25 Diffusion models

25.1 Introduction

In this chapter, we consider a class of models called **diffusion models**. This class of models has recently generated a lot of interest, due to its ability to generate diverse, high quality, samples, and the relative simplicity of the training scheme, which allows very large models to be trained at scale. Diffusion models are closely related to VAEs (Chapter 21), normalizing flows (Chapter 23), and EBMs (Chapter 24), as we will see.

The basic idea behind these models is based on the observation that it is 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 $\boldsymbol{x}_0$ into a noisy version $\boldsymbol{x}_T$ by passing the data through T steps of a stochastic encoder $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$. After enough steps, we have $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, or some other convenient reference distribution. We then learn a **reverse process** to undo this, by passing the noise through T steps of a decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ until we generate $\boldsymbol{x}_0$. See Figure 25.1 for an overall sketch of the approach. In the following sections, we discuss this class of models in more detail. Our presentation is based in part on the excellent tutorial [KGV22]. More details can be found in the recent review papers [Yan+22; Cao+22], as well as specialized papers, such as [Kar+22]. There are also many other excellent resources online, such as `https://github.com/heejkoo/Awesome-Diffusion-Models` and `https://scorebasedgenerativemodeling.github.io/`. For a detailed tutorial on the underlying math, see [McA23].

25.2 Denoising diffusion probabilistic models (DDPMs)

In this section, we discuss **denoising diffusion probabilistic models** or **DDPMs**, introduced in [SD+15b], and then extended in [HJA20; Kin+21] and many other works. We can think of the DDPM as similar to a hierarchical variational autoencoder (Section 21.5), except that all the latent states (denoted $\boldsymbol{x}_t$ for $t = 1 : T$) have the same dimensionality as the input $\boldsymbol{x}_0$. (In this respect, a DDPM is similar to a normalizing flow (Chapter 23); however, in a diffusion model, the hidden layers are stochastic, and do not need to use invertible transformations.) In addition, the encoder network q is a simple linear Gaussian model, rather than being learned[1], and the decoder network p is shared across all time steps. These restrictions result in a very simple training objective, which

1. Later we will discuss some extensions in which the noise level of the encoder can also be learned. Nevertheless, the encoder remains simple, by design.

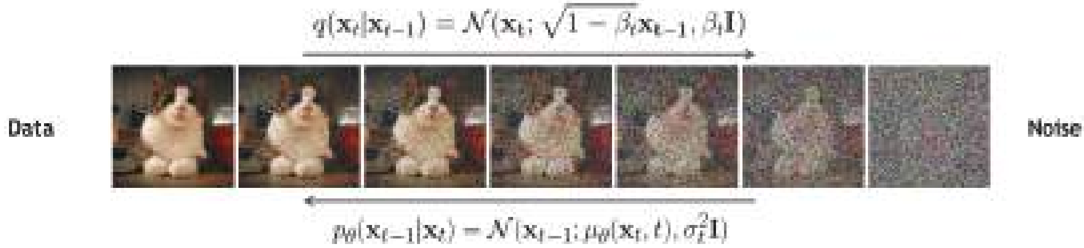


Figure 25.1: Denoising diffusion probabilistic model. The forwards diffusion process, $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$, implements the (non-learned) inference network; this just adds Gaussian noise at each step. The reverse diffusion process, $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, implements the decoder; this is a learned Gaussian model. From Slide 16 of [KGV22]. Used with kind permission of Arash Vahdat.

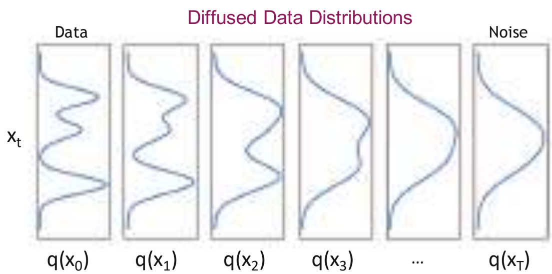


Figure 25.2: Illustration of a diffusion model on 1d data. The forwards diffusion process gradually transforms the empirical data distribution $q(\boldsymbol{x}_0)$ into a simple target distribution, here $q(\boldsymbol{x}_T) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. To generate from the model, we sample a point $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and then run the Markov chain backwards, by sampling $\boldsymbol{x}_t \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$ until we get a sample in the original data space, $\boldsymbol{x}_0$. From Slide 19 of [KGV22]. Used with kind permission of Arash Vahdat.

allows deep models to be easily trained without any risk of posterior collapse (Section 21.4). In particular, in Section 25.2.3, we will see that training reduces to a series of weighted nonlinear least squares problems.

25.2.1 Encoder (forwards diffusion)

The forwards encoder process is defined to be a simple linear Gaussian model:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t|\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\mathbf{I}) \tag{25.1}$$

where the values of $\beta_t \in (0, 1)$ are chosen according to a noise schedule (see Section 25.2.4). The joint distribution over all the latent states, conditioned on the input, is given by

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \tag{25.2}$$

Since this defines a linear Gaussian Markov chain, we can compute marginals of it in closed form. In particular, we have

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t|\sqrt{\overline{\alpha}_t}\boldsymbol{x}_0, (1-\overline{\alpha}_t)\mathbf{I}) \tag{25.3}$$

where we define

$$\alpha_t \triangleq 1-\beta_t, \overline{\alpha}_t = \prod_{s=1}^{t} \alpha_s \tag{25.4}$$

We choose the noise schedule such that $\overline{\alpha}_T \approx 0$, so that $q(\boldsymbol{x}_T|\boldsymbol{x}_0) \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$.

The distribution $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ is known as the **diffusion kernel**. Applying this to the input data distribution and then computing the result unconditional marginals is equivalent to Gaussian convolution:

$$q(\boldsymbol{x}_t) = \int q_0(\boldsymbol{x}_0) q(\boldsymbol{x}_t|\boldsymbol{x}_0) d\boldsymbol{x}_0 \tag{25.5}$$

As t increases, the marginals become simpler, as shown in Figure 25.2. In the image domain, this process will first remove high-frequency content (i.e., low-level details, such as texture), and later will remove low-frequency content (i.e., high-level or “semantic” information, such as shape), as shown in Figure 25.1.

25.2.2 Decoder (reverse diffusion)

In the reverse diffusion process, we would like to invert the forwards diffusion process. If we know the input $\boldsymbol{x}_0$, we can derive the reverse of one forwards step as follows:[2]

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_{t-1}|\tilde{\mu}_t(\boldsymbol{x}_t, \boldsymbol{x}_0), \tilde{\beta}_t\mathbf{I}) \tag{25.6}$$

$$\tilde{\mu}_t(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t}\boldsymbol{x}_0 + \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\boldsymbol{x}_t \tag{25.7}$$

$$\tilde{\beta}_t = \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t \tag{25.8}$$

Of course, when generating a new datapoint, we do not know $\boldsymbol{x}_0$, but we will train the generator to approximate the above distribution averaged over $\boldsymbol{x}_0$. Thus we choose the generator to have the form

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}|\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)) \tag{25.9}$$

We often set $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \sigma_t^2\mathbf{I}$. We discuss how to learn σ_t^2 in Section 25.2.4, but two natural choices are $\sigma_t^2 = \beta_t$ and $\sigma_t^2 = \tilde{\beta}_t$; these correspond to upper and lower bounds on the reverse process entropy, as shown in [HJA20].

The corresponding joint distribution over all the generated variables is given by $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_T)\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, where we set $p(\boldsymbol{x}_T) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. We can sample from this distribution using the pseudocode in Algorithm 25.2.

2. We just need to use Bayes' rule for Gaussians. See e.g., https://lilianweng.github.io/posts/2021-07-11-diffusion-models/ for a detailed derivation.

