
- 343
- 344
- 345
- 346
- 347
- 348
- 349
- 350
- 351
- 352
- 353
- 354
- 355
- 356
- 357
- 358
- 359
- 360
- 361
- 362
- 363
- 364
- 365
- 366
- 367
- 368
- 369
- 370
- 371
- 372
- 373
- 374
- 375
- 376
- 377
- 378
- 379
- 380
- 381
- 382
- 383
- 384
- 385
- 386
- 387
- 388
- 389
- 390
- 391
- 392
- 393
- 394
- 395
- 396
- 397
- 398
- 399
- 400
- 401
- 402
- 403
- 404
- 405
- 406
- 407
- 408
- 409
- 410
- 411
- 412
- 413
- 414
- 415
- 416
- 417
- 418
- 419
- 420
- 421
- 422
- 423
- 424
- 425
- 426
- 427
- 428
- 429
- 430
- 431
- 432
- 433
- 434
- 435
- 436
- 437
- 438
- 439
- 440
- 441
- 442
- 443
- 444
- 445
- 446
- 447
- 448
- 449
- 450
- 451
- 452
- 453
- 454
- 455
- 456
- 457
- 458
- 459
- 460
- 461
- 462
- 463
- 464
- 465
- 466
- 467
- 468
- 469
- 470
- 471
- 472
- 473
- 474
- 475
- 476
- 477
- 478
- 479
- 480
- 481
- 482
- 483
- 484
- 485
- 486
- 487
- 488
- 489
- 490
- 491
- 492
- 493
- 494
- 495
- 496
- 497
- 498
- 499
- 500
- 501
- 502
- 503
- 504
- 505
- 506
- 507
- 508
- 509
- 510
- 511
- 512
- 513
- 514
- 515
- 516
- 517
- 518
- 519
- 520
- 521
- 522
- 523
- 524
- 525
- 526
- 527
- 528
- 529
- 530
- 531
- 532
- 533
- 534
- 535
- 536
- 537
- 538
- 539
- 540
- 541
- 542
- 543
- 544
- 545
- 546
- 547
- 548
- 549
- 550
- 551
- 552
- 553
- 554
- 555
- 556
- 557
- 558
- 559
- 560
- 561
- 562
- 563
- 564
- 565
- 566
- 567
- 568
- 569
- 570
- 571
- 572
- 573
- 574
- 575
- 576
- 577
- 578
- 579
- 580
- 581
- 582
- 583
- 584
- 585
- 586
- 587
- 588
- 589
- 590
- 591
- 592
- 593
- 594
- 595
- 596
- 597
- 598
- 599
- 600
- 601
- 602
- 603
- 604
- 605
- 606
- 607
- 608
- 609
- 610
- 611
- 612
- 613
- 614
- 615
- 616
- 617
- 618
- 619
- 620
- 621
- 622
- 623
- 624
- 625
- 626
- 627
- 628
- 629
- 630
- 631
- 632
- 633
- 634
- 635
- 636
- 637
- 638
- 639
- 640
- 641
- 642
- 643
- 644
- 645
- 646
- 647
- 648
- 649
- 650
- 651
- 652
- 653
- 654
- 655
- 656
- 657
- 658
- 659
- 660
- 661
- 662
- 663
- 664
- 665
- 666
- 667
- 668
- 669
- 670
- 671
- 672
- 673
- 674
- 675
- 676
- 677
- 678
- 679
- 680
- 681
- 682
- 683
- 684
- 685
- 686
- 687
- 688
- 689
- 690
- 691
- 692
- 693
- 694
- 695
- 696
- 697
- 698
- 699
- 700
- 701
- 702
- 703
- 704
- 705
- 706
- 707
- 708
- 709
- 710
- 711
- 712
- 713
- 714
- 715
- 716
- 717
- 718
- 719
- 720
- 721
- 722
- 723
- 724
- 725
- 726
- 727
- 728
- 729
- 730
- 731
- 732
- 733
- 734
- 735
- 736
- 737
- 738
- 739
- 740
- 741
- 742
- 743
- 744
- 745
- 746
- 747
- 748
- 749
- 750
- 751
- 752
- 753
- 754
- 755
- 756
- 757
- 758
- 759
- 760
- 761
- 762
- 763
- 764
- 765
- 766
- 767
- 768
- 769
- 770
- 771
- 772
- 773
- 774
- 775
- 776
- 777
- 778
- 779
- 780
- 781
- 782
- 783
- 784
- 785
- 786
- 787
- 788
- 789
- 790
- 791
- 792
- 793
- 794
- 795
- 796
- 797
- 798
- 799
- 800
- 801
- 802
- 803
- 804
- 805
- 806
- 807
- 808
- 809
- 810
- 811
- 812
- 813
- 814
- 815
- 816
- 817
- 818
- 819
- 820
- 821
- 822
- 823
- 824
- 825
- 826
- 827
- 828
- 829
- 830
- 831
- 832
- 833
- 834
- 835
- 836
- 837
- 838
- 839
- 840
- 841
- 842
- 843
- 844
- 845
- 846
- 847
- 848
- 849
- 850
- 851
- 852
- 853
- 854
- 855
- 856
- 857
- 858
- 859
- 860
- 861
- 862
- 863
- 864
- 865
- 866
- 867
- 868
- 869
- 870
- 871
- 872
- 873
- 874
- 875
- 876
- 877
- 878
- 879
- 880
- 881
- 882
- 883
- 884
- 885
- 886
- 887
- 888</p

- 1
- 2 *dations and Trends in Machine Learning* 11.1 (2018), pp. 1–96.
- 3 [Rus+95] S. Russell, J. Binder, D. Koller, and K. Kanazawa. “Local learning in probabilistic networks with hidden variables”. In: *IJCAI*. 1995.
- 4 [RV19] S. Ravuri and O. Vinyals. “Classification accuracy score for conditional generative models”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 12268–12279.
- 5 [RW06] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.
- 6 [RW11] M. D. Reid and R. C. Williamson. “Information, Divergence and Risk for Binary Experiments”. In: *Journal of Machine Learning Research* 12.3 (2011).
- 7 [RW15] D. Rosenbaum and Y. Weiss. “The Return of the Gating Network: Combining Generative Models and Discriminative Training in Natural Image Priors”. In: *NIPS*. 2015, pp. 2665–2673.
- 8 [RW18] E. Richardson and Y. Weiss. “On GANs and GMMs”. In: *NIPS*. 2018.
- 9 [RWD17] G. Roeder, Y. Wu, and D. Duvenaud. “Sticking the Landing: An Asymptotically Zero-Variance Gradient Estimator for Variational Inference”. In: *NIPS*. 2017.
- 10 [RY21] D. Roberts and S. Yaida. *The Principles of Deep Learninging Theory: An Effective Theory Approach to Understanding Neural Network*. 2021.
- 11 [Ryc+19] B. Rychalska, D. Basaj, A. Gosiewska, and P. Biecek. “Models in the Wild: On Corruption Robustness of Neural NLP Systems”. In: *International Conference on Neural Information Processing (ICONIP)*. Springer International Publishing, 2019, pp. 235–247.
- 12 [Ryu+20] M. Ryu, Y. Chow, R. Anderson, C. Tjandraatmadja, and C. Boutilier. “CAQL: Continuous Action Q-Learning”. In: *ICLR*. 2020.
- 13 [RZL17] P. Ramachandran, B. Zoph, and Q. V. Le. “Searching for Activation Functions”. In: (Oct. 2017). arXiv: 1710.05941 [cs.NE].
- 14 [SA19] F. Schafer and A. Anandkumar. “Competitive gradient descent”. In: *NIPS*. 2019, pp. 7625–7635.
- 15 [Sac+05] K. Sachs, O. Perez, D. Pe'er, D. Lauffenburger, and G. Nolan. “Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data”. In: *Science* 308 (2005).
- 16 [SAC17] J. Schulman, P. Abbeel, and X. Chen. *Equivalence Between Policy Gradients and Soft Q-Learning*. arXiv:1704.06440. 2017.
- 17 [Sai+20] M. Saito, S. Saito, M. Koyama, and S. Kobayashi. “Train Sparsely, Generate Densely: Memory-Efficient Unsupervised Training of High-Resolution Temporal GAN”. In: *International Journal of Computer Vision* 128 (2020), pp. 2586–2606.
- 18 [Saj+18] M. S. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. “Assessing generative models via precision and recall”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 2018, pp. 5234–5243.
- 19 [Sal16] T. Salimans. “A Structured Variational Auto-encoder for Learning Deep Hierarchies of Sparse Features”. In: (2016). arXiv: 1602.08734 [stat.ML].
- 20 [Sal+16] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. “Improved Techniques for Training GANs”. In: (2016). arXiv: 1606.03498 [cs.LG].
- 21 [Sal+17a] M. Salehi, A. Karbasi, D. Scheinost, and R. T. Constable. “A Submodular Approach to Create Individualized Parcellations of the Human Brain”. In: *Medical Image Computing and Computer Assisted Intervention - MICCAI 2017*. Ed. by M. Descoteaux, L. Maier-Hein, A. Franz, P. Jannin, D. L. Collins, and S. Duchesne. Cham: Springer International Publishing, 2017, pp. 478–485.
- 22 [Sal+17b] T. Salimans, J. Ho, X. Chen, and I. Sutskever. “Evolution Strategies as a Scalable Alternative to Reinforcement Learning”. In: (Mar. 2017). arXiv: 1703.03864 [stat.ML].
- 23 [Sal+17c] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. “PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications”. In: *ICLR*. 2017.
- 24 [Sal+19a] D. Salinas, M. Bohlke-Schneider, L. Callot, R. Medico, and J. Gasthaus. “High-Dimensional Multivariate Forecasting with Low-Rank Gaussian Copula Processes”. In: *NIPS*. 2019.
- 25 [Sal+19b] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski. “DeepAR: Probabilistic forecasting with autoregressive recurrent networks”. In: *International Journal of Forecasting*. 2019.
- 26 [Sal+20] H. Salman, A. Ilyas, L. Engstrom, A. Kapoor, and A. Madry. “Do Adversarially Robust ImageNet Models Transfer Better?”. In: *arXiv preprint arXiv:2007.08489* (2020).
- 27 [Sal+21] M. Salehi, H. Mirzaei, D. Hendrycks, Y. Li, M. H. Rohban, and M. Sabokrou. “A Unified Survey on Anomaly, Novelty, Open-Set, and Out-of-Distribution Detection: Solutions and Future Challenges”. In: (Oct. 2021). arXiv: 2110.14051 [cs.CV].
- 28 [Sam68] F. Sampson. “A Novitiate in a Period of Change: An Experimental and Case Study of Social Relationships”. PhD thesis. Cornell, 1968.
- 29 [Sam74] P. A. Samuelson. “Complementarity: An essay on the 40th anniversary of the Hicks-Allen revolution in demand theory”. In: *Journal of Economic literature* 12.4 (1974), pp. 1255–1289.
- 30 [San+17] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. “A simple neural network module for relational reasoning”. In: *Advances in neural information processing systems*. 2017, pp. 4967–4976.
- 31 [Sar13] S. Sarkka. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.
- 32 [Sar18] H. Sarin. “Playing a game of GANstruction”. In: *The Gradient* (2018).
- 33 [Sat15] C. Satzinger. “On Tree-Decompositions and their Algorithmic Implications for Bounded-Treewidth Graphs”. PhD thesis. U. Wien, 2015.
- 34 [Say+19] R. Sayres, S. Xu, T. Saensuksoopa, M. Le, and D. R. Webster. “Assistance from a deep learning system improves diabetic retinopathy assessment in optometrists”. In: *Investigative Ophthalmology & Visual Science* 60.9 (2019), pp. 1433–1433.
- 35 [SB01] A. J. Smola and P. L. Bartlett. “Sparse Greedy Gaussian Process Regression”. In: *NIPS*. Ed. by T. K. Leen, T. G. Dietterich, and V. Tresp. MIT Press, 2001, pp. 619–625.
- 36 [SB12] J. Staines and D. Barber. “Variational Optimization”. In: (Dec. 2012). arXiv: 1212.4507 [stat.ML].
- 37 [SB13] J. Staines and D. Barber. “Optimization by Variational Bounding”. In: *European Symposium on ANNs*. elen.ucl.ac.be, 2013.
- 38 [SB18] R. Sutton and A. Barto. *Reinforcement learning: an introduction* (2nd edn). MIT Press, 2018.
- 39 [SBG07] S. Siddiqi, B. Boots, and G. Gordon. “A constraint generation approach to learning stable linear dynamical systems”. In: *NIPS*. 2007.
- 40 [SPB17] Y. Sun, P. Babu, and D. P. Palomar. “Majorization-Minimization Algorithms in Signal Processing, Communications, and Machine Learning”. In: *IEEE Trans. Signal Process*. 65.3 (Feb. 2017), pp. 794–816.
- 41 [SC13] C. Schäfer and N. Chopin. “Sequential Monte Carlo on large binary sampling spaces”. In: *Stat. Comput.* 23.2 (Mar. 2013), pp. 163–184.
- 42 [SC86] R. Smith and P. Cheeseman. “On the Representation and Estimation of Spatial Uncertainty”. In: *Intl. J. Robotics Research* 5.4 (1986), pp. 56–68.
- 43 [SC90] R. Schwarz and Y. Chow. “The n-best algorithm: an efficient and exact procedure for finding the n most likely hypotheses”. In: *ICASSP*. 1990.
- 44 [Scd21] S. Scardapane. *Lecture 8: Beyond single-task supervised learning*. 2021.
- 45 [SCC17] S. Sun, C. Chen, and L. Carin. “Learning Structured Weight Uncertainty in Bayesian Neural Networks”. In: *AISTATS*. 2017, pp. 1283–1292.
- 46 [Sch00] A. Schrijver. “A combinatorial algorithm minimizing submodular functions in strongly polynomial time”. In: *Journal of Combinatorial Theory, Series B* 80.2 (2000), pp. 346–355.
- 47 [Sch02] N. N. Schraudolph. “Fast Curvature Matrix-Vector Products for Second-Order Gradient Descent”. In: *Neural Computation* 14 (2002).
- 48 [Sch04] A. Schrijver. *Combinatorial Optimization*. Springer, 2004.
- 49 [Sch+12a] B. Schoelkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. Mooij. “On causal and anticausal learning”. In: *ICML*. 2012.
- 50 [Sch+12b] B. Schölkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. Mooij. “On causal and anticausal learning”. In: *Proceedings of the 29th International Conference on International Conference on Machine Learning*. 2012, pp. 459–466.
- 51 [Sch+15a] J. Schulman, N. Heess, T. Weber, and P. Abbeel. “Gradient Estimation Using Stochastic Computation Graphs”. In: *NIPS*. 2015.
- 52 [Sch+15b] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. “Trust Region Policy Optimization”. In: *ICML*. 2015.
- 53 [Sch+16a] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. “Prioritized Experience Replay”. In: *ICLR*. 2016.
- 54 [Sch+16b] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. “High-Dimensional Continuous Control Using Generalized Advantage Estimation”. In: *ICLR*. 2016.

- 1
- 2 [Sch+17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. “Proximal Policy Optimization Algorithms”. In: (July 2017). arXiv: 1707.06347 [cs.LG].
- 3 [Sch+18] J. Schwarz, J. Luketina, W. M. Czarnecki, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell. “Progress & Compress: A scalable framework for continual learning”. In: ICML. May 2018.
- 4 [Sch19] B. Schölkopf. “Causality for Machine Learning”. In: (Nov. 2019). arXiv: 1911.10500 [cs.LG].
- 5 [Sch+21a] D. O. Scharfstein, R. Nabi, E. H. Kennedy, M.-Y. Huang, M. Bonvini, and M. Smid. *Semiparametric Sensitivity Analysis: Unmeasured Confounding In Observational Studies*. 2021. arXiv: 2104.08300 [stat.ME].