25.2.3 Model fitting

We will fit the model by maximizing the evidence lower bound (ELBO), similar to how we train VAEs (see Section 21.2). In particular, for each data example $\boldsymbol{x}_0$ we have

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0) = \log \left[\int d\boldsymbol{x}_{1:T} q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \right] \tag{25.10}$$

$$\geq \int d\boldsymbol{x}_{1:T} q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) \log \left(p(\boldsymbol{x}_T) \prod_{t=1}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})} \right) \tag{25.11}$$

$$= \mathbb{E}_q \left[\log p(\boldsymbol{x}_T) + \sum_{t=1}^{T} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})} \right] \triangleq \text{Ł}(\boldsymbol{x}_0) \tag{25.12}$$

We now discuss how to compute the terms in the ELBO. By the Markov property we have $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)$, and by Bayes' rule, we have

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) q(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)} \tag{25.13}$$

Plugging Equation (25.13) into the ELBO we get

$$\text{Ł}(\boldsymbol{x}_0) = \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log p(\boldsymbol{x}_T) + \sum_{t=2}^{T} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)} + \underbrace{\sum_{t=2}^{T} \log \frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}}_{*} + \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} \right] \tag{25.14}$$

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

$$* = \log q(\boldsymbol{x}_{T-1}|\boldsymbol{x}_0) + \cdots + \log q(\boldsymbol{x}_2|\boldsymbol{x}_0) + \log q(\boldsymbol{x}_1|\boldsymbol{x}_0) \tag{25.15}$$

$$- \log q(\boldsymbol{x}_T|\boldsymbol{x}_0) - \log q(\boldsymbol{x}_{T-1}|\boldsymbol{x}_0) - \cdots - \log q(\boldsymbol{x}_2|\boldsymbol{x}_0) \tag{25.16}$$

$$= - \log q(\boldsymbol{x}_T|\boldsymbol{x}_0) + \log q(\boldsymbol{x}_1|\boldsymbol{x}_0) \tag{25.17}$$

Hence the negative ELBO (variational upper bound) becomes

$$\mathcal{L}(\boldsymbol{x}_0) = -\mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log \frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)} + \sum_{t=2}^{T} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)} + \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1) \right] \tag{25.18}$$

$$= \underbrace{D_{\mathbb{KL}} \left(q(\boldsymbol{x}_T|\boldsymbol{x}_0) \parallel p(\boldsymbol{x}_T) \right)}_{L_T(\boldsymbol{x}_0)} \tag{25.19}$$

$$+ \sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \underbrace{D_{\mathbb{KL}} \left(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) \right)}_{L_{t-1}(\boldsymbol{x}_0)} - \underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)}_{L_0(\boldsymbol{x}_0)} \tag{25.20}$$

Each of these KL terms can be computed analytically, since all the distributions are Gaussian. Below we focus on the L_{t-1} term. Since $\boldsymbol{x}_t = \sqrt{\overline{\alpha}_t}\boldsymbol{x}_0 + \sqrt{(1-\overline{\alpha}_t)}\boldsymbol{\epsilon}_t$, we can write

$$\tilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}\right) \tag{25.21}$$

Thus instead of training the model to predict the mean of the denoised version of $\boldsymbol{x}_{t-1}$ given its noisy input $\boldsymbol{x}_t$, we can train the model to predict the noise, from which we can compute the mean:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\right) \tag{25.22}$$

where the dependence on $\boldsymbol{x}_0$ is implicit. With this parameterization, the loss (averaged over the dataset) becomes

$$L_{t-1} = \mathbb{E}_{\boldsymbol{x}_0 \sim q_0(\boldsymbol{x}_0), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}\left[\underbrace{\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\overline{\alpha}_t)}}_{\lambda_t} ||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\underbrace{\sqrt{\overline{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}}_{\boldsymbol{x}_t}, t\right)||^2\right] \tag{25.23}$$

The time dependent weight λ_t ensures that the training objective corresponds to maximum likelihood training (assuming the variational bound is tight). However, it has been found empirically that the model produces better looking samples if we set $\lambda_t = 1$. The resulting simplified loss (also averaging over time steps t in the model) is given by

$$L_{\text{simple}} = \mathbb{E}_{\boldsymbol{x}_0 \sim q_0(\boldsymbol{x}_0), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \text{Unif}(1,T)}\left[||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\underbrace{\sqrt{\overline{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}}_{\boldsymbol{x}_t}, t\right)||^2\right] \tag{25.24}$$

The overall training procedure is shown in Algorithm 25.1. We can improve the perceptual quality of samples using more advanced weighting schemes, are discussed in [Cho+22]. Conversely, if the goal is to improve likelihood scores, we can optimize the noise schedule, as discussed in Section 25.2.4.

Algorithm 25.1: Training a DDPM model with L_{simple}.

while *not converged* **do**
- $\boldsymbol{x}_0 \sim q_0(\boldsymbol{x}_0)$
- $t \sim \text{Unif}(\{1, \ldots, T\})$
- $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Take gradient descent step on $\nabla_{\boldsymbol{\theta}} ||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\sqrt{\overline{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}, t\right)||^2$

After the model is trained, we can generate data using ancestral sampling, as shown in Algorithm 25.2.

Algorithm 25.2: Sampling from a DDPM model.

$\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
foreach $t = T, \ldots, 1$ **do**
  $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \right) + \sigma_t \boldsymbol{\epsilon}_t$
Return $\boldsymbol{x}_0$

25.2.4 Learning the noise schedule

In this section, we describe a way to optimize the noise schedule used by the encoder so as to maximize the ELBO; this approach is called a **variational diffusion model** or **VDM** [Kin+21].

We will use the following parameterization of the encoder:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t|\hat{\alpha}_t \boldsymbol{x}_0, \hat{\sigma}_t^2 \mathbf{I}) \tag{25.25}$$

(Note that $\hat{\alpha}_t$ and $\hat{\sigma}_t$ are different to the parameters α_t and σ_t in Section 25.2.1.) Rather than working with $\hat{\alpha}_t$ and $\hat{\sigma}_t^2$ separately, we will learn to predict their ratio, which is known as the **signal to noise ratio**:

$$R(t) = \hat{\alpha}_t^2 / \hat{\sigma}_t^2 \tag{25.26}$$

This should be monotonically decreasing in t. This can be ensured by defining $R(t) = \exp(-\gamma_{\boldsymbol{\phi}}(t))$, where $\gamma_{\boldsymbol{\phi}}(t)$ is a monotonic neural network. We usually set $\hat{\alpha}_t = \sqrt{1 - \sigma_t^2}$, to corresponde to the variance preserving SDE discussed in Section 25.4.

Following the derivation in Section 25.2.3, the negative ELBO (variational upper bound) can be written as

$$\mathcal{L}(\boldsymbol{x}_0) = \underbrace{D_{\mathbb{KL}}\left(q(\boldsymbol{x}_T|\boldsymbol{x}_0) \parallel p(\boldsymbol{x}_T)\right)}_{\text{prior loss}} + \underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}[-\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)]}_{\text{reconstruction loss}} + \underbrace{\mathcal{L}_D(\boldsymbol{x}_0)}_{\text{diffusion loss}} \tag{25.27}$$

where the first two terms are similar to a standard VAE, and the final diffusion loss is given below:[3]

$$\mathcal{L}_D(\boldsymbol{x}_0) = \frac{1}{2} \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \int_0^1 R'(t) \left\| \boldsymbol{x}_0 - \hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{z}_t, t) \right\|_2^2 dt \tag{25.28}$$

where $R'(t)$ is the derivative of the SNR function, and $\boldsymbol{z}_t = \alpha_t \boldsymbol{x}_0 + \sigma_t \boldsymbol{\epsilon}_t$. (See [Kin+21] for the derivation.)

Since the SNR functon is invertible, due to the monotonicity assumtion, we can perform a change of variables, and make everything a function of $v = R(t)$ instead of t. In particular, let $\boldsymbol{z}_v = \alpha_v \boldsymbol{x}_0 + \sigma_v \boldsymbol{\epsilon}$, and $\tilde{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{z}, v) = \hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{z}, R^{-1}(v))$. Then we can rewrite Equation (25.28) as

$$\mathcal{L}_D(\boldsymbol{x}_0) = \frac{1}{2} \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \int_{R_{\min}}^{R_{\max}} \left\| \boldsymbol{x}_0 - \tilde{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{z}_v, v) \right\|_2^2 dv \tag{25.29}$$

3. We present a simplified form of the loss that uses the continuous time limit, which we discuss in Section 25.4.

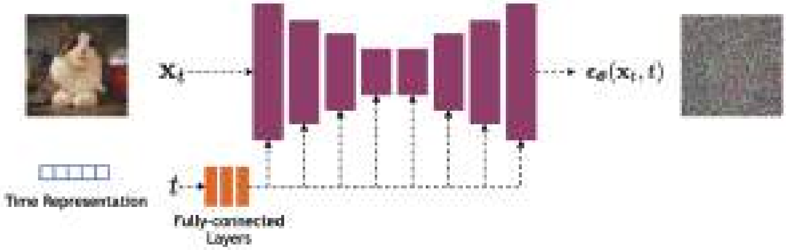


Figure 25.3: Illustration of the U-net architecture used in the denoising step. From Slide 26 of [KGV22]. Used with kind permission of Arash Vahdat.

(a) (b) (c)

Figure 25.4: Some sample images generated by a small variational diffusion model trained on EMNIST for about 30 minutes on a K40 GPU. (a) Unconditional sampling. (b) Conditioned on class label. (c) Using classifier-free guidance (see Section 25.6.3). Generated by diffusion_emnist.ipynb. Used with kind permission of Alex Alemi.

where $R_{\min} = R(1)$ and $R_{\max} = R(0)$. Thus we see that the shape of the SNR schedule does not matter, except for its value at the two end points.

The integral in Equation (25.29) can be estimated by sampling a timestep uniformly at random. When processing a minibatch of k examples, we can produce a lower variance estimate of the variational bound by using a **low-discrepancy sampler** (cf., Section 11.6.5). In this approach, instead of sampling the timesteps independently, we sample a single uniform random number $u_0 \sim \text{Unif}(0, 1)$, and then set $t^i = \mod (u_0 + i/k, 1)$ for the i'th sample. We can also optimize the noise schedule wrt the variance of the diffusion loss.

25.2.5 Example: image generation

Diffusion models are often used to generate images. The most common architecture for image generation is based on the **U-net** model [RFB15], as shown in Figure 25.3. The time step t is encoded as a vector, using sinusoidal positional encoding or random Fourier features, and is then fed into the residual blocks, using either simple spatial addition or by conditioning the group norm layers [DN21a]. Of course, other architectures besides U-net are possible. For example, recently [PX22; Li+22; Bao+22a] have proposed the use of transformers, to replace the convolutional and deconvolutional layers.

The results of training a small U-net VDM on EMNIST images are shown in Figure 25.4. By training big models (billions of parameters) for a long time (days) on lots of data (millions of images), diffusion models can be made to generate very high quality images (see Figure 20.2). Results can be further improved by using conditional diffusion models, where guidance is provided about what kinds of images to generate (see Section 25.6).

25.3 Score-based generative models (SGMs)

This section is written with Yang Song and Durk Kingma.

In Section 24.3, we discussed how to fit energy based models (EBMs) using score matching. This adjusts the parameters of the EBM so that the score function of the model, $\nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x})$, matches the score function of the data, $\nabla_{\boldsymbol{x}} \log p_{\mathcal{D}}(\boldsymbol{x})$. An alternative to fitting a scalar energy function and computing its score is to directly learn the score function. This is called a **score-based generative model** or **SGM** [SE19; SE20b; Son+21b]. We can optimize the score function $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x})$ using basic score matching (Section 24.3.1), sliced score matching (Section 24.3.3 or denoising score matching (Section 24.3.2). We discuss this class of models in more detail below. (For a comparison with EBMs, see [SH21].)

25.3.1 Example

In Figure 25.5a, we show the **Swiss roll** dataset. We estimate the score function by fitting an MLP with 2 hidden layers, each with 128 hidden units, using basic score matching. In Figure 25.5b, we show the output of the network after training for 10,000 steps of SGD. We see that there are no major false negatives (since wherever the density of the data is highest, the gradient field is zero), but there are some false positives (since some regions of zero gradient do not correspond to data regions). The comparison of the predicted outputs with the empirical data density is shown more clearly in Figure 25.5c. In Figure 25.5d, we show some samples from the learned model, generated using Langevin sampling.

25.3.2 Adding noise at multiple scales

In general, score matching can have difficulty when there are regions of low data density. To see this, suppose $p_{\mathcal{D}}(\mathbf{x}) = \pi p_0(\mathbf{x}) + (1 - \pi) p_1(\mathbf{x})$. 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 the supports of $p_0(\mathbf{x})$ and $p_1(\mathbf{x})$ respectively. When they are disjoint from each other, the score of $p_{\mathcal{D}}(\mathbf{x})$ is given by

$$\nabla_{\mathbf{x}} \log p_{\mathcal{D}}(\mathbf{x}) = \begin{cases} \nabla_{\mathbf{x}} \log p_0(\mathbf{x}), & \mathbf{x} \in \mathcal{S}_0 \\ \nabla_{\mathbf{x}} \log p_1(\mathbf{x}), & \mathbf{x} \in \mathcal{S}_1, \end{cases} \tag{25.30}$$