- 6 [Sch+21b] B. Schölkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio. “Toward Causal Representation Learning”. In: Proc. IEEE 109.5 (May 2021), pp. 612–634.
- 7 [Sch78] G. Schwarz. “Estimating the dimension of a model”. In: Annals of Statistics 6.2 (1978), pp. 461–464.
- 8 [Sco02] S Scott. “Bayesian methods for hidden Markov models: Recursive computing in the 21st century.” In: JASA (2002).
- 9 [Sco09] S. Scott. “Data augmentation, frequentist estimation, and the Bayesian analysis of multinomial logit models”. In: Statistical Papers (2009).
- 10 [Sco10] S. Scott. “A modern Bayesian look at the multi-armed bandit”. In: Applied Stochastic Models in Business and Industry 26 (2010), pp. 639–658.
- 11 [SCS19] A. Subbaswamy, B. Chen, and S. Sarria. *A Universal Hierarchy of Shift-Stable Distributions and the Tradeoff Between Stability and Performance*. 2019. arXiv: 1905.11374 [stat.ML].
- 12 [SD12] J. Sohl-Dickstein. “The Natural Gradient by Analogy to Signal Whitening, and Recipes and Tricks for its Use”. In: (May 2012). arXiv: 1205.1828 [cs.LG].
- 13 [SD+15a] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguly. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. In: ICML. 2015, pp. 2256–2265.
- 14 [SD+15b] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguly. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. In: ICML. Mar. 2015.
- 15 [SD17] H. Salimbeni and M. Deisenroth. “Doubly Stochastic Variational Inference for Deep Gaussian Processes”. In: NIPS. 2017.
- 16 [SDBD11] J. Sohl-Dickstein, P. Battaglino, and M. R. DeWeese. “Minimum probability flow learning”. In: Proceedings of the 28th International Conference on International Conference on Machine Learning. 2011, pp. 905–912.
- 17 [SE19] Y. Song and S. Ermon. “Generative Modeling by Estimating Gradients of the Data Distribution”. In: NIPS. 2019, pp. 11895–11907.
- 18 [SE20a] Y. Song and S. Ermon. “Multi-label Contrastive Predictive Coding”. In: NIPS. 2020.
- 19 [SE20b] Y. Song and S. Ermon. “Improved Techniques for Training Score-Based Generative Models”. In: NIPS. 2020.
- 20 [Sel+17] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. “Grad-cam: Visual explanations from deep networks via gradient-based localization”. In: Proceedings of the IEEE international conference on computer vision. 2017, pp. 618–626.
- 21 [Sel+19] A. D. Selbst, D. Boyd, S. A. Friedler, S. Venkatasubramanian, and T. Vertesi. “Fairness and Abstraction in Socio-technical Systems”. In: Proceedings of the Conference on Fairness, Accountability, and Transparency. FAT\*’19, Atlanta, GA, USA. Association for Computing Machinery, 2019, 59–68.
- 22 [Ser15] O. Serang. “Fast Computation on Semiring Isomorphic to  $(\times, \max)$  on  $\mathbb{R}_+$ ”. In: (Nov. 2015). arXiv: 1511.05690 [cs.DS].
- 23 [Ser+20] J. Serra, D. Álvarez, V. Gómez, O. Slizovskaia, J. F. Núñez, and J. Luque. “Input complexity and out-of-distribution detection with likelihood-based generative models”. In: ICLR. 2020.
- 24 [Set09] B. Settles. *Active learning literature survey*. Tech. rep. University of Wisconsin-Madison Department of Computer Sciences, 2009.
- 25 [Set94] J. Sethuraman. “A constructive definition of Dirichlet priors”. In: Statistica Sinica (1994), pp. 639–650.
- 26 [SF08] Z. Svitskina and L. Fleischer. “Submodular approximation: Sampling-based algorithms and lower bounds”. In: FOCS. 2008.
- 27 [SF20] K. Sokol and P. Flach. “Explainability fact sheets: a framework for systematic assessment of explainable approaches”. In: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency. 2020, pp. 56–67.
- 28 [SFB18] S. A. Sisson, Y. Fan, and M. A. Beaumont. *Overview of ABC*. In: Handbook of approximate Bayesian computation. Chapman and Hall/CRC, 2018, pp. 3–54.
- 29 [SG05] E. Snelson and Z. Ghahramani. “Compact Approximations to Bayesian Predictive Distributions”. In: ICML. 2005.
- 30 [SG06a] E. Snelson and Z. Ghahramani. “Sparse Gaussian processes using pseudo-inputs”. In: NIPS. 2006.
- 31 [SG06b] E. Snelson and Z. Ghahramani. “Sparse Gaussian Processes using Pseudo-inputs”. In: Advances in Neural Information Processing Systems. Ed. by Y. Weiss, B. Schölkopf, and J. Platt. Vol. 18. MIT Press, 2006.
- 32 [SG09] R. Silva and Z. Ghahramani. “The Hidden Life of Latent Variables: Bayesian Learning with Mixed Graph Models”. In: JMLR 10 (2009), pp. 1187–1238.
- 33 [SGF21] S. Särkkä and A. F. García-Fernández. “Temporal Parallelization of Bayesian Filters and Smoothers”. In: IEEE Trans. Automat. Contr. 66.1 (2021).
- 34 [SGS16] A. Sharghi, B. Gong, and M. Shah. “Query-focused extractive video summarization”. In: European Conference on Computer Vision. Springer, 2016, pp. 3–19.
- 35 [SH07] R. Salakhutdinov and G. Hinton. “Using Deep Belief Nets to Learn Covariance Kernels for Gaussian Processes”. In: NIPS. 2007.
- 36 [SH09] R. Salakhutdinov and G. Hinton. “Deep Boltzmann Machines”. In: AISTATS. Vol. 5. 2009, pp. 448–455.
- 37 [SH10] R. Salakhutdinov and G. Hinton. “Replicated Softmax: An Undirected Topic Model”. In: NIPS. 2010.
- 38 [SH22] T. Salimans and J. Ho. “Progressive Distillation for Fast Sampling of Diffusion Models”. In: ICLR. Feb. 2022.
- 39 [Sha+16] B. Shahriri, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. “Taking the Human Out of the Loop: A Review of Bayesian Optimization”. In: Proc. IEEE 104.1 (Jan. 2016), pp. 148–175.
- 40 [Sha16] L. S. Shapley. *17. A value for n-person games*. Princeton University Press, 2016.
- 41 [Sha+16] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. “Accessories to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition”. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016, pp. 1528–1540.
- 42 [Sha+19] A. Shaikhha, A. Fitzgibbon, D. Vytiniotis, and S. Peyton Jones. “Efficient differentiable programming in a functional array-processing language”. In: Proceedings of the ACM on Programming Languages 3.ICFP (2019), pp. 1–30.
- 43 [Sha+20] H. Shah, K. Tamuly, A. Raghunathan, P. Jain, and P. Netrapalli. “The Pitfalls of Simplicity Bias in Neural Networks”. In: NIPS. June 2020.
- 44 [Sha22] C. Shalizi. *Advanced Data Analysis from an Elementary Point of View*. Cambridge University Press, 2022.
- 45 [Sha48] C. Shannon. “A mathematical theory of communication”. In: Bell Systems Tech. Journal 27 (1948), pp. 379–423.
- 46 [Sha98] R. Shachter. “Bayes-Ball: The Rational Pastime (for determining Irrelevance and Requisite Information in Belief Networks and Influence Diagrams)”. In: UAI. 1998.
- 47 [She+11] C. Shen, X. Li, L. Li, and M. C. Were. “Sensitivity analysis for causal inference using inverse probability weighting”. In: Biometrical Journal 53.5 (2011), pp. 822–837. eprint: <https://onlinelibrary.wiley.com/doi/10.1002/bimj.201100042>.
- 48 [She+17] T. Shen, T. Lei, R. Barzilay, and T. Jaakkola. “Style transfer from non-parallel text by cross-alignment”. In: Advances in neural information processing systems 30 (2017), pp. 6830–6841.
- 49 [She+20] T. Shen, J. Mueller, R. Barzilay, and T. Jaakkola. “Educating Text Autoencoders: Latent Representation Guidance via Denoising”. In: ICML. 2020.
- 50 [She+21] Z. Shen, J. Liu, Y. He, X. Zhang, R. Xu, H. Yu, and P. Cui. “Towards Out-Of-Distribution Generalization: A Survey”. In: (Aug. 2021). arXiv: 2108.13624 [cs.LG].
- 51 [SHF15] R. Steorts, R. Hall, and S. Fienberg. “A Bayesian Approach to Graphical Record Linkage and De-duplication”. In: JASA (2015).
- 52 [Shi00a] H. Shimodaira. “Improving predictive inference under covariate shift by weighting the log-likelihood function”. In: J. Stat. Plan. Inference 90.2 (Oct. 2000), pp. 227–244.
- 53 [Shi00b], B. Shipley. *Cause and Correlation in Biology: A User’s Guide to Path Analysis, Structural Equations and Causal Inference*. Cambridge, 2000.
- 54 [Shi+21] C. Shi, D. Sridhar, V. Misra, and D. M. Blei. “On the Assumptions of Synthetic Control Methods”. In: (Dec. 2021). arXiv: 2112.05671 [stat.ME].
- 55 [SHJ97] P. Smyth, D. Heckerman, and M. I. Jordan. “Probabilistic Independence Networks for Hidden Markov Probability Models”. In: Neural Computation 9.2 (1997), pp. 227–269.
- 56 [SHM14] D. Soudry, I. Hubara, and R. Meir. “Expectation back-propagation: Parameter-free training of multilayer neural networks with continuous or discrete weights”. In: NIPS. 2014.

- 1
- [Shr+16] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje. “Not just a black box: Learning important features through propagating activation differences”. In: *arXiv preprint arXiv:1605.01713* (2016).
- 2
- [SHS1] D. Stutz, M. Hein, and B. Schiele. “Confidence-calibrated adversarial training: Generalizing to unseen attacks”. In: ( ).
- 3
- [SHS01] B. Schölkopf, R. Herbrich, and A. J. Smola. “A Generalized Representer Theorem”. In: *COLT, COLT '01/EuroCOLT '01*. London, UK, UK: Springer-Verlag, 2001, pp. 416–426.
- 4
- [Shu+19] K. Shu, L. Cui, S. Wang, D. Lee, and H. Liu. “defend: Explainable fake news detection”. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 395–405.
- 5
- [SII00] M. Sato and S. Ishii. “On-line EM algorithm for the normalized Gaussian network”. In: *Neural Computation* 12 (2000), pp. 407–432.
- 6
- [Sil+14] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. “Deterministic Policy Gradient Algorithms”. In: *ICML, ICML'14*. Beijing, China: JMLR.org, 2014, pp. I-387–I-395.
- 7
- [Sil+16] D. Silver et al. “Mastering the game of Go with deep neural networks and tree search”. en. In: *Nature* 529.7587 (2016), pp. 484–489.
- 8
- [Sil18] D. Silver. *Lecture 9L Exploration and Exploitation*. 2018.
- 9
- [Sil+18] D. Silver et al. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”. en. In: *Science* 362.6419 (Dec. 2018), pp. 1140–1144.
- 10
- [Sil85] B. W. Silverman. “Some Aspects of the Spline Smoothing Approach to Non-Parametric Regression Curve Fitting”. In: *J. R. Stat. Soc. Series B Stat. Methodol.* 47.1 (1985), pp. 1–52.
- 11
- [Sim06] D. Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley, 2006.
- 12
- [Sin+00] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári. “Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms”. In: *MLJ* 38.3 (Mar. 2000), pp. 287–308.
- 13
- [Sin67] R. Sinkhorn. “Diagonal Equivalence to Matrices with Prescribed Row and Column Sums”. In: *The American Mathematical Monthly* 74.4 (1967), pp. 402–405.
- 14
- [SJ08] S. Shridharan and D. W. Jacobs. “Approximate earth mover’s distance in linear time”. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.
- 15
- [SJ15] A. Swaminathan and T. Joachims. “Batch Learning from Logged Bandit Feedback through Counterfactual Risk Minimization”. In: *JMLR* 16.1 (2015), pp. 1731–1755.
- 16
- [SJ95] L. Saul and M. Jordan. “Exploiting tractable substructures in intractable networks”. In: *NIPS*. Vol. 8. 1995.
- 17
- [SJ96] L. Saul, T. Jaakkola, and M. Jordan. “Mean Field Theory for Sigmoid Belief Networks”. In: *JAIR* 4 (1996), pp. 61–76.
- 18
- [SJR04] S. Singh, M. James, and M. Rudary. “Predictive state representations: A new theory for modeling dynamical systems”. In: *UAI*. 2004.
- 19
- [SK19] C. Shorthen and T. M. Khoshgoftaar. “A survey on Image Data Augmentation for Deep Learning”. en. In: *Journal of Big Data* 6.1 (July 2019), pp. 1–48.
- 20
- [SK20] S. Singh and S. Krishnan. “Filter Response Normalization Layer: Eliminating Batch Dependence in the Training of Deep Neural Networks”. In: *CVPR*. 2020.
- 21
- [SK21] Y. Song and D. P. Kingma. “How to Train Your Energy-Based Models”. In: (Jan. 2021). arXiv: 2101.03288 [cs, LG].
- 22
- [SK89] R. Shachter and C. R. Kenley. “Gaussian Influence Diagrams”. In: *Management Science* 35.5 (1989), pp. 527–550.
- 23
- [Ski89] J. Skilling. “The eigenvalues of mega-dimensional matrices”. In: *Maximum Entropy and Bayesian Methods*. Springer, 1989, pp. 455–466.
- 24
- [SKM07] M. Sugiyama, M. Krauledat, and K.-R. Müller. “Covariate Shift Adaptation by Importance Weighted Cross Validation”. In: *J. Mach. Learn. Res.* 8.35 (2007), pp. 985–1005.
- 25
- [SKM21] M. Shanahan, C. Kaplanis, and J. Mitrović. “Encoders and Ensembles for Task-Free Continual Learning”. In: (May 2021). arXiv: 2105.13272 [cs, LG].
- 26
- [SKTF18] H. Shao, A. Kumar, and P. Thomas Fletcher. “The Riemannian Geometry of Deep Generative Models”. In: *CVPR*. 2018, pp. 315–323.
- 27
- [SKW15] T. Salimans, D. Kingma, and M. Welling. “Markov Chain Monte Carlo and Variational Inference: Bridging the Gap”. In: *ICML*, 2015, pp. 1218–1226.
- 28
- [SKZ19] J. Shi, M. E. Khan, and J. Zhu. “Scalable Training of Inference Networks for Gaussian-Process Models”. In: *ICML*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceed-
- 29
- ings of Machine Learning Research. PMLR, 2019, pp. 5758–5768.
- 30
- [SL08] A. L. Strehl and M. L. Littman. “An Analysis of Model-based Interval Estimation for Markov Decision Processes”. In: *J. of Comp. and Sys. Sci.* 74.8 (2008), pp. 1309–1331.
- 31
- [SL18] S. L. Smith and Q. V. Le. “A Bayesian Perspective on Generalization and Stochastic Gradient Descent”. In: *ICLR*. 2018.
- 32
- [SL+21] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. B. Wiltschko. “A Gentle Introduction to Graph Neural Networks”. In: *Distill* (2021). <https://distill.pub/2021/gnn-intro>.
- 33
- [SL90] D. J. Spiegelhalter and S. L. Lauritzen. “Sequential updating of conditional probabilities on directed graphical structures”. In: *Networks* 20 (1990).
- 34
- [Sla+20] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju. “Fooling lime and shap: Adversarial attacks on post hoc explanation methods”. In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 2020, pp. 180–186.