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

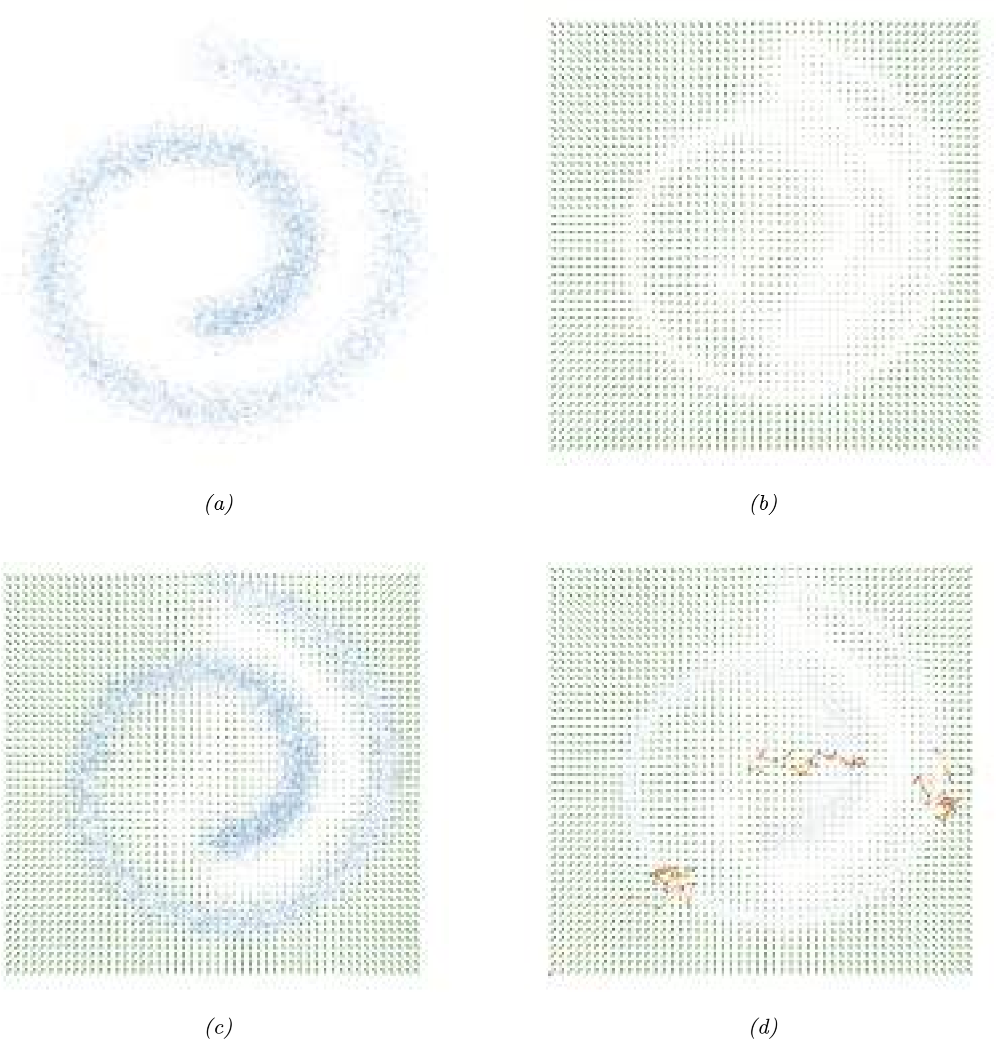


Figure 25.5: Fitting a score-based generative model to the 2d Swiss roll dataset. (a) Training set. (b) Learned score function trained using the basic score matching. (c) Superposition of learned score function and empirical density. (d) Langevin sampling applied to the learned model. We show 3 different trajectories, each of length 25. Generated by score_matching_swiss_roll.ipynb.

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

$$q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) = \mathcal{N}(\tilde{\boldsymbol{x}}|\boldsymbol{x}, \sigma^2\mathbf{I}) \tag{25.31}$$

$$q_\sigma(\tilde{\boldsymbol{x}}) = \int p_{\mathcal{D}}(\boldsymbol{x}) q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) d\boldsymbol{x} \tag{25.32}$$

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

In practice, all score models share weights and are implemented with a single neural network conditioned on the noise scale; this is called a **noise conditional score network**, and has the form $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, \sigma)$. Scores of different scales are estimated by training a mixture of score matching objectives, one per noise scale. If we use the denoising score matching objective in Equation (24.33), we get

$$\mathcal{L}(\boldsymbol{\theta}; \sigma) = \mathbb{E}_{q(\mathbf{x}, \tilde{\mathbf{x}})} \left[\frac{1}{2} \|\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \sigma) - \nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})\|_2^2 \right] \tag{25.33}$$

$$= \frac{1}{2} \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})} \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathcal{N}(\boldsymbol{x}, \sigma^2\mathbf{I})} \left\{ \left\| \boldsymbol{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \sigma) + \frac{(\tilde{\mathbf{x}} - \boldsymbol{x})}{\sigma^2} \right\|_2^2 \right\} \tag{25.34}$$

where we used the fact that, for a Gaussian, the score is given by

$$\nabla_{\boldsymbol{x}} \log \mathcal{N}(\tilde{\boldsymbol{x}}|\boldsymbol{x}, \sigma^2\mathbf{I}) = -\nabla_{\boldsymbol{x}} \frac{1}{2\sigma^2} (\boldsymbol{x} - \tilde{\boldsymbol{x}})^\mathsf{T} (\boldsymbol{x} - \tilde{\boldsymbol{x}}) = \frac{\boldsymbol{x} - \tilde{\boldsymbol{x}}}{\sigma^2} \tag{25.35}$$

If we have T different noise scales, we can combine the losses in a weighted fashion using

$$\mathcal{L}(\boldsymbol{\theta}; \sigma_{1:T}) = \sum_{t=1}^{T} \lambda_t \mathcal{L}(\boldsymbol{\theta}; \sigma_t) \tag{25.36}$$

where we choose $\sigma_1 > \sigma_2 > \cdots > \sigma_T$, and the weighting term satisfies $\lambda_t > 0$.

25.3.3 Equivalence to DDPM

We now show that the above score-based generative model training objective is equivalent to the DDPM loss. To see this, first let us replace $p_{\mathcal{D}}(\boldsymbol{x})$ with $q_0(\boldsymbol{x}_0)$, $\tilde{\boldsymbol{x}}$ with $\boldsymbol{x}_t$, and $\boldsymbol{s}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}, \sigma)$ with $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$. We will also compute a stochastic approximation to Equation (25.36) by sampling a time step uniformly at random. Then Equation (25.36) becomes

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{x}_0 \sim q_0(\boldsymbol{x}_0), \boldsymbol{x}_t \sim q(\boldsymbol{x}_t|\boldsymbol{x}_0), t \sim \text{Unif}(1,T)} \left[\lambda_t \left\| \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) + \frac{(\boldsymbol{x}_t - \boldsymbol{x}_0)}{\sigma_t^2} \right\|_2^2 \right] \tag{25.37}$$

If we use the fact that $\boldsymbol{x}_t = \boldsymbol{x}_0 + \sigma_t \boldsymbol{\epsilon}$, and if we define $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = -\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)}{\sigma_t}$, we can rewrite this as

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{x}_0 \sim q_0(\boldsymbol{x}_0), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \text{Unif}(1,T)} \left[\frac{\lambda_t}{\sigma_t^2} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \right\|_2^2 \right] \tag{25.38}$$

If we set $\lambda_t = \sigma_t^2$, we recover L_{simple} loss in Equation (25.24).

25.4 Continuous time models using differential equations

In this section, we consider a DDPM model in the limit of an infinite number of hidden layers, or equivalently, an SGM in the limit of an infinite number of noise levels. This requires switching from discrete time to continuous time, which complicates the mathematics. The advantage is that we can leverage the large existing literature on solvers for ordinary and stochastic differential equations to enable faster generation, as we will see.

25.4.1 Forwards diffusion SDE

Let us first consider a diffusion process where the noise level β_t gets rewritten as $\beta(t)\Delta t$, where Δt is a step size:

$$\boldsymbol{x}_t = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\mathcal{N}(\mathbf{0}, \mathbf{I}) = \sqrt{1-\beta(t)\Delta t}\boldsymbol{x}_{t-1} + \sqrt{\beta(t)\Delta t}\mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{25.39}$$

If Δt is small, we can approximate the first term with a first-order Taylor series expansion to get

$$\boldsymbol{x}_t \approx \boldsymbol{x}_{t-1} - \frac{\beta(t)\Delta t}{2}\boldsymbol{x}_{t-1} + \sqrt{\beta(t)\Delta t}\mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{25.40}$$

Hence for small Δt we have

$$\frac{\boldsymbol{x}_t - \boldsymbol{x}_{t-1}}{\Delta t} \approx -\frac{\beta(t)}{2}\boldsymbol{x}_{t-1} + \frac{\sqrt{\beta(t)}}{\sqrt{\Delta t}}\mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{25.41}$$

We can now switch to the **continuous time** limit, and write this as the following **stochastic differential equation** or **SDE**:

$$\frac{d\boldsymbol{x}(t)}{dt} = -\frac{1}{2}\beta(t)\boldsymbol{x}(t) + \sqrt{\beta(t)}\frac{d\boldsymbol{w}(t)}{dt} \tag{25.42}$$

where $\boldsymbol{w}(t)$ represents a standard **Wiener process**, also called **Brownian noise**. More generally, we can write such SDEs as follows, where we use **Itô calculus** notation (see e.g., [SS19]):

$$d\boldsymbol{x} = \underbrace{\boldsymbol{f}(\boldsymbol{x}, t)}_{\text{drift}} dt + \underbrace{g(t)}_{\text{diffusion}} d\boldsymbol{w} \tag{25.43}$$

The first term in the above SDE is called the **drift coefficient**, and the second term is called the **diffusion coefficient**.