- 35
- [SLG17] A. Sharghi, J. S. Laurel, and B. Gong. “Query-focused video summarization: Dataset, evaluation, and a memory network based approach”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4788–4797.
- 36
- [Sli19] A. Slivkins. “Introduction to Multi-Armed Bandits”. In: *Foundations and Trends in Machine Learning* (2019).
- 37
- [SLL09] A. L. Strehl, L. Li, and M. L. Littman. “Reinforcement Learning in Finite MDPs: PAC Analysis”. In: *JMLR* 10 (2009), pp. 2413–2444.
- 38
- [SLW19] M. Sadinle, J. Lei, and L. Wasserman. “Least Ambiguous Set-Valued Classifiers With Bounded Error Levels”. In: *JASA* 114.525 (Jan. 2019), pp. 223–234.
- 39
- [SM07] C. Sutton and A. McCallum. “Improved Dynamic Schedules for Belief Propagation”. In: *UAI*. 2007.
- 40
- [SM12] Y. Saika and K. Morimoto. “Generalized MAP estimation via parameter scheduling and maximizer of the posterior marginal estimate for image reconstruction using multiple half-toned images”. In: *12th International Conference on Control, Automation and Systems*. 2012, pp. 1285–1289.
- 41
- [SMB10] H. Schulz, A. Müller, and S. Behnke. “Investigating convergence of restricted boltzmann machine learning”. In: *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*. Vol. 1. 2. 2010, pp. 6–1.
- 42
- [SMH07] R. R. Salakutdinov, A. Mnih, and G. E. Hinton. “Restricted Boltzmann machines for collaborative filtering”. In: *ICML*. Vol. 24. 2007, pp. 791–798.
- 43
- [Smi+00] G. Smith, J. F. G. de Freitas, T. Robinson, and M. Niranjani. “Speech Modelling Using Subspace and EM Techniques”. In: *NIPS*. MIT Press, 2000, pp. 796–802.
- 44
- [Smi+06] V. Smith, J. Yu, T. Smulders, A. Hartemink, and E. Jarvis. “Computational Inference of Neural Information Flow Networks”. In: *PLOS Computational Biology* 2 (2006), pp. 1436–1439.
- 45
- [Smi+17] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. “SmoothGrad: removing noise by adding noise”. 2017. arXiv: 1706.03825 [cs, LG].
- 46
- [Smo86] P. Smolensky. “Information processing in dynamical systems: foundations of harmony theory”. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1*. Ed. by D. Rumelhart and J. McClelland. McGraw-Hill, 1986.
- 47
- [SMS17] M. Saito, E. Matsumoto, and S. Saito. “Temporal Generative Adversarial Nets with Singular Value Clipping”. In: *ICCV*. 2017.
- 48
- [SMT18] M. R. U. Saputra, A. Markham, and N. Trigoni. “Visual SLAM and Structure from Motion in Dynamic Environments: A Survey”. In: *ACM Computing Surveys* 51.2 (Feb. 2018), pp. 1–36.
- 49
- [Smy20] S. Smyl. “A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting”. In: *Int. J. Forecast.* 36.1 (Jan. 2020), pp. 75–85.
- 50
- [Son+16] C. K. Sønderby, T. Raiko, L. Maaløe, S. R. K. Sønderby, and O. Winther. “Ladder Variational Autoencoders”. In: *NIPS*. Curran Associates, Inc., 2016, pp. 3738–3746.
- 51
- [SNM16] M. Suzuki, K. Nakayama, and Y. Matsuo. “Joint Multimodal Learning with Deep Generative Models”. In: (2016). arXiv: 1611.01891 [stat, ML].
- 52
- [Son+16] C. Sønderby, T. Raiko, L. Maaløe, S. Sønderby, and O. Winther. “How to Train Deep Variational Autoencoders and Probabilistic Ladder Networks”. In: *ICML*. 2016.
- 53
- [Son+19] Y. Song, S. Garg, J. Shi, and S. Ermon. “Sliced Score Matching: A Scalable Approach to Density and Score Estimation”. In: *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22–25, 2019*. 2019, p. 204.
- 54
- [Son+21] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. “Score-Based Generative Modeling through Stochastic Differential Equations”. In: *ICLR*. 2021.

- 1
- 2 [Son98] E. D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. 2nd. Vol. 6. Texts in Applied Mathematics. Springer, 1998.
- 3 [SPD92] S. Shah, F. Palmieri, and M. Datum. "Optimal filtering algorithms for fast learning in feedforward neural networks". In: *Neural Netw.* 5.5 (Sept. 1992), pp. 779–787.
- 4 [Spi71] M. Spivak. *Calculus On Manifolds: A Modern Approach To Classical Theorems Of Advanced Calculus*. Westview Press; 5th edition, 1971.
- 5 [Spr+14] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. "Striving for simplicity: The all convolutional net". In: *arXiv preprint arXiv:1412.6806* (2014).
- 6 [Spr+16] J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter. "Bayesian Optimization with Robust Bayesian Neural Networks". In: *NIPS*. 2016, pp. 4141–4149.
- 7 [Spr17] M. W. Spratling. "A review of predictive coding algorithms". en. In: *Brain Cogn.* 112 (Mar. 2017), pp. 92–97.
- 8 [SPW18] M. H. S. Segler, M. Preuss, and M. P. Waller. "Planning chemical syntheses with deep neural networks and symbolic AI". en. In: *Nature* 555.7698 (Mar. 2018), pp. 604–610.
- 9 [SPZ09] P. Schniter, L. C. Potter, and J. Ziniel. "Fast Bayesian Matching Pursuit: Model Uncertainty and Parameter Estimation for Sparse Linear Models". In: *IEEE Trans. on Signal Processing* (2009).
- 10 [SRG03] R. Salakhutdinov, S. T. Roweis, and Z. Ghahramani. "Optimization with EM and Expectation-Conjugate-Gradient". In: *ICML*. 2003.
- 11 [Sri+09] B. K. Sriperumbudur, K. Fukumizu, A. Gretton, B. Schölkopf, and G. R. G. Lanckriet. "On integral probability metrics,  $\phi$ -divergences and binary classification". In: (Jan. 2009). arXiv: 0901.2698 [cs.IT].
- 12 [Sri+10] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design". In: *ICML*. 2010, pp. 1015–1022.
- 13 [Sri+14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *JMLR* (2014).
- 14 [Sri+17] A. Srivastava, L. Volkov, C. Russell, M. U. Gutmann, and C. Sutton. "Veegan: Reducing mode collapse in gans using implicit variational learning". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 3310–3320.
- 15 [SRS10] P. Schnitzspan, S. Roth, and B. Schiele. "Automatic discovery of meaningful object parts with latent CRFs". In: *CVPR*. 2010.
- 16 [SS17a] A. Srivastava and C. Sutton. "Autoencoding Variational Inference For Topic Models". In: *ICLR*. 2017.
- 17 [SS17b] A. Svensson and T. B. Schön. "A flexible state-space model for learning nonlinear dynamical systems". In: *Automatica* 80 (June 2017), pp. 189–199.
- 18 [SS18a] O. Sener and S. Savarese. "Active Learning for Convolutional Neural Networks: A Core-Set Approach". In: *International Conference on Learning Representations*. 2018.
- 19 [SS18b] A. Subbaswamy and S. Saria. "Counterfactual Normalization: Proactively Addressing Dataset Shift and Improving Reliability Using Causal Mechanisms". In: *UAI*. Aug. 2018.
- 20 [SS18c] A. Subbaswamy and S. Saria. "Counterfactual Normalization: Proactively Addressing Dataset Shift and Improving Reliability Using Causal Mechanisms". In: *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018. 2018.
- 21 [SS19] S. Sarkka and A. Solin. *Applied stochastic differential equations*. en. Cambridge University Press, 2019.
- 22 [SS20] K. E. Smith and A. O. Smith. "Conditional GAN for time-series generation". In: *arXiv preprint arXiv:2006.16477* (2020).
- 23 [SS21] I. Sucholutsky and M. Schonlau. "Soft-Label Dataset Distillation and Text Dataset Distillation". In: *2021 International Joint Conference on Neural Networks (IJCNN)*. 2021, pp. 1–8.
- 24 [SS90] G. R. Shafer and P. P. Shenoy. "Probability propagation". In: *Annals of Mathematics and AI* 2 (1990), pp. 327–352.
- 25 [SSA14] K. Swersky, J. Snoek, and R. P. Adams. "Freeze-Thaw Bayesian Optimization". In: (June 2014). arXiv: 1406 . 3896 [stat.ML].
- 26 [SSA18] K. Shmelkov, C. Schmid, and K. Alahari. "How good is my GAN?". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 213–229.
- 27 [SSB17] S. Semeniuta, A. Severyn, and E. Barth. "A Hybrid Convolutional Variational Autoencoder for Text Generation". In: (2017). arXiv: 1702.02390 [cs.CL].
- 28 [SSE18] Y. Song, J. Song, and S. Ermon. "Accelerating Natural Gradients with Higher-Order Invariance". In: *ICML*. 2018.
- 29 [SSF16] M. W. Seeger, D. Salinas, and V. Flunkert. "Bayesian Intermittent Demand Forecasting for Large Inventories". In: *NIPS*. 2016, pp. 4646–4654.
- 30 [SSG18a] S. Semeniuta, A. Severyn, and S. Gelly. "On Accurate Evaluation of GANs for Language Generation". In: *arXiv preprint arXiv:1806.04936* (2018).
- 31 [SSG18b] S. Semeniuta, A. Severyn, and S. Gelly. "On Accurate Evaluation of GANs for Language Generation". In: (June 2018). arXiv: 1806.04936 [cs.CL].
- 32 [SSG19] R. Singh, M. Sahani, and A. Gretton. "Kernel Instrumental Variable Regression". In: *Advances in Neural Information Processing Systems*. 2019, pp. 4593–4605.
- 33 [SSH13] S. Sarkka, A. Solin, and J. Hartikainen. "Spatio-Temporal Learning via Infinite-Dimensional Bayesian Filtering and Smoothing: A look at Gaussian process regression through Kalman filtering". In: *IEEE Signal Processing Magazine* (2013).
- 34 [SSJ12] R. Sipos, P. Shivaswamy, and T. Joachims. "Large-margin learning of submodular summarization models". In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. 2012, pp. 224–233.
- 35 [SSK12] M. Sugiyama, T. Suzuki, and T. Kanamori. *Density Ratio Estimation in Machine Learning*. en. Cambridge University Press, Feb. 2012.
- 36 [SSM18] S. Santurkar, L. Schmidt, and A. Madry. "A classification-based study of covariate shift in gan distributions". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4480–4489.
- 37 [SSZ17] J. Snell, K. Swersky, and R. Zemel. "Prototypical networks for few-shot learning". In: *NIPS*. 2017, pp. 4077–4087.
- 38 [Sta] *Scientific Explanation*. <https://plato.stanford.edu/entries/scientific-explanation/#ConcOpenIssueFutureDir>. Accessed: 2021-11-23.
- 39 [Sta+07] K. O. Stanley. "Compositional pattern producing networks: A novel abstraction of development". In: *Genet. Program. Evolvable Mach.* 8.2 (Oct. 2007), pp. 131–162.
- 40 [Sta+17] N. Stallard, F. Miller, S. Day, S. W. Hee, J. Madan, S. Zahor, and M. Posch. "Determination of the optimal sample size for a clinical trial accounting for the population size". en. In: *Biom. J.* 59.4 (July 2017), pp. 609–625.
- 41 [Ste81] C. M. Stein. "Estimation of the mean of a multivariate normal distribution". In: *The annals of Statistics* (1981), pp. 1135–1151.
- 42 [Sto09] A. J. Storkey. "When Training and Test Sets are Different: Characterising Learning Transfer". In: *Dataset Shift in Machine Learning*. 2009.
- 43 [Sto17] J. Stoehr. "A review on statistical inference methods for discrete Markov random fields". In: (Apr. 2017). arXiv: 1704 . 0331 [stat.ME].
- 44 [STR10] Y. Saatchi, R. Turner, and C. E. Rasmussen. "Gaussian Process Change Point Models". In: *ICML*. unknown, Aug. 2010, pp. 927–934.
- 45 [Str15] S. H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering, Second Edition*. CRC Press, 2015.
- 46 [Str+17] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. "Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks". In: *IEEE transactions on visualization and computer graphics* 24.1 (2017), pp. 667–676.
- 47 [Str19] M. Streeter. "Bayes Optimal Early Stopping Policies for Black-Box Optimization". In: (Feb. 2019). arXiv: 1902 . 08285 [cs.LG].
- 48 [Stu+22] D. Stutz, Krishnamurthy, Dvijotham, A. T. Cemgil, and A. Doucet. "Learning Optimal Conformal Classifiers". In: *ICLR*. 2022.
- 49 [STY17] M. Sundararajan, A. Taly, and Q. Yan. *Axiomatic Attribution for Deep Networks*. 2017. arXiv: 1703 . 01365 [cs.LG].
- 50 [Suc+20] F. P. Such, A. Rawal, J. Lehman, K. Stanley, and J. Clune. "Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9206–9216.
- 51 [Sud+03] E. Sudderth, A. Ihler, W. Freeman, and A. Wilsky. "Nonparametric Belief Propagation". In: *CVPR*. 2003.
- 52 [Sud06] E. Sudderth. "Graphical Models for Visual Object Recognition and Tracking". PhD thesis. MIT, 2006.
- 53 [Sud+10] E. Sudderth, A. Ihler, M. Isard, W. Freeman, and A. Wilsky. "Nonparametric Belief Propagation". In: *Comm. of the ACM* 53.10 (2010).
- 54 [Sug+13] M. Sugiyama, T. Kanamori, T. Suzuki, M. C. du Plessis, S. Liu, and I. Takeuchi. "Density-difference estimation". en. In: *Neural Comput.* 25.10 (Oct. 2013), pp. 2734–2775.
- 55 [Sun+09] L. Sun, S. Ji, S. Yu, and J. Ye. "On the Equivalence Between Canonical Correlation Analysis and Orthonormalized Partial Least Squares". In: *IJCAC*. 2009.
- 56 [Sun+18] S. Sun, G. Zhang, C. Wang, W. Zeng, J. Li, and R. Gross. "Differentiable Compositional Kernel Learning for Gaussian Processes". In: *ICML*. 2018.

- 1
- [Sun+19a] S. Sun, G. Zhang, J. Shi, and R. Grosse. "Functional variational bayesian neural networks". In: *arXiv preprint arXiv:1903.05779* (2019).
- 2
- [Sun+19b] S. Sun, Z. Cao, H. Zhu, and J. Zhao. "A Survey of Optimization Methods from a Machine Learning Perspective". In: (June 2019). arXiv: [1906.06821 \[cs.LG\]](#).
- 3
- [Sun+19c] M. Sundararajan, J. Xu, A. Taly, R. Sayres, and A. Najmi. "Exploring Principled Visualizations for Deep Network Attributions". In: *IUT Workshops*. Vol. 4. 2019.
- 4
- [Sun+20] Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt. "Test-Time Training with Self-Supervision for Generalization under Distribution Shifts". In: *ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 9229–9248.
- 5
- [Sut+17] D. J. Sutherland, H.-Y. Tung, H. Strathmann, S. De, A. Ramdas, A. Smola, and A. Gretton. "Generative Models and Model Criticism via Optimized Maximum Mean Discrepancy". In: *ICLR*. 2017.
- 6
- [Sut88] R. Sutton. "Learning to predict by the methods of temporal differences". In: *Machine Learning* 3.1 (1988), pp. 9–44.
- 7
- [Sut90] R. S. Sutton. "Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming". In: *ICML*. Ed. by B. Porter and R. Mooney. San Francisco (CA): Morgan Kaufmann, 1990, pp. 216–224.
- 8
- [Sut96] R. S. Sutton. "Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding". In: *NIPS*. Ed. by D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo. MIT Press, 1996, pp. 1038–1044.
- 9
- [Sut+99] R. Sutton, D. McAllester, S. Singh, and Y. Mansour. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *NIPS*. 1999.
- 10
- [SV08] G. Shafer and V. Vovk. "A Tutorial on Conformal Prediction". In: *JMLR* 9.Mar (2008), pp. 371–421.
- 11
- [SV98] M. Studeny and J. Vejnarova. "The multi-information function as a tool for measuring stochastic dependence". In: *Learning in graphical models*. Ed. by M. Jordan. MIT Press, 1998, pp. 261–297.
- 12
- [SVE04] A. Smola, S. V. N. Vishwanathan, and E. Eskin. "Laplace Propagation". In: *NIPS*. MIT Press, 2004, pp. 441–448.
- 13
- [Svi04] M. Sviridenko. "A note on maximizing a submodular set function subject to a knapsack constraint". In: *Operations Research Letters* 32.1 (2004), pp. 41–43.
- 14
- [SVK19] J. Su, D. V. Vargas, and S. Kouichi. "One pixel attack for fooling deep neural networks". In: *IEEE Trans. Evol. Comput.* 23.5 (2019).
- 15
- [SVZ13] K. Simonyan, A. Vedaldi, and A. Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps". In: *arXiv preprint arXiv:1312.6034* (2013).
- 16
- [SW06] J. E. Smith and R. L. Winkler. "The Optimizer's Curse: Skepticism and Postdecision Surprise in Decision Analysis". In: *Manage. Sci.* 52.3 (Mar. 2006), pp. 311–322.
- 17
- [SW13] G. J. Sussman and J. Wisdom. *Functional Differential Geometry*. Functional Differential Geometry. MIT Press, 2013.
- 18
- [SW20] V. G. Satorras and M. Welling. "Neural Enhanced Belief Propagation on Factor Graphs". In: (Mar. 2020). arXiv: [2003.01998 \[cs.LG\]](#).
- 19
- [SW87] R. Swendsen and J.-S. Wang. "Nonuniversal critical dynamics in Monte Carlo simulations". In: *Physical Review Letters* 58 (1987), pp. 86–88.
- 20
- [SW89] S. Singhal and L. Wu. "Training Multilayer Perceptrons with the Extended Kalman Algorithm". In: *NIPS*. Vol. 1. 1989.
- 21
- [Swe+10] K. Swersky, B. Chen, B. Marlin, and N. de Freitas. "A Tutorial on Stochastic Approximation Algorithms for Training Restricted Boltzmann Machines and Deep Belief Nets". In: *Information Theory and Applications (ITA) Workshop*. 2010.
- 22
- [Swe+13] K. Swersky, D. Duvenaud, J. Snoek, F. Hutter, and M. A. Osborne. "Raiders of the Lost Architecture: Kernels for Bayesian Optimization in Conditional Parameter Spaces". In: *NIPS BayesOpt workshop*. 2013.
- 23
- [SWL03] M. Seeger, C. K. I. Williams, and N. D. Lawrence. "Fast Forward Selection to Speed Up Sparse Gaussian Process Regression". In: *AISTATS*. 2003.
- 24
- [SWW08] E. Sudderth, M. Wainwright, and A. Willsky. "Loop series and Bethe variational bounds in attractive graphical models". In: *NIPS* (2008).
- 25
- [SYD19] R. Sen, H.-F. Yu, and I. Dhillon. "Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting". In: *NIPS*. 2019.
- 26
- [SZZ22] R. Shwartz-Ziv. "Information Flow in Deep Neural Networks". PhD thesis. Feb. 2022.
- 27
- [Sze10] C. Szepesvari. *Algorithms for Reinforcement Learning*. Morgan Claypool, 2010.
- 28
- [Sze+14] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. "Intriguing properties of neural networks". In: *ICLR*. 2014.
- 29
- [Sze+15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going Deeper with Convolutions". In: *CVPR*. 2015.
- 30
- [TAH20] D. Teney, E. Abbasnejad, and A. van den Hengel. "Learning What Makes a Difference from Counterfactual Examples and Gradient Supervision". In: *CoRR* abs/2004.09034 (2020). arXiv: [2004.09034](#).
- 31
- [Tan+16] X. Tan, S. A. Naqvi, K. A. Heller, and V. A. Rao. "Content-based Modeling of Reciprocal Relationships using Hawkes and Gaussian Processes". In: *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*. 2016.
- 32
- [TB16] P. S. Thomas and E. Brunskill. "Data-Efficient Off-Policy Policy Evaluation for Reinforcement Learning". In: *ICML*. 2016, pp. 2139–2148.
- 33
- [TB99] M. Tipping and C. Bishop. "Probabilistic principal component analysis". In: *J. of Royal Stat. Soc. Series B* 21.3 (1999), pp. 611–622.
- 34
- [TBA19] N. Tremblay, S. Barthélémy, and P.-O. Amblard. "Determinantal Point Processes for Coresets". In: *J. Mach. Learn. Res.* 20 (2019), pp. 168–1.
- 35
- [TBB19] J. Townsend, T. Bird, and D. Barber. "Practical Lossless Compression with Latent Variables using Bits Back Coding". In: *ICLR*. 2019.
- 36
- [TBF06] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2006.
- 37
- [TBS13] R. Turner, S. Bottone, and C. Stanek. "Online variational approximations to non-exponential family change point models: With application to radar tracking". In: *NIPS*. 2013.
- 38
- [TDR10] R. Turner, M. Deisenroth, and C. Rasmussen. "State-Space Inference and Learning with Gaussian Processes". In: *AISTATS*. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 2010, pp. 868–875.
- 39
- [Teh06] Y. W. Teh. "A hierarchical Bayesian language model based on Pitman-Yor processes". In: *Proc. of the Assoc. for Computational Linguistics*. 2006, 985–992.
- 40
- [Teh+06] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. "Hierarchical Dirichlet Processes". In: *JASA* 101.476 (2006), pp. 1566–1581.
- 41
- [Teh+20] N. Tehrani, N. S. Arora, Y. L. Li, K. D. Shah, D. Nouris, M. Tingley, N. Torabi, S. Masouleh, E. Lippert, and E. Meijer. "Beam Machine: A Declarative Probabilistic Programming Language For Efficient Programmable Inference". In: *Proceedings of the 10th International Conference on Probabilistic Graphical Models*. Ed. by M. Jaeger and T. D. Nielsen. Vol. 138. Proceedings of Machine Learning Research. PMLR, 2020, pp. 485–496.
- 42
- [Ten+20] I. Tenney, J. Wexler, J. Bastings, T. Bolukbasi, A. Cohen, S. Germann, E. Jiang, M. Pushkarna, C. Radobaugh, E. Reif, et al. "The language interpretability tool: Extensible, interactive visualizations and analysis for NLP models". In: *arXiv preprint arXiv:2008.05122* (2020).
- 43
- [TF03] M. Tipping and A. Faul. "Fast marginal likelihood maximisation for sparse Bayesian models". In: *AI/Stats*. 2003.
- 44
- [TG09] Y. W. Teh and D. Gorur. "Indian buffet processes with power-law behavior". In: *NIPS*. 2009, pp. 1838–1846.
- 45
- [TG18] L. Tu and K. Gimpel. "Learning Approximate Inference Networks for Structured Prediction". In: *ICLR*. 2018.
- 46
- [Tho+19] V. Thomas, F. Pedregosa, B. van Merriënboer, P.-A. Mangazol, Y. Bengio, and N. Le Roux. "Information matrices and generalization". In: (June 2019). arXiv: [1906.07774 \[cs.LG\]](#).
- 47
- [Tho33] W. R. Thompson. "On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples". In: *Biometrika* 25.3/4 (1933), pp. 285–294.
- 48
- [Thr+04] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot. "FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association". In: *JMLR* 2004 (2004).
- 49
- [Thr98] S. Thrun. "Lifelong learning algorithms". In: *Learning to learn*. Ed. by S. Thrun and L. Pratt. Kluwer, 1998, pp. 181–209.
- 50
- [Thu+21] S. Thulasidasan, S. Thapa, S. Dhaubhadel, G. Chennupati, T. Bhattacharya, and J. Bilmes. "An Effective Baseline for Robustness to Distributional Shift". In: (May 2021). arXiv: [2105.07107 \[cs.LG\]](#).
- 51
- [Tib+19] R. J. Tibshirani, R. F. Barber, E. J. Candes, and A. Ramdas. "Conformal Prediction Under Covariate Shift". In: *NIPS*. Apr. 2019.
- 52
- [Tib96] R. Tibshirani. "Regression shrinkage and selection via the lasso". In: *J. Royal. Statist. Soc B* 58.1 (1996), pp. 267–288.
- 53
- [Tie08] T. Tielemans. "Training restricted Boltzmann machines using approximations to the likelihood gradient". In: *Proceed-*

- 1
- 2      *ings of the 25th international conference on Machine learning.* ACM New York, NY, USA. 2008, pp. 1064–1071.
- 3 [Tip01] M. Tipping. “Sparse Bayesian learning and the relevance vector machine”. In: *JMLR* 1 (2001), pp. 211–244.
- 4 [Tip98] M. Tipping. “Probabilistic visualization of high-dimensional binary data”. In: *NIPS*. 1998.
- 5 [Tit09] M. K. Titsias. “Variational Learning of Inducing Variables in Sparse Gaussian Processes”. In: *AISTATS*. 2009.
- 6 [TK86] L. Tierney and J. Kadane. “Accurate approximations for posterior moments and marginal densities”. In: *JASA* 81.393 (1986).
- 7 [TKW08] Y. W. Teh, K. Kurihara, and M. Welling. “Collapsed variational inference for HDP”. In: *Advances in neural information processing systems*. 2008, pp. 1481–1488.
- 8 [TL05] E. Todorov and W. Li. “A Generalized Iterative LQG Method for Locally-Optimal Feedback Control of Constrained Nonlinear Stochastic Systems”. In: *ACC*. 2005, pp. 300–306.
- 9 [TL18a] S. J. Taylor and B. Letham. “Forecasting at scale”. en. In: *The American Statistician* 72.1 (Jan. 2018), pp. 37–45.
- 10 [TL18b] G. Tucker and D. Lawson. “Doubly Reparameterized Gradient Estimators for Monte Carlo Objectives”. In: *1st Symposium on Advances in Approximate Bayesian Inference*. 2018.
- 11 [TLC14] M. Titsias and M. Lázaro-Gredilla. “Doubly Stochastic Variational Bayes for non-Conjugate Inference”. In: *ICML*. 2014, pp. 1971–1979.
- 12 [TMD12] A. Talhouk, K. Murphy, and A. Doucet. “Efficient Bayesian Inference for Multivariate Probit Models with Sparse Inverse Correlation Matrices”. In: *J. Comp. Graph. Statist.* 21.3 (2012), pp. 739–757.
- 13 [TMK04] G. Theodorou, K. Murphy, and L. Kaelbling. “Representing hierarchical POMDPs as DBNs for multi-scale robot localization”. In: *ICRA*. 2004.
- 14 [TN13] L. S. L. Tan and D. J. Nott. “Variational Inference for Generalized Linear Mixed Models Using Partially Noncentered Parametrizations”. In: *Stat. Sci.* (2013).
- 15 [TN18] L. S. L. Tan and D. J. Nott. “Gaussian variational approximation with sparse precision matrices”. In: *Stat. Comput.* 28.2 (Mar. 2018), pp. 259–275.
- 16 [TND21] M.-N. Tran, T.-N. Nguyen, and V.-H. Dao. “A practical tutorial on Variational Bayes”. In: (Mar. 2021). arXiv: 2103.01327 [stat.CO].
- 17 [TOB16] L. Theis, A. van den Oord, and M. Bethge. “A note on the evaluation of generative models”. In: *ICLR*. 2016.
- 18 [Tob+17] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. “Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World”. In: *IROS*. 2017.
- 19 [Tom+20] R. Tomsett, D. Harborne, S. Chakraborty, P. Gurrum, and A. Preece. “Sanity checks for saliency metrics”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 04. 2020, pp. 6021–6029.
- 20 [Tom22] J. M. Tomczak. *Deep Generative Modeling*. en. 1st ed. Springer, Feb. 2022.
- 21 [Tou09] M. Toussaint. “Robot Rbrajectory Optimization using Approximate Inference”. In: *ICML*. 2009, pp. 1049–1056.
- 22 [Tou+19] J. Toubeau, J. Bottieau, F. Vallée, and Z. De Grève. “Deep Learning-Based Multivariate Probabilistic Forecasting for Short-Term Scheduling in Power Markets”. In: *IEEE Trans. Power Syst.* 34.2 (Mar. 2019), pp. 1203–1215.
- 23 [TOV18] C. Truong, L. Oudre, and N. Vayatis. “Selective review of offline change point detection methods”. In: (Jan. 2018). arXiv: 1801.00718 [cs.CE].
- 24 [TP97] S. Thrun and L. Pratt, eds. *Learning to learn*. Kluwer, 1997.
- 25 [TPB99] N. Tishby, F. Pereira, and W. Biale. “The Information Bottleneck method”. In: *The 37th annual Allerton Conf. on Communication, Control, and Computing*. 1999, pp. 368–377.
- 26 [TR19] M. K. Titsias and F. Ruiz. “Unbiased Implicit Variational Inference”. In: *AISTATS*. Ed. by K. Chaudhuri and M. Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, 2019, pp. 167–176.
- 27 [TR97] J. Tsitsiklis and B. V. Roy. “An analysis of temporal-difference learning with function approximation”. In: *IEEE Trans. on Automatic Control* 42.5 (1997), pp. 674–690.
- 28 [Tra+19] D. Tran, K. Vafa, K. K. Agrawal, L. Dinh, and B. Poole. “Discrete Flows: Invertible Generative Models of Discrete Data”. In: *Advances in Neural Information Processing Systems*. 2019.
- 29 [Tra+20a] L. Tran, B. S. Veeling, K. Roth, J. Swiatkowski, J. V. Dillon, J. Snoek, S. Mandt, T. Salimans, S. Nowozin, and R. Jefton. “Hydra: Preserving Ensemble Diversity for Model Distillation”. In: (Jan. 2020). arXiv: 2001.04694 [cs.LG].
- 30 [Tra+20b] M.-N. Tran, N. Nguyen, D. Nott, and R. Kohn. “Bayesian Deep Net GLM and GLMM”. In: *J. Comput. Graph. Stat.* 29.1 (Jan. 2020), pp. 97–113.
- 31 [TRB16] D. Tran, R. Ranganath, and D. M. Blei. “The Variational Gaussian Process”. In: *ICLR*. 2016.
- 32 [Tri21] K. Triantafyllopoulos. *Bayesian Inference of State Space Models: Kalman Filtering and Beyond*. en. 1st ed. Springer, Nov. 2021.
- 33 [Tsa+18] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker. “Learning to adapt structured output space for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7472–7481.
- 34 [Tsa+19] Y.-H. H. Tsai, S. Bai, M. Yamada, L.-P. Morency, and S. Alshabani. “Transformer Dissection: An Unified Understanding for Transformer’s Attention via the Lens of Kernel”. In: *EMNLP*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4344–4353.
- 35 [Tsa88] C. Tsallis. “Possible generalization of Boltzmann-Gibbs statistics”. In: *J. of Statistical Physics* 52 (1988), pp. 479–487.
- 36 [Tsc+14] S. Tschiatschek, R. Iyer, H. Wei, and J. Bilmes. “Learning Mixtures of Submodular Functions for Image Collection Summarization”. In: *NIPS*. Montreal, Canada. 2014.
- 37 [Tsi+17] P. A. Tsividis, T. Poucet, J. L. Xu, J. B. Tenenbaum, and S. J. Gershman. “Human Learning in Atari”. en. In: *AAAI Spring Symposium Series*. Mar. 2017.