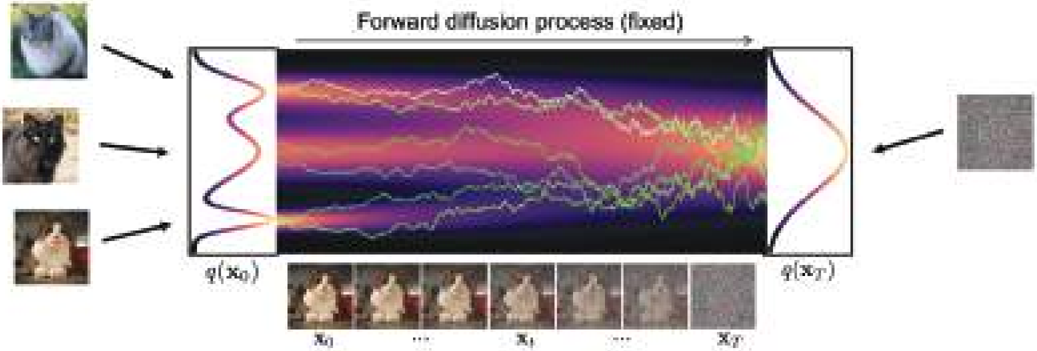


Figure 25.6: Illustration of the forwards diffusion process in continuous time. Yellow lines are sample paths from the SDE. Heat map represents the marginal distribution computed using the probability flow ODE. From Slide 43 of [KGV22]. Used with kind permission of Karsten Kreis.

We can gain some intuition for these processes by looking at the 1d example in Figure 25.6. We can draw multiple paths as follows: sample an initial state from the data distribution, and then integrate over time using **Euler-Maruyama** integration:

$$\boldsymbol{x}(t+\Delta t) = \boldsymbol{x}(t) + \boldsymbol{f}(\boldsymbol{x}(t), t)\Delta t + g(t)\sqrt{\Delta t}\mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{25.44}$$

We can see how the data distributiom at $t = 0$, on the left hand side, gradually gets transformed to a pure noise distribution at $t = 1$, on the right hand side.

In [Son+21b], they show that the SDE corresponding to DDPMs, in the $T \to \infty$ limit, is given by

$$d\boldsymbol{x} = -\frac{1}{2}\beta(t)\boldsymbol{x}dt + \sqrt{\beta(t)}d\boldsymbol{\omega} \tag{25.45}$$

where $\beta(t/T) = T\beta_t$. Here the drift term is proportional to $-\boldsymbol{x}$, which encourages the process to return to 0. Consequently, DDPM corresponds to a **variance preserving** process. By contrast, the SDE corresponding to SGMs is given by the following:

$$d\boldsymbol{x} = \sqrt{\frac{d[\sigma(t)^2]}{dt}}d\boldsymbol{\omega} \tag{25.46}$$

where $\sigma(t/T) = \sigma_t$. This SDE has zero drift, so corresponds to a **variance exploding** process.

25.4.2 Forwards diffusion ODE

Instead of adding Gaussian noise at every step, we can just sample the initial state, and then let it evolve deterministically over time according to the following **ordinary differential equation** or **ODE**:

$$d\boldsymbol{x} = \underbrace{\left[f(\boldsymbol{x}, t) - \frac{1}{2}g(t)^2\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x})\right]}_{h(\boldsymbol{x},t)} dt \tag{25.47}$$

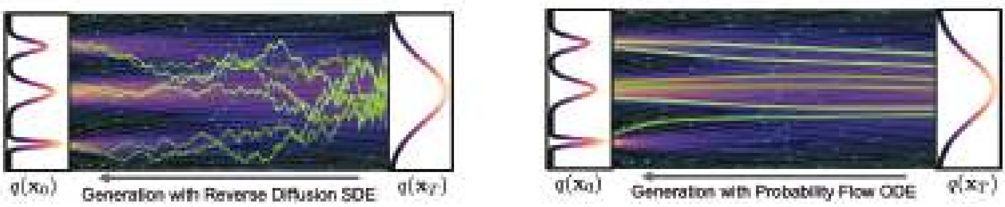


Figure 25.7: Illustration of the reverse diffusion process. (a) Sample paths from the SDE. (b) Deterministic trajectories from the probability flow ODE. From Slide 65 of [KGV22]. Used with kind permission of Karsten Kreis.

This is called the **probability flow ODE** [Son+21b, Sec D.3]. We can compute the state at any moment in time using any ODE solver:

$$\boldsymbol{x}(t) = \boldsymbol{x}(0) + \int_0^t h(\boldsymbol{x}, \tau) d\tau \tag{25.48}$$

See Figure 25.7b for a visualization of a sample trajectory. If we start the solver from different random states $\boldsymbol{x}(0)$, then the induced distribution over paths will have the same marginals as the SDE model. See the heatmap in Figure 25.6 for an illustration.

25.4.3 Reverse diffusion SDE

To generate samples from this model, we need to be able to reverse the SDE. In a remarkable result, [And82] showed that any forwards SDE of the form in Equation (25.43) can be reversed to get the following **reverse-time SDE**:

$$d\boldsymbol{x} = \left[f(\boldsymbol{x}, t) - g(t)^2 \nabla_{\boldsymbol{x}} \log q_t(\boldsymbol{x}) \right] dt + g(t) d\overline{\boldsymbol{w}} \tag{25.49}$$

where $\overline{\boldsymbol{w}}$ is the standard Wiener process when time flows backwards, dt is an infinitesimal negative time step, and $\nabla_{\boldsymbol{x}} \log q_t(\boldsymbol{x})$ is the score function.

In the case of the DDPM, the reverse SDE has the following form:

$$d\boldsymbol{x}_t = \left[-\frac{1}{2}\beta(t)\boldsymbol{x}_t - \beta(t)\nabla_{\boldsymbol{x}_t} \log q_t(\boldsymbol{x}_t) \right] dt + \sqrt{\beta(t)} d\overline{\boldsymbol{w}}_t \tag{25.50}$$

To estimate the score function, we can use denoising score matching as we discussed in Section 25.3, to get

$$\nabla_{\boldsymbol{x}_t} \log q_t(\boldsymbol{x}_t) \approx \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \tag{25.51}$$

(In practice, it is advisable to use variance reduction techniques, such as importance sampling, as discussed in [Son+21a].) The SDE becomes

$$d\boldsymbol{x}_t = -\frac{1}{2}\beta(t) \left[\boldsymbol{x}_t + 2\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \right] dt + \sqrt{\beta(t)} d\overline{\boldsymbol{w}}_t \tag{25.52}$$

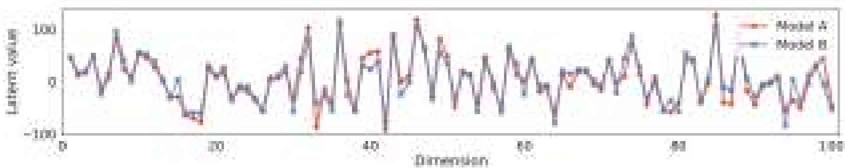


Figure 25.8: Comparing the first 100 dimensions of the latent code obtained for a random CIFAR-100 image. "Model A" and "Model B" are separately trained with different architectures. From Figure 7 of [Son+21b]. Used with kind permission of Yang Song.

After fitting the score network, we can sample from it using ancestral sampling (as in Section 25.2), or we can use the Euler-Maruyama integration scheme in Equation (25.44), which gives

$$\boldsymbol{x}_{t-1} = \boldsymbol{x}_t + \frac{1}{2}\beta(t)\left[\boldsymbol{x}_t + 2\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\right]\Delta t + \sqrt{\beta(t)\Delta t}\mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{25.53}$$

See Figure 25.7a for an illustration.