- 38 [TT13] E. G. Tabak and C. V. Turner. “A family of nonparametric density estimation algorithms”. In: *Communications on Pure and Applied Mathematics* 66.2 (2013), pp. 145–164.
- 39 [TT17] B. Trippe and R. Turner. “Overpruning in Variational Bayesian Neural Networks”. In: *NIPS Workshop on Advances in Approximate Bayesian Inference*. 2017.
- 40 [Tuc+19] G. Tucker, D. Lawson, B. Dai, and R. Ranganath. “Revisiting auxiliary latent variables in generative models”. In: *(2019)*.
- 41 [TUI17] T. Taketomi, H. Uchiyama, and S. Ikeda. “Visual SLAM algorithms: a survey from 2010 to 2016”. en. In: *IPSJ Transactions on Computer Vision and Applications* 9.1 (June 2017), p. 16.
- 42 [Tul+18] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. “MocoGAN: Decomposing motion and content for video generation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1526–1535.
- 43 [Tur+08] R. Turner, P. Berkes, M. Sahani, and D. Mackay. *Counterexamples to variational free energy compactness folk theorems*. Tech. rep. U. Cambridge, 2008.
- 44 [TVE10] E. G. Tabak and E. Vanden-Eijnden. “Density estimation by dual ascent of the log-likelihood”. In: *Communications in Mathematical Sciences* 8.1 (2010), pp. 217–233.
- 45 [TW16] J. M. Tomczak and M. Welling. “Improving variational auto-encoders using Householder flow”. In: *NeurIPS Workshop on Bayesian Deep Learning* (2016).
- 46 [TX00] J. B. Tenenbaum and F. Xu. “Word learning as Bayesian inference”. In: *Proc. 22nd Annual Conf. of the Cognitive Science Society*. 2000.
- 47 [TZ02] Z. Tu and S. Zhu. “Image Segmentation by Data-Driven Markov Chain Monte Carlo”. In: *IEEE PAMI* 24.5 (2002), pp. 657–673.
- 48 [Tze+17] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. “Adversarial discriminative domain adaptation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7167–7176.
- 49 [UCS17] S. Ubaru, J. Chen, and Y. Saad. “Fast Estimation of  $tr(f(A))$  via Stochastic Lanczos Quadrature”. In: *SIAM J. Matrix Anal. Appl.* 38.4 (Jan. 2017), pp. 1075–1099.
- 50 [Ude+16] M. Udell, C. Horn, R. Zadeh, and S. Boyd. “Generalized Low Rank Models”. In: *Foundations and Trends in Machine Learning* 9.1 (2016), pp. 1–118.
- 51 [UFF06] R. Urtasun, D. Fleet, and P. Fua. “3D people tracking with Gaussian process dynamical models”. In: *CVPR*. 2006.
- 52 [UHJ20] M. Uehara, J. Huang, and N. Jiang. “Minimax Weight and Q-Function Learning for Off-Policy Evaluation”. In: *ICML*. 2020.
- 53 [UML13] B. Uria, I. Murray, and H. Larochelle. “RNADE: The real-valued neural autoregressive density-estimator”. In: *NIPS*. 2013.
- 54 [UML14] B. Uria, I. Murray, and H. Larochelle. “A Deep and Tractable Density Estimator”. In: *ICML*. 2014.
- 55 [UN98] N. Ueda and R. Nakano. “Deterministic annealing EM algorithm”. In: *Real Networks* 11 (1998), pp. 271–282.
- 56 [Urt16] B. Ustun and C. Rudin. “Supersparse linear integer models for optimized medical scoring systems”. In: *Machine Learning* 102.3 (2016), pp. 349–391.

- 1
- [Uri+16] B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle. "Neural Autoregressive Distribution Estimation". In: *JMLR* (May 2016).
- 2
- [UTR14] B. Ustun, S. Tracà, and C. Rudin. *Supersparse Linear Integer Models for Interpretable Classification*. 2014. arXiv: 1306.6677 [stat.ML].
- 3
- [Uur+17] V. Uurtio, J. M. Monteiro, J. Kandola, J. Shawe-Taylor, D. Fernandez-Reyes, and J. Rousu. "A Tutorial on Canonical Correlation Methods". In: *ACM Computing Surveys* (2017).
- 4
- [UVL16] D. Ulyanov, A. Vedaldi, and V. Lempitsky. "Instance Normalization: The Missing Ingredient for Fast Stylization". In: (2016). arXiv: 1607.08022 [cs.CV].
- 5
- [UVL18] D. Ulyanov, A. Vedaldi, and V. Lempitsky. "Deep Image Prior". In: *CVPR*. 2018.
- 6
- [VA15] L. Vandenberghe and M. S. Andersen. "Chordal Graphs and Semidefinite Optimization". In: *Foundations and Trends in Optimization* 1.4 (2015), pp. 241–433.
- 7
- [Vaa00] A. W. Van der Vaart. *Asymptotic statistics*. Vol. 3. Cambridge university press, 2000.
- 8
- [Val00] H. Valpola. "Bayesian Ensemble Learning for Nonlinear Factor Analysis" PhD thesis. Helsinki University of Technology, 2000.
- 9
- [Van10] J. Vanhatalo. "Speeding up the inference in Gaussian process models". PhD thesis. Helsinki Univ. Technology, 2010.
- 10
- [van+18] H. van Hasselt, Y. Doron, F. Strub, M. Hessel, N. Sonnerat, and J. Modayil. *Deep Reinforcement Learning and the Deadly Triad*. arXiv:1812.02648. 2018.
- 11
- [Vas+17a] A. B. Vasudevan, M. Gygli, A. Volokitin, and L. Van Gool. "Query-adaptive video summarization via quality-aware relevance estimation". In: *Proceedings of the 25th ACM international conference on Multimedia*. 2017, pp. 582–590.
- 12
- [Vas+17b] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention is all you need". In: *NIPS*. 2017, pp. 5998–6008.
- 13
- [Vas+17c] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention Is All You Need". In: *NIPS*. 2017.
- 14
- [Vaz+22] S. Vaze, K. Han, A. Vedaldi, and A. Zisserman. "Open-Set Recognition: A Good Closed-Set Classifier is All You Need". In: *ICLR*. 2022.
- 15
- [VBW15] S. S. Villar, J. Bowden, and J. Wason. "Multi-armed Bandit Models for the Optimal Design of Clinical Trials: Benefits and Challenges". en. In: *Stat. Sci.* 30.2 (2015), pp. 199–215.
- 16
- [Ved+18] R. Vedantam, I. Fischer, J. Huang, and K. Murphy. "Generative Models of Visually Grounded Imagination". In: *ICLR*. 2018.
- 17
- [Veh+19] A. Vehtari, A. Gelman, D. Simpson, B. Carpenter, and P.-C. Bürkner. "Rank-normalization, folding, and localization: An improved  $\hat{R}$  for assessing convergence of MCMC". In: (Mar. 2019). arXiv: 1903.08008 [stat.CO].
- 18
- [Vei+21] V. Veitch, A. D'Amour, S. Yadlowsky, and J. Eisenstein. "Counterfactual Invariance to Spurious Correlations: Why and How to Pass Stress Tests". In: *Advances in Neural Information Processing Systems*.
- 19
- [Vel+17] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. "Graph attention networks". In: *arXiv preprint arXiv:1710.10903* (2017).
- 20
- [Vel+18] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. "Graph attention networks". In: *ICLR*. 2018.
- 21
- [Ver18] R. Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. en. 1 edition. Cambridge University Press, Sept. 2018.
- 22
- [Ver+19] A. Vergari, A. Molina, R. Peharz, Z. Ghahramani, K. Kersting, and I. Valera. "Automatic Bayesian Density Analysis". In: *AAAI*. 2019.
- 23
- [VF+80] B. C. Van Fraassen et al. *The scientific image*. Oxford University Press, 1980.
- 24
- [VG+09] L. Van Gool, M. D. Breitenstein, F. Reichlin, B. Leibe, and E. Koller-Meier. "Robust Tracking-by-Detection using a Detector Confidence Particle Filter". In: *ICCV*. 2009.
- 25
- [VGG17] A. Vehtari, A. Gelman, and J. Gabry. "Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC". In: *Stat. Comput.* 27.5 (Sept. 2017), pp. 1413–1432.
- 26
- [VGS05] V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic Learning in a Random World*. en. 2005th ed. Springer, Mar. 2005.
- 27
- [Vid99] P. Vidoni. "Exponential Family State Space Models Based on a Conjugate Latent Process". In: *J. R. Stat. Soc. Series B Stat. Methodol.* 61.1 (1999), pp. 213–221.
- 28
- [Vil08] C. Villani. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2008.
- 29
- 30
- 31
- 32
- 33
- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44
- 45
- 46
- 47
- [Vil+19] R. Villegas, A. Pathak, H. Kannan, D. Erhan, Q. V. Le, and H. Lee. "High Fidelity Video Prediction with Large Stochastic Recurrent Neural Networks". In: *NIPS*. 2019.
- [Vin+10] M. Vinyals, J. Cerdeira, J. Rodriguez-Aguilar, and A. Farinelli. "Worst-case bounds on the quality of max-product fixed-points". In: *NIPS*. 2010.
- [Vin11] P. Vincent. "A connection between score matching and denoising autoencoders". In: *Neural computation* 23.7 (2011), pp. 1661–1674.
- [Vir10] S. Virtanen. "Bayesian exponential family projections". MA thesis. Aalto University, 2010.
- [Vis+06] S. V. N. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. P. Murphy. "Accelerated training of conditional random fields with stochastic gradient methods". In: *ICML*. Pittsburgh, Pennsylvania: ACM Press, 2006.
- [Vis+10] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. "Graph Kernels". In: *JMLR* 11 (2010), pp. 1201–1242.
- [Vit67] A. Viterbi. "Error bounds for convolutional codes and an asymptotically optimal decoding algorithm". In: *IEEE Trans. on Information Theory* 13.2 (1967), pp. 260–269.
- [VK20] A. Vahdat and J. Kautz. "NVAE: A Deep Hierarchical Variational Autoencoder". In: *NIPS*. 2020.
- [VLT21] G. M. van de Ven, Z. Li, and A. S. Tolias. "Class-Incremental Learning with Generative Classifiers". In: *CVPR workshop on Continual Learning in Computer Vision (CLVi-sion)*. Apr. 2021.
- [Vo+15] B.-N. Vo, M. Mallick, Y. Bar-Shalom, S. Coraluppi, R. Osborne, R. Mahler, B. t Vo, and J. Webster. *Multitarget tracking*. John Wiley and Sons, 2015.
- [Von13] P. Vontobel. "The Bethe permanent of a non-negative matrix". In: *IEEE Trans. Info. Theory* 59.3 (2013).
- [Vov13] V. Vovk. "Kernel Ridge Regression". In: *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*. Ed. by B. Schölkopf, Z. Liu, and V. Vovk. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 105–116.
- [VPV10] J. Vanhatalo, P. Piiroinen, and A. Vehtari. "Approximate inference for disease mapping with sparse Gaussian processes". In: *Statistics in Medicine* 29.15 (2010), pp. 1580–1607.
- [vR11] M. van der Laan and S. Rose. *Targeted Learning: Causal Inference for Observational and Experimental Data*. Jan. 2011.
- [VRF11] C. Varin, N. Reid, and D. Firth. "An overview of composite likelihood methods". In: *Stat. Sin.* 21.1 (2011), pp. 5–42.
- [VT11] S. I. VanderWeele TJ. "A new criterion for confounder selection". In: *Biometrics* (2011).
- [VT18] G. M. van de Ven and A. S. Tolias. "Three scenarios for continual learning". In: *NeurIPS Continual Learning workshop*. 2018.
- [Vyt+19] D. Vytiniotis, D. Belov, R. Wei, G. Plotkin, and M. Abadi. "The differentiable curry". In: *NeurIPS 2019 Workshop Program Transformations*. 2019.
- [VZ20] V. Veitch and A. Zaveri. "Sense and Sensitivity Analysis: Simple Post-Hoc Analysis of Bias Due to Unobserved Confounding". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 10999–11009.
- [WA13] A. Wilson and R. Adams. "Gaussian process kernels for pattern discovery and extrapolation". In: *International conference on machine learning*. 2013, pp. 1067–1075.
- [Wan+16] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas. "Dueling Network Architectures for Deep Reinforcement Learning". In: *ICML*. 2016.
- [Wan17] M. P. Wand. "Fast Approximate Inference for Arbitrarily Large Semiparametric Regression Models via Message Passing". In: *J. Am. Stat. Assoc.* 112.517 (Jan. 2017), pp. 137–168.
- [Wan+17a] T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, and P. MacNeille. "A bayesian framework for learning rule sets for interpretable classification". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 2357–2393.
- [Wan+17b] X. Wang, T. Li, S. Sun, and J. M. Corchado. "A Survey of Recent Advances in Particle Filters and Remaining Challenges for Multitarget Tracking". en. In: *Sensors* 17.12 (Nov. 2017).
- [Wan+17c] Y. Wang et al. "Tacotron: Towards End-to-End Speech Synthesis". In: *Interspeech*. 2017.
- [Wan+18] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro. "Video-to-video synthesis". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 2018, pp. 1152–1164.
- [Wan+19a] K. Wang, G. Pleiss, J. Gardner, S. Tyree, K. Q. Weinberger, and A. G. Wilson. "Exact Gaussian Processes on a Million Data Points". In: *NIPS*. 2019, pp. 14622–14632.

- 1
- 2 [Wan+19b] S. Wang, W. Bai, C. Lavania, and J. Bilmes. "Fixing  
3 Mini-batch Sequences with Hierarchical Robust Partitioning".  
4 In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. Ed. by K. Chaudhuri and M. Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, 2019, pp. 3352–3361.
- 5 [Wan+19c] Y. Wang, A. Smola, D. C. Maddix, J. Gasthaus, D.  
Foster, and T. Januschowski. "Deep Factors for Forecasting".  
In: *ICML*, 2019.
- 6 [Wan+20a] H. Wang, X. Wu, Z. Huang, and E. P. Xing. "High-  
frequency component helps explain the generalization of con-  
volutional neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.  
2020, pp. 8684–8694.
- 7 [Wan+20b] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros.  
*Dataset Distillation*. 2020. arXiv: 1811.10589 [cs.LG].
- 8 [Wan+20c] Z. Wang, S. Cheng, L. Yuero, J. Zhu, and B. Zhang.  
"A Wasserstein Minimum Velocity Approach to Learning Un-  
normalized Models". In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*.  
Ed. by S. Chiappa and R. Calandri. Vol. 108. Proceedings of Machine Learning Research. Online: PMLR, 2020, pp. 3728–3738.
- 9 [Wan+21] J. Wang, C. Lan, C. Liu, Y. Ouyang, W. Zeng, and  
T. Qin. "Generalizing to Unseen Domains: A Survey on Domain Generalization". In: *IJCAI*. Mar. 2021.
- 10 [Was06] L. Wasserman. *All of Nonparametric Statistics*.  
Springer, 2006.
- 11 [Wat10] S. Watanabe. "Asymptotic Equivalence of Bayes Cross  
Validation and Widely Applicable Information Criterion in Sin-  
gular Learning Theory". In: *JMLR* 11 (Dec. 2010), pp. 3571–  
3594.
- 12 [Wat13] S. Watanabe. "A Widely Applicable Bayesian Informa-  
tion Criterion". In: *JMLR* 14 (2013), pp. 867–897.
- 13 [Wat60] S. Watanabe. "Information theoretical analysis of mul-  
tivariate correlation". In: *IBJ. J. of Research and Development*  
4 (1960), pp. 66–82.
- 14 [WB05] J. Winn and C. Bishop. "Variational Message Passing".  
In: *JMLR* 6 (2005), pp. 661–694.
- 15 [WB12] C. Wang and D. M. Blei. "Truncation-free online vari-  
ational inference for Bayesian nonparametric models". In:  
*Advances in neural information processing systems*. 2012,  
pp. 413–421.
- 16 [WB20] T. Wang and J. Ba. "Exploring Model-based Planning  
with Policy Networks". In: *ICLR*. 2020.
- 17 [WBC21] Y. Wang, D. M. Blei, and J. P. Cunningham. "Pos-  
terior Collapse and Latent Variable Non-identifiability". In:  
*NIPS*. 2021.
- 18 [WCS08] M. Welling, C. Chemudugunta, and N. Sutter. "Deter-  
ministic Latent Variable Models and their Pitfalls". In: *ICDM*.  
2008.
- 19 [WD92] C. Watkins and P. Dayan. "Q-learning". In: *Machine  
Learning* 8.3 (1992), pp. 279–292.
- 20 [WDN15] A. G. Wilson, C. Dann, and H. Nickisch. "Thoughts  
on Massively Scalable Gaussian Processes". In: *arXiv preprint  
arXiv:1511.01870* (2015). <https://arxiv.org/abs/1511.01870>.
- 21 [Web+17] T. Weber et al. "Imagination-Augmented Agents for  
Deep Reinforcement Learning". In: *NIPS*. 2017.
- 22 [Wei00] Y. Weiss. "Correctness of local probability propagation  
in graphical models with loops". In: *Neural Computation* 12  
(2000), pp. 1–41.
- 23 [Wei+13] K. Wei, Y. Liu, K. Kirchhoff, and J. Bilmes. "Using  
Document Summarization Techniques for Speech Data Subset  
Selection". In: *HLT-NAACL*. 2013, pp. 721–726.
- 24 [Wei+14] K. Wei, Y. Liu, K. Kirchhoff, and J. Bilmes. "Unsuper-  
vised Submodular Subset Selection for Speech Data". In: *Proc.  
IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*.  
Florence, Italy, 2014.
- 25 [Wei+15a] K. Wei, R. Iyer, S. Wang, W. Bai, and J. Bilmes.  
"How to Intelligently Distribute Training Data to Multiple  
Compute Nodes: Distributed Machine Learning via Submod-  
ular Partitioning". In: *Neural Information Processing Society  
(NeurIPS, formerly NIPS) Workshop*. LearningSys Workshop,  
<http://learningsys.org>. Montreal, Canada, 2015.
- 26 [Wei+15b] K. Wei, R. Iyer, S. Wang, W. Bai, and J. Bilmes.  
"Mixed Robust/Average Submodular Partitioning: Fast Algo-  
rithms, Guarantees, and Applications". In: *Neural Information  
Processing Society (NeurIPS, formerly NIPS)*. Montreal,  
Canada, 2015.
- 27 [Wei+22] J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B.  
Lester, N. Du, A. M. Dai, and Q. V. Le. "Finetuned Language  
Models are Zero-Shot Learners". In: *ICLR*. 2022.
- 28 [Wei11] M. Welling. "Bayesian Learning via Stochastic Gradient  
Langevin Dynamics". In: *ICML*. 2011.
- 29 [Wen+17] R. Wen, K. Torkkola, B. Narayanaswamy, and D.  
Madeka. "A Multi-Horizon Quantile Recurrent Forecaster". In:  
*NIPS Time Series Workshop*. 2017.
- 30 [Wen+19a] L. Wenliang, D. Sutherland, H. Strathmann, and A.  
Gretton. "Learning deep kernels for exponential family densities". In: *International Conference on Machine Learning*. 2019,  
pp. 6737–6746.
- 31 [Wen+19b] F. Wenzel, T. Galy-Fajou, C. Donner, M. Kloft,  
and M. Opper. "Efficient Gaussian Process Classification Us-  
ing Polya-Gamma Data Augmentation". In: *AAAI*. 2019.
- 32 [Wen+20a] C. Wendler, A. Amrollahi, B. Seifert, A. Krause,  
and M. Püschel. "Learning set functions that are sparse  
in non-orthogonal Fourier bases". In: *arXiv preprint  
arXiv:2010.00439* (2020).
- 33 [Wen+20b] F. Wenzel, K. Roth, B. S. Veeling, J. Świątkowski,  
L. Tran, S. Mandt, J. Snoek, T. Salimans, R. Jenatton, and  
S. Nowozin. "How Good is the Bayes Posterior in Deep Neural  
Networks Really?". In: *ICML*. 2020.
- 34 [Wen+20c] F. Wenzel, J. Snoek, D. Tran, and R. Jenatton. "Hy-  
perparameter Ensembles for Robustness and Uncertainty Quan-  
tification". In: *NIPS*. 2020.
- 35 [Wen21] L. Weng. "What are diffusion models?". In:  
[lilianweng.github.io/tlk-log](https://lilianweng.github.io/tlk-log) (2021).
- 36 [Wes03] M. West. "Bayesian Factor Regression Models in the  
"Large p, Small n" Paradigm". In: *Bayesian Statistics 7* (2003).
- 37 [Wes87] M. West. "On scale mixtures of normal distributions".  
In: *Biometrika* 74 (1987), pp. 646–648.
- 38 [WF01a] Y. Weiss and W. T. Freeman. "Correctness of belief  
propagation in Gaussian graphical models of arbitrary topol-  
ogy". In: *Neural Computation* 13.10 (2001), pp. 2173–2200.
- 39 [WF01b] Y. Weiss and W. T. Freeman. "On the optimality of  
solutions of the max-product belief propagation algorithm in  
arbitrary graphs". In: *IEEE Trans. Information Theory, Spec-  
cial Issue on Codes on Graphs and Iterative Algorithms* 47.2  
(2001), pp. 723–735.
- 40 [WF14] Z. Wang and N. de Freitas. "Theoretical Analysis of  
Bayesian Optimisation with Unknown Gaussian Process Hyper-  
Parameters". In: (June 2014). arXiv: 1406.7758 [stat.ML].
- 41 [WF16] Z. Wang and N. de Freitas. "Theoretical Analysis of  
Bayesian Optimisation with Unknown Gaussian Process Hyper-  
Parameters". In: *BayesOpt Workshop*. 2016.
- 42 [WF18] M. Wu and N. Goodman. "Multimodal Generative Mod-  
els for Scalable Weakly-Supervised Learning". In: *NIPS*. Feb.  
2018.
- 43 [WGY21] G. Weiss, Y. Goldberg, and E. Yahav. "Thinking Like  
Transformers". In: *ICML*. June 2021.
- 44 [WHD02] M. Welling and G. E. Hinton. "A new learning algo-  
rithm for mean field Boltzmann machines". In: *International  
Conference on Artificial Neural Networks*. Springer, 2002,  
pp. 351–357.
- 45 [WH18] Y. Wu and K. He. "Group Normalization". In: *ECCV*.  
2018.
- 46 [WH97] M. West and J. Harrison. *Bayesian forecasting and dy-  
namic models*. Springer, 1997.
- 47 [WHD18] J. T. Wilson, F. Hutter, and M. P. Deisenroth. "Max-  
imizing acquisition functions for Bayesian optimization". In:  
*NIPS*. 2018.
- 48 [WHF06] J. Wang, A. Hertzmann, and D. J. Fleet. "Gaus-  
sian Process Dynamical Models". In: *NIPS*. MIT Press, 2006,  
pp. 1441–1448.
- 49 [Whi16] T. White. "Sampling Generative Networks". In: *arXiv*  
(2016).
- 50 [Whi88] P. Whittle. "Restless bandits: activity allocation in a  
changing world". In: *J. Appl. Probab.* 25.A (1988), pp. 287–  
298.
- 51 [WHT19] Y. Wang, H. He, and X. Tan. "Truly Proximal Policy  
Optimization". In: *UAI*. 2019.
- 52 [WI20] A. G. Wilson and P. Izmailov. "Bayesian Deep Learning  
and a Probabilistic Perspective of Generalization". In: *NIPS*.  
Feb. 2020.
- 53 [WIB15] K. Wei, R. Iyer, and J. Bilmes. "Submodularity in  
Data Subset Selection and Active Learning". In: *Proceedings of  
the 32nd international conference on Machine learning*. Lille,  
France, 2015.
- 54 [Wik21] Wikipedia contributors. *CliffsNotes — Wikipedia, The  
Free Encyclopedia*. [Online; accessed December 29–2021]. 2021.
- 55 [Wil+14] A. G. Wilson, E. Gilboa, A. Nehorai, and J. P. Cun-  
ningham. "Fast kernel learning for multidimensional pattern  
extrapolation". In: *Advances in Neural Information Process-  
ing Systems*. 2014, pp. 3626–3634.
- 56 [Wil14] A. G. Wilson. "Covariance kernels for fast automatic  
pattern discovery and extrapolation with Gaussian processes".  
PhD thesis. University of Cambridge, 2014.
- 57 [Wil+16a] A. G. Wilson, Z. Hu, R. R. Salakhutdinov, and E. P.  
Xing. "Stochastic Variational Deep Kernel Learning". In: *NIPS*.  
Curran Associates, Inc., 2016, pp. 2586–2594.

- [Wil+16b] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, “Deep Kernel Learning”. In: *AISTATS*. May 2016, pp. 370–378.
- [Wil+17] A. G. Wills, J. Hendriks, C. Renton, and B. Ninness, “A Bayesian Filtering Algorithm for Gaussian Mixture Models”. In: (May 2017). arXiv: 1705.06495 [stat.ML].
- [Wil20] A. G. Wilson, “The Case for Bayesian Deep Learning”. In: (Jan. 2020). arXiv: 2001.10995 [cs.LG].
- [Wil+20a] J. T. Wilson, V. Borovitskiy, A. Terenin, P. Mostowsky, and M. P. Deisenroth, “Efficiently Sampling Functions from Gaussian Process Posterior”. In: *ICML*. 2020.
- [Wil+20b] A. B. Wiltschko, B. Sanchez-Lengeling, B. Lee, E. Reif, J. Wei, K. J. McCloskey, L. Colwell, W. Qian, and Y. Wang, “Evaluating Attribution for Graph Neural Networks”. In: (2020).
- [Wil69] A. G. Wilson, “The use of entropy maximising models, in the theory of trip distribution, mode split and route split”. In: *Journal of transport economics and policy* (1969), pp. 108–126.
- [Wil92] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *MLJ* 8.3-4 (1992), pp. 229–256.
- [Wil98] C. Williams, “Computation with infinite networks”. In: *Neural Computation* 10.5 (1998), pp. 1203–1216.
- [Win] J. Winn. *VIBES*.
- [WIP20] J. Watson, A. Imohiosen, and J. Peters, “Active Inference or Control as Inference? A Unifying View”. In: *International Workshop on Active Inference*. Oct. 2020.
- [Wit] DEEPFAKES: PREPARE NOW (PERSPECTIVES FROM BRAZIL). <https://lab.witness.org/brazil-deepfakes-prepare-now/>. Accessed: 2021-08-18.
- [Wiy+19] R. R. Wiyatno, A. Xu, O. Dia, and A. de Berker, “Adversarial Examples in Modern Machine Learning: A Review”. In: (Nov. 2019). arXiv: 1911.05268 [cs.LG].
- [WJ08] M. J. Wainwright and M. I. Jordan, “Graphical models, exponential families, and variational inference”. In: *Foundations and Trends in Machine Learning* 1–2 (2008), pp. 1–305.
- [WJW03] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, “Tree-based reparameterization framework for analysis of sum-product and related algorithms”. In: *IEEE Trans. on Information Theory* 49.5 (2003), pp. 1120–1146.
- [WK18] E. Wong and Z. Kolter, “Provable defenses against adversarial examples via the convex outer adversarial polytope”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5286–5295.
- [WK96] G. Widmer and M. Kubat, “Learning in the presence of concept drift and hidden contexts”. In: *Mach. Learn.* 23.1 (Apr. 1996), pp. 69–101.
- [WKS21] V. Wild, M. Kanagawa, and D. Sejdinovic, “Connections and Equivalences between the Nyström Method and Sparse Variational Gaussian Processes”. In: (June 2021). arXiv: 2106.01121 [stat.ML].
- [WL14] J. L. Williams and R. A. Lau, “Approximate evaluation of marginal association probabilities with belief propagation”. In: *IEEE Trans. Aerosp. Electron. Syst.* 50.4 (2014).
- [WL19] A. Wehenkel and G. Louppe, “Unconstrained Monotonic Neural Networks”. In: *NIPS*. 2019.
- [WLL16] W. Wang, H. Lee, and K. Livescu, “Deep Variational Canonical Correlation Analysis”. In: *arXiv* (Nov. 2016).
- [WLZ19] D. Widmann, F. Lindsten, and D. Zachariah, “Calibration tests in multi-class classification: A unifying framework”. In: *NIPS*. Curran Associates, Inc., 2019, pp. 12236–12246.
- [WM01] E. A. Wan and R. V. der Merwe, “The Unscented Kalman Filter”. In: *Kalman Filtering and Neural Networks*. Ed. by S. Haykin. Wiley, 2001.
- [WM12] K. Wakabayashi and T. Miura, “Forward-Backward Activation Algorithm for Hierarchical Hidden Markov Models”. In: *NIPS*. 2012.
- [WMR17] S. Wachter, B. Mittelstadt, and C. Russell, “Counterfactual explanations without opening the black box: Automated decisions and the GDPR”. In: *Harv. JL & Tech.* 31 (2017), p. 841.
- [WMR18] S. Wachter, B. Mittelstadt, and C. Russell, *Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR*. 2018. arXiv: 1711.00399 [cs.AI].
- [WN07] D. Wipf and S. Nagarajan, “A new view of automatic relevance determination”. In: *NIPS*. 2007.
- [WN10] D. Wipf and S. Nagarajan, “Iterative Reweighted  $\ell_1$  and  $\ell_2$  Methods for Finding Sparse Solutions”. In: *J. of Selected Topics in Signal Processing (Special Issue on Compressive Sensing)* 4.2 (2010).
- [WN15] A. G. Wilson and H. Nickisch, “Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP)”. In: *ICML*. Lille, France: JMLR.org, 2015, pp. 1775–1784.
- [WN18] C. K. I. Williams and C. Nash, “Autoencoders and Probabilistic Inference with Missing Data: An Exact Solution for The Factor Analysis Case”. In: (Jan. 2018). arXiv: 1801.03851 [cs.LG].
- [Wol12] M. Wiering and M. van Otterlo, eds. *Reinforcement learning: State-of-the-art*. Springer, 2012.
- [Wol21] M. Wołczyk, M. Zajac, R. Pascanu, Ł. Kuciński, and P. Miśoś, “Continual World: A Robotic Benchmark for Continual Reinforcement Learning”. In: *NIPS*. May 2021.
- [Wol76] P. Wolfe, “Finding the nearest point in a polytope”. In: *Mathematical Programming* 11 (1976), pp. 128–149.
- [Wol92] D. Wolpert, “Stacked Generalization”. In: *Neural Networks* 5.2 (1992), pp. 241–259.
- [Woo+09] F. Wood, C. Archambeau, J. Gasthaus, L. James, and Y. W. Teh, “A Stochastic Memoizer for Sequence Data”. In: *ICML*. 2009.
- [Woo+11] F. Wood, J. Gasthaus, C. Archambeau, L. James, and Y. W. Teh, “The sequence memoizer”. In: *Comm. of the ACM* 54.2 (2011), pp. 91–98.
- [Woo+19] B. Woodworth, S. Gunasekar, P. Savarese, E. Moroshko, I. Golan, J. Lee, D. Soudry, and N. Srebro, “Kernel and Rich Regimes in Overparametrized Models”. In: (June 2019). arXiv: 1906.05827 [cs.LG].