25.4.4 Reverse diffusion ODE

Based on the results in Section 25.4.2, we can derive the probability flow ODE from the reverse-time SDE in Equation (25.49) to get

$$d\boldsymbol{x}_t = \left[f(\boldsymbol{x}, t) - \frac{1}{2}g(t)^2 \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\right] dt \tag{25.54}$$

If we set $f(\boldsymbol{x}, t) = -\frac{1}{2}\beta(t)$ and $g(t) = \sqrt{\beta(t)}$, as in DDPM, this becomes

$$d\boldsymbol{x}_t = -\frac{1}{2}\beta(t)\left[\boldsymbol{x}_t + \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\right] dt \tag{25.55}$$

See Figure 25.7b for an illustration. A simple way to solve this ODE is to use **Euler's method**:

$$\boldsymbol{x}_{t-1} = \boldsymbol{x}_t + \frac{1}{2}\beta(t)\left[\boldsymbol{x}_t + \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\right]\Delta t \tag{25.56}$$

However, in practice one can get better results using higher-order ODE solvers, such as **Heun's method** [Kar+22].

This model is a special case of a **neural ODE**, also called a **continuous normalizing flow** (see Section 23.2.6). Consequently we can derive the exact log marginal likelihood. However, instead of maximizing this directly (which is expensive), we use score matching to fit the model.

Another advantage of the deterministic ODE approach is that it guarantees that the generative model is **identifiable**. To see this, note that the ODE (in both forwards and reverse directions) is deterministic, and is uniquely determined by the score function. If the architecture is sufficiently flexible, and if there is enough data, then score matching will recover the true score function of the data generating process. Thus, after training, a given datapoint will map to a unique point in latent space, regardless of the model architecture or initialization (see Figure 25.8).

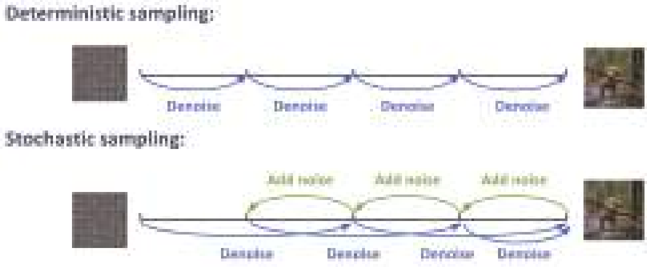


Figure 25.9: Generatng from the reverse diffusion process using 4 steps. (Top) Deterministic sampling. (Bottom) A mix of deterministic and stochastic sampling. Used with kind permission of Ruiqi Gao.

Furthermore, since every point in latent space decodes to a unique image, we can perform "semantic interpolation" in the latent space to generate images with properties that are in between two input examples (cf., Figure 20.9).

25.4.5 Comparison of the SDE and ODE approach

In Section 25.4.3 we described the reverse diffusion process as an SDE, and in Section 25.4.4, we described it as an ODE. We can see the connection between these methods by rewriting the SDE in Equation (25.49) as follows:

$$d\boldsymbol{x}_t = \underbrace{-\frac{1}{2}\beta(t)[\boldsymbol{x}_t + \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)]dt}_{\text{probability flow ODE}} \underbrace{-\frac{1}{2}\beta(t)\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)dt + \sqrt{\beta(t)}d\overline{\boldsymbol{w}}_t}_{\text{Langevin diffusion SDE}} \tag{25.57}$$

The continuous noise injection can compensate for errors introduced by the numerical integration of the ODE term. Consequently, the resulting samples often look better. However, the ODE approach can be faster. Fortunately it is possible to combine these techniques, as proposed in [Kar+22]. The basic idea is illustrated in Figure 25.9: we alternate between performing a deterministic step using an ODE solver, and then adding a small amount noise to the result. This can be repeated for some number of steps. (We discuss ways to reduce the number of required steps in Section 25.5.)

25.4.6 Example

A simple JAX implementation of the above ideas, written by Winnie Xu, can be found in diffusion_mnist.ipynb. This fits a small model to MNIST images using denoising score matching. It then generates from the model by solving the probability flow ODE using the diffrax library. By scaling this kind of method up to a much larger model, and training for a much longer time, it is possible to produce very impressive looking results, as shown in Figure 25.10.

25.5 Speeding up diffusion models

One of the main disadvantages of diffusion models is that generating from them takes many small steps, which can be slow. While it is possible to just take fewer, larger steps, the results are much

Figure 25.10: Synthetic faces from a score-based generative model trained on CelebA-HQ-256 images. From Figure 12 of [Son+21b]. Used with kind permission of Yang Song.

worse. In this section, we briefly mention a few techniques that have been proposed to tackle this important problem. Many other techniques are mentioned in the recent review papers [UAP22; Yan+22; Cao+22].

25.5.1 DDIM sampler

In this section, we describe the denoising diffusion implicit model or **DDIM** of [SME21], which can be used for efficient deterministic generation. The first step is to use a **non-Markovian** forwards diffusion process, so it always conditions on the input in addition to the previous step:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}(\sqrt{\overline{\alpha}_{t-1}}\boldsymbol{x}_0 + \sqrt{1 - \overline{\alpha}_{t-1} - \tilde{\sigma}_t^2}\frac{\boldsymbol{x}_t - \sqrt{\overline{\alpha}_t}\boldsymbol{x}_0}{\sqrt{1-\overline{\alpha}_t}}, \tilde{\sigma}_t^2\mathbf{I}) \tag{25.58}$$

The corresponding reverse process is

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\sqrt{\overline{\alpha}_{t-1}}\hat{\boldsymbol{x}}_0 + \sqrt{1 - \overline{\alpha}_{t-1} - \tilde{\sigma}_t^2}\frac{\boldsymbol{x}_t - \sqrt{\overline{\alpha}_t}\hat{\boldsymbol{x}}_0}{\sqrt{1-\overline{\alpha}_t}}, \tilde{\sigma}_t^2\mathbf{I}) \tag{25.59}$$

where $\hat{\boldsymbol{x}}_0 = \hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ is the predicted output from the model. By setting $\tilde{\sigma}_t^2 = 0$, the reverse process becomes fully deterministic, given the initial prior sample (whose variance is controlled by $\tilde{\sigma}_T^2$). The resulting probability flow ODE gives better results when using a small number of steps compared to the methods discussed in Section 25.4.4.

Note that the weighted negative VLB for this model is the same as L_{simple} in Section 25.2, so the DDIM sampler can be applied to a trained DDPM model.

25.5.2 Non-Gaussian decoder networks

If the reverse diffusion process takes larger steps, then the induced distribution over clean outputs given a noisy input will become multimodal, as illustrated in Figure 25.11. This requires more complicated forms for the distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$. In [Gao+21], they use an EBM to fit this distribution. However, this still requires the use of MCMC to draw a sample. In [XKV22], they use a GAN (Chapter 26) to fit this distribution. This enables us to easily draw a sample by passing

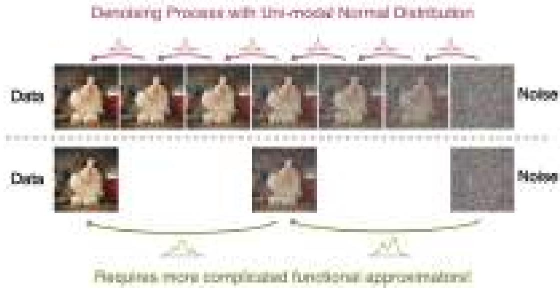


Figure 25.11: Illustration of why taking larger steps in the reverse diffusion process needs more complex, mulit-modal conditional distributions. From Slide 90 of [KGV22]. Used with kind permission of Arash Vahdat.

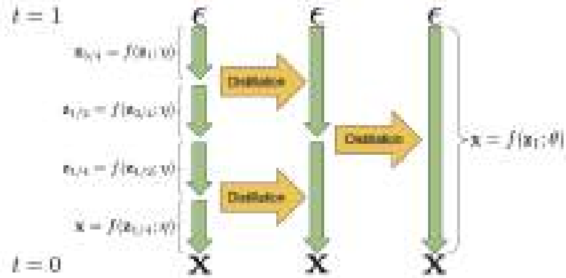


Figure 25.12: Progressive distillation. From Figure 1 of [SH22]. Used with kind permission of Tim Salimans.

Gaussian noise through the generator. The benefits over a single stage GAN is that both the generator and discriminator are solving a much simpler problem, resulting in increased mode coverage, and better training stability. The benefit over a standard diffusion model is that we can generate high quality samples in many fewer steps.

25.5.3 Distillation

In this section, we discuss the **progressive distillation** method of [SH22], which provides a way to create a diffusion model that only needs a small number of steps to create high quality samples. The basic idea is follows. First we train a DDPM model in the usual way, and sample from it using the DDIM method; we treat this as the "teacher" model. We use this to generate intermediate latent states, and train a "student" model to predict the output of the teacher on every second step, as shown in Figure 25.12. After the student has been trained, it can generate results that are as good as the teacher, but in half the number of steps. This student can then teach a new generation of even faster students. See Algorithm 25.4 for the pseudocode, which should be compared to Algorithm 25.3 for the standard training procedure. Note that each round of teaching becomes faster, because the teachers become smaller, so the total time to perform the distillation is relatively small. The resulting model can generate high quality samples in as few as 4 steps.

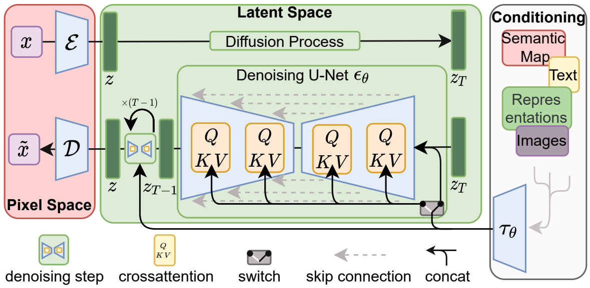


Figure 25.13: Combining a VAE with a diffusion model. Here $\mathcal{E}$ and $\mathcal{D}$ are the encoder and decoder of the VAE. The diffusion model conditions on the inputs either by using concatentation or by using a cross-attention mechanism. From Figure 3 of [Rom+22]. Used with kind permission of Robin Rombach.

Algorithm 25.3: Standard training

Input: Model $\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{z}_t)$ to be trained
Input: Dataset $\mathcal{D}$
Input: Loss weight function w

while *not converged* **do**
- $\boldsymbol{x} \sim \mathcal{D}$
- $t \sim \text{Unif}(0, 1)$
- $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- $\boldsymbol{z}_t = \alpha_t \boldsymbol{x} + \sigma_t \boldsymbol{\epsilon}$
- $\tilde{\boldsymbol{x}} = \boldsymbol{x}$ (Clean data is target)
- $\lambda_t = \log(\alpha_t^2/\sigma_t^2)$ (Log SNR)
- $L_{\boldsymbol{\theta}} = w(\lambda_t)\, \|\tilde{\boldsymbol{x}} - \hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{z}_t)\|_2^2$
- $\boldsymbol{\theta} := \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} L_{\boldsymbol{\theta}}$

Algorithm 25.4: Progressive distillation

Input: Trained teacher model $\hat{\boldsymbol{x}}_{\boldsymbol{\eta}}(\boldsymbol{z}_t)$
Input: Dataset $\mathcal{D}$
Input: Loss weight function w
Input: Student sampling steps N

foreach K *iterations* **do**
- $\boldsymbol{\theta} := \boldsymbol{\eta}$ (Assign student)
- **while** *not converged* **do**
  - $\boldsymbol{x} \sim \mathcal{D}$
  - $t = i/N,\ i \sim \text{Cat}(1, 2, \ldots, N)$
  - $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - $\boldsymbol{z}_t = \alpha_t \boldsymbol{x} + \sigma_t \boldsymbol{\epsilon}$
  - $t' = t - 0.5/N,\ t'' = t - 1/N$
  - $\boldsymbol{z}_{t'} = \alpha_{t'} \hat{\boldsymbol{x}}_{\boldsymbol{\eta}}(\boldsymbol{z}_t) + \frac{\sigma_{t'}}{\sigma_t}(\boldsymbol{z}_t - \alpha_t \hat{\boldsymbol{x}}_{\boldsymbol{\eta}}(\boldsymbol{z}_t))$
  - $\boldsymbol{z}_{t''} = \alpha_{t''} \hat{\boldsymbol{x}}_{\boldsymbol{\eta}}(\boldsymbol{z}_{t'}) + \frac{\sigma_{t''}}{\sigma_{t'}}(\boldsymbol{z}_{t'} - \alpha_{t'} \hat{\boldsymbol{x}}_{\boldsymbol{\eta}}(\boldsymbol{z}_{t'}))$
  - $\tilde{\boldsymbol{x}} = \frac{\boldsymbol{z}_{t''} - (\sigma_{t''}/\sigma_t)\boldsymbol{z}_t}{\alpha_{t''} - (\sigma_{t''}/\sigma_t)\alpha_t}$ (Teacher is target)
  - $\lambda_t = \log(\alpha_t^2/\sigma_t^2)$
  - $L_{\boldsymbol{\theta}} = w(\lambda_t)\, \|\tilde{\boldsymbol{x}} - \hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{z}_t)\|_2^2$
  - $\boldsymbol{\theta} := \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} L_{\boldsymbol{\theta}}$
- $\boldsymbol{\eta} : - = \boldsymbol{\theta}$ (Student becomes next teacher)
- $N := N/2$ (Halve number of sampling steps)

25.5.4 Latent space diffusion

Another approach to speeding up diffusion models for images is to first embed the images into a lower dimensional space, and then fit the diffusion model to the embeddings. This idea has been pursued in several papers.

In the latent diffusion model (**LDM**) of [Rom+22], they adopt a two-stage training scheme, in which they first fit the VAE, augmented with a perceptual loss, and then fit the diffusion model to the embedding. The architecture is illustrated in Figure 25.13. The LDM forms the foundation of the very popular **stable diffusion** system created by **Stability AI**. In the latent score-based generative model (**LSGM**) of [VKK21], they first train a hierarchical VAE, and then jointly train the VAE and a diffusion model.

In addition to speed, an additional advantage of combining diffusion models with autoencoders is that it makes it simple to apply diffusion to many different kinds of data, such as text and graphs: we just need to define a suitable architecture to embed the input domain into a continuous space. Note, however, that it is also possible to define diffusion directly on discrete state spaces, as we discuss in Section 25.7.

So far we have discussed applying diffusion "on top of" a VAE. However, we can also do the reverse, and fit a VAE on top of a DDPM model, where we use the diffusion model to "post process" blurry samples coming from the VAE. See [Pan+22] for details.

25.6 Conditional generation

In this section, we discuss how to generate samples from a diffusion model where we condition on some side information $\boldsymbol{c}$, such as a class label or text prompt.