- [Woo+20] F. Wood, A. Warrington, S. Naderiparizi, C. Weilbach, V. Masrani, W. Harvey, A. Scibior, B. Beronov, and A. Nasseri, *Planning as Inference in Epidemiological Models*. arXiv:2003.13221. 2020.
- [WP19] S. Wiegreffe and Y. Pinter, “Attention is not not explanation”. In: *arXiv preprint arXiv:1908.04626* (2019).
- [WR15] F. Wang and C. Rudin, “Falling Rule Lists”. In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. Ed. by G. Lebanon and S. V. N. Vishwanathan. Vol. 38. Proceedings of Machine Learning Research. San Diego, California, USA: PMLR, 2015, pp. 1013–1022.
- [WRN10] D. Wipf, B. Rao, and S. Nagarajan, “Latent Variable Bayesian Models for Promoting Sparsity”. In: *IEEE Transactions on Information Theory* (2010).
- [WRZH04] M. Welling, M. Rosen-Zvi, and G. Hinton, “Exponential family harmoniums with an application to information retrieval”. In: *NIPS-14*. 2004.
- [WS01] C. K. I. Williams and M. Seeger, “Using the Nyström Method to Speed Up Kernel Machines”. In: *NIPS*. Ed. by T. K. Leen, T. G. Dietterich, and V. Tresp. MIT Press, 2001, pp. 682–688.
- [WS05] M. Welling and C. Sutton, “Learning in Markov Random Fields with Contrastive Free Energies”. In: *Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS)*. 2005.
- [WS93] D. Wolpert and C. Strauss, “What Bayes has to say about the evidence procedure”. In: *Proc. Workshop on Maximum Entropy and Bayesian methods*. 1993.
- [WSG21] C. Wang, S. Sun, and R. Grosse, “Beyond Marginal Uncertainty: How Accurately can Bayesian Regression Models Estimate Posterior Predictive Correlations”. In: *AISTATS*. Ed. by A. Banerjee and K. Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, 2021, pp. 2476–2484.
- [WSN00] B. Williams, T. Santner, and W. Notz, “Sequential design of computer experiments to minimize integrated response functions”. In: *Statistica Sinica* 10 (2000), pp. 1133–1152.
- [WT01] M. Welling and Y.-W. Teh, “Belief Optimization for Binary Networks: Stable Alternative to Loopy Belief Propagation”. In: *UAI*. 2001.
- [WT19] R. Wen and K. Torkkola, “Deep Generative Quantile-Copula Models for Probabilistic Forecasting”. In: *ICML*. 2019.
- [WT90] G. Wei and M. Tanner, “A Monte Carlo implementation of the EM algorithm and the poor man’s data augmentation algorithms”. In: *JASA* 85.411 (1990), pp. 699–704.
- [WTB20] Y. Wen, D. Tran, and J. Ba, “BatchEnsemble: an Alternative Approach to Efficient Ensemble and Lifelong Learning”. In: *ICLR*. 2020.
- [WTN19] Y. Wu, G. Tucker, and O. Nachum, *Behavior Regularized Offline Reinforcement Learning*. arXiv:1911.11361. 2019.
- [Wu+06] Y. Wu, D. Hu, M. Wu, and X. Hu, “A Numerical-Integration Perspective on Gaussian Filters”. In: *IEEE Trans. Signal Process.* 54.8 (Apr. 2006), pp. 2910–2921.
- [Wu+17] Y. Wu, E. Mansimov, S. Liao, R. Grosse, and J. Ba, “Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation”. In: *NIPS*. 2017.
- [Wu+19a] A. Wu, S. Nowozin, E. Meeds, R. E. Turner, J. M. Hernández-Lobato, and A. L. Gaunt, “Fixing Variational Bayes: Deterministic Variational Inference for Bayesian Neural Networks”. In: *ICLR*. 2019.

- 1
- 2 [Wu+19b] M. Wu, S. Parbhoo, M. Hughes, R. Kindle, L. Celi,  
M. Zazzi, V. Roth, and F. Doshi-Velez, “Regional tree regularization for interpretability in black box models”. In: *AAAI* (2019).
- 3
- 4 [Wu+21] T. Wu, M. T. Ribeiro, J. Heer, and D. S. Weld, “Polyjuice: Generating Counterfactuals for Explaining, Evaluating, and Improving Models”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics: Association for Computational Linguistics*, 2021.
- 5
- 6 [Wü+16] M. Wüthrich, S. Trimpe, C. García Cifuentes, D. Kappeler, and S. Schaal, “A new perspective and extension of the Gaussian Filter”, en: *The International Journal of Robotics Research* 35.14 (Dec. 2016), pp. 1731–1749.
- 7
- 8 [WW12] Y. Wu and D. P. Wipf, “Dual-Space Analysis of the Sparse Linear Model”, In: *NIPS*. 2012.
- 9
- 10 [WY02] D. Wilkinson and S. Yeung, “Conditional simulation from highly structured Gaussian systems with application to blocking-MCMC for the Bayesian analysis of very large linear models”, In: *Statistics and Computing* 12 (2002), pp. 287–300.
- 11
- 12 [WYG14] J. Wen, C.-N. Yu, and R. Greiner, “Robust Learning under Uncertain Test Distributions: Relating Covariate Shift to Model Misspecification”, In: *ICML*. Vol. 32. Proceedings of Machine Learning Research. Beijing, China: PMLR, 2014, pp. 631–639.
- 13
- 14 [WZ19] J. Wei and K. Zou, “EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks”, In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6382–6388.
- 15
- 16 [WZR20] S. Wu, H. R. Zhang, and C. Ré, “Understanding and Improving Information Transfer in Multi-Task Learning”, In: *International Conference on Learning Representations*. 2020.
- 17
- 18 [XC19] Z. Xia and A. Chakrabarti, “Training Image Estimators without Image Ground-Truth”, In: *NIPS*. 2019.
- 19
- 20 [XD18] J. Xu and G. Durrett, “Spherical Latent Spaces for Stable Variational Autoencoders”, In: *EMNLP*. 2018.
- 21 [Xia+21] K. Xiao, L. Engstrom, A. Ilyas, and A. Madry, “Noise or Signal: The Role of Image Backgrounds in Object Recognition”, In: *ICLR*. 2021.
- 22 [Xie+16] J. Xie, Y. Lu, S.-C. Zhu, and Y. N. Wu, “A Theory of Generative ConvNet”, In: *ICML*. 2016.
- 23 [Xie+18] J. Xie, Y. Lu, R. Gao, and Y. N. Wu, “Cooperative Learning of Energy-Based Model and Latent Variable Model via MCMC Teaching”, In: *AAAI*. Vol. 1. 6. 2018, p. 7.
- 24
- 25 [XJ96] L. Xu and M. I. Jordan, “On Convergence Properties of the EM Algorithm for Gaussian Mixtures”, In: *Neural Computation* 8 (1996), pp. 129–151.
- 26
- 27 [Xu+06] Z. Xu, V. Tresp, K. Yu, and H.-P. Kriegel, “Infinite hidden relational models”, In: *UAI*. 2006.
- 28 [Xu+07] Z. Xu, V. Tresp, S. Yu, K. Yu, and H.-P. Kriegel, “Fast Inference in Infinite Hidden Relational Models”, In: *Workshop on Mining and Learning with Graphs*. 2007.
- 29
- 30 [Xu+15] J. Xu, L. Mukherjee, Y. Li, J. Warner, J. M. Rehg, and V. Singh, “Gaze-enabled egocentric video summarization via constrained submodular maximization”, In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 2235–2244.
- 31
- 32 [Xu+19] M. Xu, M. Quiroz, R. Kohn, and S. A. Sisson, “Variance reduction properties of the reparameterization trick”, In: *AISTATS*. Ed. by K. Chaudhuri and M. Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, 2019, pp. 2711–2720.
- 33
- 34 [Yad+18] S. Yadlowsky, H. Namkoong, S. Basu, J. Duchi, and Tian, “Bounds on the conditional and average treatment effect with unobserved confounding factors”, In: *arXiv e-prints*, arXiv:1808.09521 (Aug. 2018), arXiv:1808.09521. arXiv: 1808.09521 [stat.ME].
- 35
- 36 [Yad+21] S. Yadlowsky, S. Fleming, N. Shah, E. Brunskill, and S. Wager, “Evaluating Treatment Prioritization Rules via Rank-Weighted Average Treatment Effects”, 2021. arXiv: 2111.07966 [stat.ME].
- 37
- 38 [Yan+17] S. Yang, L. Xie, X. Chen, X. Lou, X. Zhu, D. Huang, and H. Li, “Statistical parametric speech synthesis using generative adversarial networks under a multi-task learning framework”, In: *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. Dec. 2017, pp. 685–691.
- 39
- 40
- 41 [Yan19] G. Yang, “Scaling Limits of Wide Neural Networks with Weight Sharing: Gaussian Process Behavior, Gradient Independence, and Neural Tangent Kernel Derivation”, In: (Feb. 2019). arXiv: 1902.04760 [cs.NE].
- 42
- 43 [Yan+21] J. Yang, K. Zhou, Y. Li, and Z. Liu, “Generalized OOD Detection: A Survey”, In: (2021).
- 44
- 45 [Yan81] M. Yannakakis, “Computing the minimum fill-in is NP-complete”, In: *SIAM J. Alg. Discrete Methods* 2 (1981), pp. 77–79.
- 46
- 47 [Yao+18a] Y. Yao, A. Vehtari, D. Simpson, and A. Gelman, “Using Stacking to Average Bayesian Predictive Distributions (with Discussion)”, en. In: *Bayesian Analysis* 13.3 (Sept. 2018), pp. 917–1007.
- 48 [Yao+18b] Y. Yao, A. Vehtari, D. Simpson, and A. Gelman, “Yes, but Did It Work?: Evaluating Variational Inference”, In: *ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 5581–5590.
- 49 [YB20] Y. Yang, R. Bamler, and S. Mandt, *Improving Inference for Neural Image Compression*. 2020. arXiv: 2006.04240 [eess.IV].
- 50 [YBS11] P. Yadollahpour, D. Batra, and G. Shakhnarovich, “Diverse M-best Solutions in MRFs”, In: *NIPS workshop on Discrete Optimization in Machine Learning*. 2011.
- 51 [YBW15] F. Yang, S. Balakrishnan, and M. J. Wainwright, “Statistical and Computational Guarantees for the Baum-Welch Algorithm”, In: (2015). arXiv: 1512.08269 [stat.ML].
- 52 [Ye+19] Z. Ye, Q. Guo, Q. Gan, X. Qiu, and Z. Zhang, “BP-Transformer: Modelling Long-Range Context via Binary Partitioning”, In: (Nov. 2019). arXiv: 1911.04070 [cs.CL].
- 53 [Yed11] J. S. Yedidia, “Message-Passing Algorithms for Inference and Optimization”, In: *J. Stat. Phys.* 145.4 (Nov. 2011), pp. 860–890.
- 54 [Yeh+18] C.-K. Yeh, J. S. Kim, I. E. H. Yen, and P. Ravikumar, “Representative Point Selection for Explaining Deep Neural Networks”, In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 10967–10978.
- 55 [Yeh+19b] C.-K. Yeh, B. Kim, S. O. Arik, C.-L. Li, T. Pfister, and P. Ravikumar, “On completeness-aware concept-based explanations in deep neural networks”, In: *arXiv preprint arXiv:1910.07969* (2019).
- 56 [Yeu+17] S. Yeung, A. Kannan, Y. Dauphin, and L. Fei-Fei, “Tackling Over-pruning in Variational Autoencoders”, In: *ICML Workshop on ‘Principled Approaches to Deep Learning’*. June 2017.
- 57 [Yeu91a] R. W. Yeung, “A new outlook on Shannon’s information measures”, In: *IEEE Trans. Inf. Theory* 37.3 (May 1991), pp. 466–474.
- 58 [Yeu91b] R. W. Yeung, “A new outlook on Shannon’s information measures”, In: *IEEE Trans. on Information Theory* 37 (1991), pp. 466–474.
- 59 [YFW00] J. Yedidia, W. T. Freeman, and Y. Weiss, “Generalized Belief Propagation”, In: *NIPS*. 2000.
- 60 [Yin+19a] D. Yin, R. G. Lopes, J. Shlens, E. D. Cubuk, and J. Gilmer, “A Fourier Perspective on Model Robustness in Computer Vision”, In: *NIPS*. 2019.
- 61 [Yin+19b] P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, and J. Xin, “Understanding Straight-through Estimator in Training Activation Quantized Neural Nets”, In: *ICLR*. 2019.
- 62 [Yin+19c] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, “Gnnexplainer: Generating explanations for graph neural networks”, In: *Advances in neural information processing systems* 32 (2019), p. 9240.
- 63 [Yin+20] D. Yin, M. Farajtabar, A. Li, N. Levine, and A. Mott, “Optimization and Generalization of Regularization-Based Continual Learning: a Loss Approximation Viewpoint”, In: (June 2020). arXiv: 2006.10974 [cs.LG].
- 64 [YK06] A. Yuille and D. Kersten, “Vision as Bayesian inference: analysis by synthesis?”, en. In: *Trends Cogn. Sci.* 10.7 (July 2006), pp. 301–308.
- 65 [YK19] M. Yang and B. Kim, *Benchmarking Attribution Methods with Relative Feature Importance*. 2019. arXiv: 1907.09701 [cs.LG].
- 66 [YMT22] Y. Yang, S. Mandt, and L. Theis, “An Introduction to Neural Data Compression”, In: (Feb. 2022). arXiv: 2202.06533 [cs.LG].
- 67 [Yoo+18] K. Yoon, R. Liao, Y. Xiong, L. Zhang, E. Fetaya, R. Urtasun, R. Zemel, and X. Pitkow, “Inference in Probabilistic Graphical Models by Graph Neural Networks”, In: *ICLR Workshop*.
- 68 [You19] A. Young, “Consistency without inference: Instrumental variables in practical application”, In: (2019).
- 69 [Youn91] L. Younes, “Parameter estimation for imperfectly observed Gibbsian fields”, In: *Probab. Theory and Related Fields* 82 (1989), pp. 625–645.
- 70 [Youn99] L. Younes, “On the convergence of Markovian stochastic algorithms with rapidly decreasing ergodicity rates”, In: *Stochastics: An International Journal of Probability and Stochastic Processes* 65.3-4 (1999), pp. 177–228.
- 71 [Yu+06] S. Yu, K. Yu, V. Tresp, K. H.-P., and M. Wu, “Supervised probabilistic principal component analysis”, In: *KDD*. 2006.

- 1
- [Yu10] S.-Z. Yu. "Hidden Semi-Markov Models". In: *Artificial Intelligence J.* 174.2 (2010).
- 2
- [Yu+16] F. X. X. Yu, A. T. Suresh, K. M. Choromanski, D. N. Holtmann-Rice, and S. Kumar, "Orthogonal Random Features". In: *NIPS*. Curran Associates, Inc., 2016, pp. 1795–1893.
- 3
- [Yu+17] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seggan: Sequence generative adversarial nets with policy gradient". In: *Thirty-first AAAI conference on artificial intelligence*. 2017.
- 4
- [Yu+18] Y. Yu et al, "Dynamic Control Flow in Large-Scale Machine Learning". In: *Proceedings of the Thirteenth EuroSys Conference*. EuroSys '18, Porto, Portugal: Association for Computing Machinery, 2018.
- 5
- [Yu+20] L. Yu, Y. Song, J. Song, and S. Ermon, "Training Deep Energy-Based Models with f-Divergence Minimization". In: *arXiv preprint arXiv:2003.03463* (2020).
- 6
- [Yu+21] J. Yu, X. Li, J. Y. Koh, H. Zhang, R. Pang, J. Qin, A. Ku, Y. Xu, J. Baldridge, and Y. Wu, "Vector-quantized Image Modeling with Improved VQGAN". In: (Oct. 2021). arXiv: 2110.04627 [cs.CV].
- 7
- [Yua+19] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial Examples: Attacks and Defenses for Deep Learning", en. In: *IEEE Trans. Neural Networks and Learning Systems* 30.9 (Sept. 2019), pp. 2805–2824.
- 8
- [Yui01] A. Yuille, "CCCP algorithms to minimize the Bethe and Kikuchi free energies: convergent alternatives to belief propagation". In: *Neural Computation* 14 (2001), pp. 1691–1722.
- 9
- [YW04] C. Yanover and Y. Weiss, "Finding the Most Probable Configurations in Arbitrary Graphical Models". In: *NIPS*. 2004.
- 10
- [YWX17] J.-g. Yao, X. Wan, and J. Xiao, "Recent advances in document summarization". In: *Knowledge and Information Systems* 53.2 (2017), pp. 297–336.
- 11
- [YZ19] G. Yaroslavtsev and S. Zhou, "Approximate  $F_2$ -Sketching of Valuation Functions". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*. Ed. by D. Achlioptas and L. A. Végh. Vol. 145. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2019, 69:1–69:21.
- 12
- [YZ22] Y. Yang and P. Zhai, "Click-through rate prediction in online advertising: A literature review". In: *Inf. Process. Manag.* 59.2 (Mar. 2022), p. 102853.
- 13
- [ZA12] J. Zou and R. Adams, "Priors for Diversity in Generative Latent Variable Models". In: *NIPS*. 2012.
- 14
- [Zaf+22] M. Zaffran, A. Dieuleveut, O. Féron, Y. Goude, and J. Josse, "Adaptive Conformal Predictions for Time Series". In: (Feb. 2022). arXiv: 2202.07282 [stat.ML].
- 15
- [Zai+20] S. Zaidi, A. Zela, T. Elsken, C. Holmes, F. Hutter, and Y. W. Teh, "Neural Ensemble Search for Performant and Calibrated Predictions". In: (June 2020). arXiv: 2006.08573 [cs.LG].
- 16
- [Zan21] N. Zanichelli, *IAML Distill Blog: Transformers in Vision*. 2021.
- 17
- [ZB18] T. Zhou and J. Bilmes, "Minimax curriculum learning: Machine teaching with desirable difficulties and scheduled diversity". In: *International Conference on Learning Representations*. 2018.
- 18
- [ZB21] B. Zhao and H. Bilen, "Dataset Condensation with Differentiable Siamese Augmentation". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18–24 July 2021, Virtual Event*. Ed. by M. Meila and T. Z. 0001. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 12674–12685.
- 19
- [ZCC07] G. Zhang, T. Chen, and X. Chen, "Performance Recovery in Digital Implementation of Analogue Systems". In: *SIAM J. Control Optim.* 45.6 (Jan. 2007), pp. 2207–2223.
- 20
- [ZDK15] J. Zhang, J. Djolonga, and A. Krause, "Higher-order inference for multi-class log-supermodular models". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1859–1867.
- 21
- [ZE01a] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers". In: *ICML*. 2001.
- 22
- [ZE01b] B. Zadrozny and C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates". In: *KDD*. 2001.
- 23
- [Zee+18] J. R. Zech, M. A. Badgeley, M. Liu, A. B. Costa, J. J. Titano, and E. K. Oermann, "Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: a cross-sectional study". In: *PLoS medicine* 15.11 (2018), e1002683.
- 24
- [Zel76] A. Zellner, "Bayesian and non-Bayesian analysis of the regression model with multivariate Student-t error terms". In: *JASA* 71.354 (1976), pp. 400–405.
- 25
- [Zel86] A. Zellner, "On assessing prior distributions, and Bayesian regression analysis with g-prior distributions". In: *Bayesian inference and decision techniques, Studies of Bayesian and Econometrics and Statistics volume 6*. North Holland, 1986.
- 26
- [Zen+18] C. Zeno, I. Golan, E. Hoffer, and D. Soudry, "Task Agnostic Continual Learning Using Online Variational Bayes". In: (Mar. 2018). arXiv: 1803.10123 [stat.ML].
- 27
- [Zen+21] C. Zeno, I. Golan, E. Hoffer, and D. Soudry, "Task-Agnostic Continual Learning Using Online Variational Bayes With Fixed-Point Updates". en. In: *Neural Comput.* 33.11 (Oct. 2021), pp. 3139–3177.
- 28
- [Zer+19] J. Zerilli, A. Knott, J. Maclaurin, and C. Gavaghan, "Transparency in algorithmic and human decision-making: is there a double standard?". In: *Philosophy & Technology* 32.4 (2019), pp. 661–683.
- 29
- [ZF14] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer, 2014, pp. 818–833.
- 30
- [ZVF20] G. Zeni, M. Fontana, and S. Vantini, "Conformal Prediction: a Unified Review of Theory and New Challenges". In: (May 2020). arXiv: 2005.07972 [cs.LG].
- 31
- [ZG21] D. Zou and Q. Gu, "On the Convergence of Hamiltonian Monte Carlo with Stochastic Gradients". In: *ICML*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 13012–13022.
- 32
- [ZGH10] X. Zhang, T. Graepel, and R. Herbrich, "Bayesian Online Learning for Multi-label and Multi-variate Performance Measures". In: *AISTATS*. 2010.
- 33
- [ZGR21] L. Zhang, M. Goldstein, and R. Ranganath, "Understanding Failures in Out-of-Distribution Detection with Deep Generative Models". In: *ICML*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 12427–12436.
- 34
- [ZH05] O. Zoeter and T. Heskes, "Gaussian Quadrature Based Expectation Propagation". In: *AISTATS*. 2005.
- 35
- [Zha+13a] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang, "Domain Adaptation under Target and Conditional Shift". In: *Proceedings of the 30th International Conference on Machine Learning*. 2013, pp. 819–827.
- 36
- [Zha+13b] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang, "Domain Adaptation under Target and Conditional Shift". In: *ICML*. Vol. 28. 2013.
- 37
- [Zha+16] S. Zhai, Y. Cheng, R. Feris, and Z. Zhang, "Generative adversarial networks as variational training of energy based models". In: *arXiv preprint arXiv:1611.01799* (2016).
- 38
- [Zha+17] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization". In: *ICLR*. 2017.
- 39
- [Zha+18a] G. Zhang, S. Sun, D. Duvenaud, and R. Grosse, "Noisy Natural Gradient as Variational Inference". In: *ICML*. 2018.
- 40
- [Zha+18b] Y. Zhang, A. M. Saxe, M. S. Advani, and A. A. Lee, "Energy-entropy competition and the effectiveness of stochastic gradient descent in machine learning". In: (Mar. 2018). arXiv: 1803.01927 [cs.LG].
- 41
- [Zha+19a] C. Zhang, J. Butepage, H. Kjellstrom, and S. Mandt, "Advances in Variational Inference". In: *IEEE PAMI* (2019), pp. 2008–2026.
- 42
- [Zha+19b] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks". In: *International conference on machine learning*. PMLR, 2019, pp. 7354–7363.
- 43
- [Zha+19c] L. Zhao, K. Korovina, W. Si, and M. Cheung, "Approximate inference with Graph Neural Networks". 2019.
- 44
- [Zha+20a] A. Zhang, Z. Lipton, M. Li, and A. Smola, *Dive into deep learning*. 2020.
- 45
- [Zha+20b] H. Zhang, A. Li, J. Guo, and Y. Guo, "Hybrid models for open set recognition". In: *European Conference on Computer Vision*. Springer, 2020, pp. 102–117.
- 46
- [Zha+20c] R. Zhang, B. Dai, L. Li, and D. Schuurmans, "Generalized Offline Estimation of Stationary Values". In: *ICLR*. 2020.
- 47
- [Zha+20d] R. Zhang, C. Li, J. Zhang, C. Chen, and A. G. Wilson, "Cyclical stochastic gradient MCMC for Bayesian deep learning". In: *ICLR*. 2020.
- 48
- [Zha+20e] X. Zhang, Y. Li, Z. Zhang, and Z.-L. Zhang, "f-GAIL: Learning f-Divergence for Generative Adversarial Imitation Learning". In: *Neural Information Processing Systems* (2020).
- 49
- [Zha+21] H. Zhang, J. Y. Koh, J. Baldridge, H. Lee, and Y. Yang, "Cross-Modal Contrastive Learning for Text-to-Image Generation". In: *CVPR*. 2021.
- 50
- [Zhe+15] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr, "Conditional Random Fields as Recurrent Neural Networks". In: *ICCV*. 2015.
- 51
- [Zho+19a] S. Zhou, M. Gordon, R. Krishna, A. Narcomey, L. F. Fei-Fei, and M. Bernstein, "HYPE: A Benchmark for Human-eYe Perceptual Evaluation of Generative Models". In: *NIPS*. Curran Associates, Inc., 2019, pp. 3444–3456.

- 1
- 2 [Zho+19b] S. Zhou, M. L. Gordon, R. Krishna, A. Narcomey, L.  
Fei-Fei, and M. S. Bernstein. "HYPE: a benchmark for human  
eye perceptual evaluation of generative models". In: *Proceedings of the 33rd International Conference on Neural Information  
Processing Systems*. 2019, pp. 3449–3461.
- 3
- 4 [Zho20] G. Zhou. "Mixed Hamiltonian Monte Carlo for Mixed  
Discrete and Continuous Variable". In: (2020). arXiv: 1909.04852  
[stat.CO].
- 5
- 6 [Zho+20] Y. Zhou, H. Yang, Y. W. Teh, and T. Rainforth. "Di-  
vide, Conquer, and Combine: a New Inference Strategy for  
Probabilistic Programs with Stochastic Support". In: *ICML*.  
Ed. by H. D. III and A. Singh. Vol. 119. Proceedings of Ma-  
chine Learning Research. PMLR, 2020, pp. 11534–11545.
- 7
- 8 [ZHT06] H. Zou, T. Hastie, and R. Tibshirani. "Sparse principal  
component analysis". In: *JCGS* 15.2 (2006), pp. 262–286.
- 9
- 10 [Zhu+17] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. "Un-  
paired Image-to-Image Translation using Cycle-Consistent Ad-  
versarial Networks". In: *ICCV*. 2017.
- 11 [zhu+18] X. Zhu, A. Singla, S. Zilles, and A. N. Raf-  
ferty. "An overview of machine teaching". In: *arXiv preprint*  
arXiv:1801.05927 (2018).
- 12
- 13 [zhu+21] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu,  
H. Xiong, and Q. He. "A Comprehensive Survey on Transfer  
Learning". In: *Proc. IEEE* 109.1 (2021).
- 14
- 15 [Zie+08] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K.  
Dey. "Maximum Entropy Inverse Reinforcement Learning". In:  
*AAAI*. 2008, pp. 1433–1438.
- 16 [Zin+20] L. Zintgraf, K. Shiarlis, M. Igl, S. Schulze, Y. Gal, K.  
Hofmann, and S. Whiteson. "VariBAD: A Very Good Method  
for Bayes-Adaptive Deep RL via Meta-Learning". In: *ICLR*.  
2020.
- 17
- 18 [ZLF21] M. Zhang, S. Levine, and C. Finn. "MEMO: Test Time  
Robustness via Adaptation and Augmentation". In: (Oct. 2021).  
arXiv: 2110.09506 [cs.LG].
- 19
- 20 [ZLG20] R. Zivan, O. Lev, and R. Galiki. "Beyond Trees: Analy-  
sis and Convergence of Belief Propagation in Graphs with Mul-  
tiple Cycles". In: *AAAI*. 2020.
- 21 [ZMB21] B. Zhao, K. R. Mopuri, and H. Bilen. "Dataset Conden-  
sation with Gradient Matching". In: *International Conference  
on Learning Representations*. 2021.
- 22
- 23 [ZMG19] G. Zhang, J. Martens, and R. B. Grosse. "Fast Con-  
vergence of Natural Gradient Descent for Over-Parameterized  
Neural Networks". In: *NIPS*. 2019, pp. 8082–8093.
- 24 [ZML16] J. J. Zhao, M. Mathieu, and Y. LeCun. "Energy-based  
Generative Adversarial Network". In: (2016).
- 25
- 26 [ZN20] Y. Zhang and E. Nalisnick. "On the inconsistency of  
Bayesian inference for misspecified neural networks". In: *3rd  
Symposium on Advances in Approximate Bayesian Inference*.  
2020.
- 27
- 28 [Zob09] O. Zobay. "Mean field inference for the Dirichlet pro-  
cess mixture model". In: *Electronic J. of Statistics* 3 (2009),  
pp. 507–545.
- 29
- 30 [Zoe07] O. Zoeter. "Bayesian generalized linear models in a ter-  
abyte world". In: *Proc. 5th International Symposium on Image  
and Signal Processing and Analysis*. 2007.
- 31 [Zon+18] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu,  
D. Cho, and H. Chen. "Deep Autoencoding Gaussian Mixture  
Model for Unsupervised Anomaly Detection". In: *ICLR*. 2018.
- 32
- 33 [ZP00] G. Zweig and M. Padmanabhan. "Exact alpha-beta com-  
putation in logarithmic space with application to map word  
graph construction". In: *ICSLP*. 2000.
- 34
- 35 [ZP96] N. Zhang and D. Poole. "Exploiting causal independence  
in Bayesian network inference". In: *JAIR* (1996), pp. 301–328.
- 36
- 37 [ZR19a] Z. Ziegler and A. Rush. "Latent Normalizing Flows for  
Discrete Sequences". In: *Proceedings of the 36th International  
Conference on Machine Learning*. 2019, pp. 7673–7682.
- 38
- 39 [ZR19b] Z. M. Ziegler and A. M. Rush. "Latent Normalizing  
Flows for Discrete Sequences". In: *ICML*. 2019.
- 40
- 41 [ZS11] Y. Zhang and C. Sutton. "Quasi-newton methods for  
Markov chain Monte Carlo". In: *NIPS*. 2011.
- 42
- 43 [ZSB19] Q. Zhao, D. S. Small, and B. B. Bhattacharya. "Sen-  
sitivity analysis for inverse probability weighting estimators  
via the percentile bootstrap". In: *Journal of the Royal Sta-  
tistical Society: Series B (Statistical Methodology)* 81.4 (2019),  
pp. 735–761. eprint: <https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/rssb.12327>.
- 44
- 45 [ZSE19] S. Zhao, J. Song, and S. Ermon. "InfoVAE: Information  
Maximizing Variational Autoencoders". In: *AAAI*. 2019.
- 46
- 47 [ZVP21] J. A. Zavatone-Veth and C. Pehlevan. "Exact marginal  
prior distributions of finite Bayesian neural networks". In:  
*NIPS*. May 2021.
- 48
- 49 [ZW11] D. Zoran and Y. Weiss. "From learning models of natural  
image patches to whole image restoration". In: *ICCV*. 2011.
- 50
- 51 [ZW12] D. Zoran and Y. Weiss. "Natural Images, Gaussian Mix-  
tures and Dead Leaves". In: *NIPS*. 2012, pp. 1736–1744.
- 52
- 53 [ZY21] Y. Zhang and Q. Yang. "A Survey on Multi-Task Learn-  
ing". In: *IEEE Trans. Knowl. Data Eng.* (2021).
- 54
- 55 [ZY97] Z. Zhang and R. W. Yeung. "A non-Shannon-type con-  
ditional inequality of information quantities". In: *IEEE Trans-  
actions on Information Theory* 43.6 (1997), pp. 1982–1986.
- 56
- 57 [ZY98] Z. Zhang and R. W. Yeung. "On characterization of en-  
tropy function via information inequalities". In: *IEEE Trans-  
actions on Information Theory* 44.4 (1998), pp. 1440–1452.