25.6.1 Conditional diffusion model

The simplest way to control the generation from a generative model is to train it on $(\boldsymbol{c}, \boldsymbol{x})$ pairs so as to maximize the conditional likelihood, $p(\boldsymbol{x}|\boldsymbol{c})$. If the conditioning signal c is a scalar (e.g., a class label), it can be mapped to an embedding vector, and then incorporated into the network by spatial addition, or by using it to modular the group normalization layers. If the input $\boldsymbol{c}$ is another image, we can simply concatenate it with $\boldsymbol{x}_t$ as an extra set of channels. If the input $\boldsymbol{c}$ is a text prompt, we can embed it, and then use spatial addition or cross-attention (see Figure 25.13 for an illustration).

25.6.2 Classifier guidance

One problem with conditional diffusion models is that we need to retrain them for each kind of conditioning that we want to perform. An alternative approach, known as **classifier guidance** was proposed in [DN21b], and allows us to leverage pre-trained discriminative classifiers of the form $p_{\boldsymbol{\phi}}(\boldsymbol{c}|\boldsymbol{x})$ to control the generation process. The idea is as follows. First we use Bayes' rule to write

$$\log p(\boldsymbol{x}|\boldsymbol{c}) = \log p(\boldsymbol{c}|\boldsymbol{x}) + \log p(\boldsymbol{x}) - \log p(\boldsymbol{c}) \tag{25.60}$$

from which the score function becomes

$$\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}|\boldsymbol{c}) = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \log p(\boldsymbol{c}|\boldsymbol{x}) \tag{25.61}$$

We can now use this conditional score to generate samples, rather than the unconditional score. We can further amplify the influence of the conditioning signal by scaling it by a factor $w > 1$:

$$\nabla_{\boldsymbol{x}} \log p_w(\boldsymbol{x}|\boldsymbol{c}) = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) + w\nabla_{\boldsymbol{x}} \log p(\boldsymbol{c}|\boldsymbol{x}) \tag{25.62}$$

In practice, this can be achieved as follows by generating samples from

$$\boldsymbol{x}_{t-1} \sim \mathcal{N}(\boldsymbol{\mu} + w\boldsymbol{\Sigma}\boldsymbol{g}, \boldsymbol{\Sigma}),\ \boldsymbol{\mu} = \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t),\ \boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t),\ \boldsymbol{g} = \nabla_{\boldsymbol{x}_t} \log p_{\boldsymbol{\phi}}(\boldsymbol{c}|\boldsymbol{x}_t) \tag{25.63}$$

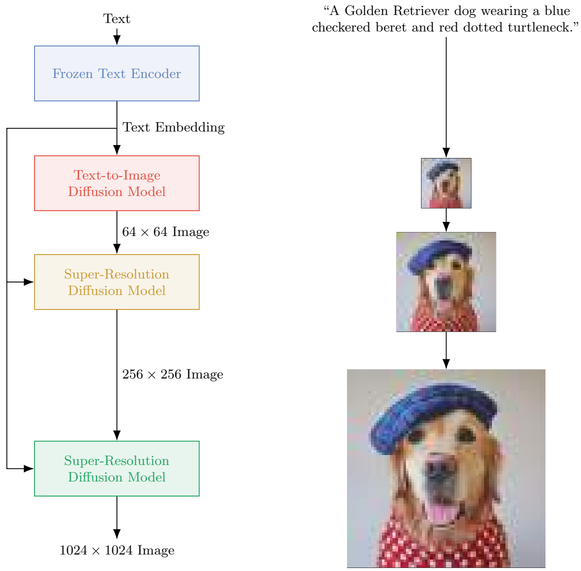


Figure 25.14: Cascaded diffusion model used by the Imagen text-to-image system. From Figure A.5 of [Sah+22b]. Used with kind permission of Saurabh Saxena.

25.6.3 Classifier-free guidance

Unfortunately, $p(\boldsymbol{c}|\boldsymbol{x}_t)$ is a discriminative model, that may ignore many details of the input $\boldsymbol{x}_t$. Hence optimizing along the directions specified by $\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{c}|\boldsymbol{x}_t)$ can give poor results, similar to what happens when we create adversarial images. In addition, we need to train a classifier for each time step, since $\boldsymbol{x}_t$ will differ in its blurriness.

In [HS21], they proposed a technique called **classifier-free guidance**, which derives the classifier from the generative model, using $p(\boldsymbol{c}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{c})p(\boldsymbol{c})}{p(\boldsymbol{x})}$, from which we get

$$\log p(\boldsymbol{c}|\boldsymbol{x}) = \log p(\boldsymbol{x}|\boldsymbol{c}) + \log p(\boldsymbol{c}) - \log p(\boldsymbol{x}) \tag{25.64}$$

This requires learning two generative models, namely $p(\boldsymbol{x}|\boldsymbol{c})$ and $p(\boldsymbol{x})$. However, in practice we can use the same model for this, and simply set $\boldsymbol{c} = \emptyset$ to represent the unconditional case. We then use this implicit classifier to get the following modified score function:

$$\nabla_{\boldsymbol{x}}[\log p(\boldsymbol{x}|\boldsymbol{c}) + w \log p(\boldsymbol{c}|\boldsymbol{x})] = \nabla_{\boldsymbol{x}}[\log p(\boldsymbol{x}|\boldsymbol{c}) + w(\log p(\boldsymbol{x}|\boldsymbol{c}) - \log p(\boldsymbol{x}))] \tag{25.65}$$

$$= \nabla_{\boldsymbol{x}}[(1+w)\log p(\boldsymbol{x}|\boldsymbol{c}) - w \log p(\boldsymbol{x})] \tag{25.66}$$

Larger guidance weight usually results in better individual sample quality, but lower diversity.

25.6.4 Generating high resolution images

In order to generate high resolution images, [Ho+21] proposed to use **cascaded generation**, in which we first train a model to generate 64×64 images, and then train a separate **super-resolution**

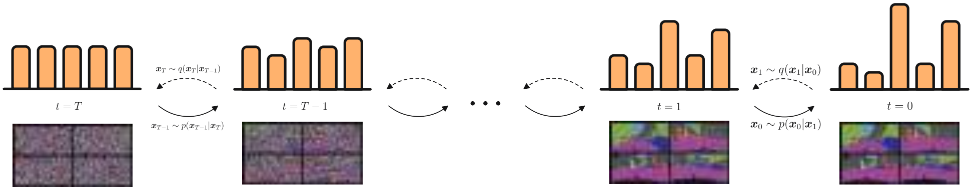


Figure 25.15: Multinomial diffusion model, applied to semantic image segmentation. The input image is on the right, and gets diffused to the noise image on the left. From Figure 1 of [Aus+21]. Used with kind permission of Emiel Hoogeboom.

model to map this to 256×256 or 1024×1024. This approach is used in Google's **Imagen** model [Sah+22b], which is a text-to-image system (see Figure 25.14). Imagen uses large pre-trained text encoder, based on T5-XXL [Raf+20a], combined with a VDM model (Section 25.2.4) based on the U-net architecture, to generate impressive-looking images (see Figure 20.2).

In addition to conditioning on text, it is possible to condition on another image to create a model for **image-to-image translation**. For example, we can learn map a gray-scale image $\boldsymbol{c}$ to a color image $\boldsymbol{x}$, or a corrupted or occluded image $\boldsymbol{c}$ to a clean version $\boldsymbol{x}$. This can be done by training a multi-task conditional diffusion model, as explained in [Sah+22a]. See Figure 20.4 for some sample outputs.

25.7 Diffusion for discrete state spaces

So far in this chapter, we have focused on Gaussian diffusion for generating real-valued data. However it is also possible to define diffusion models for discrete data, such as text or semantic segmentation labels, either by using a continuous latent embedding space (see Section 25.5.4), or by defining diffusion operations directly on the discrete state space, as we discuss beow.

25.7.1 Discrete Denoising Diffusion Probabilistic Models

In this section we discuss the Discrete Denoising Diffusion Probabilistic Model (**D3PM**) of [Aus+21], which defines a discrete time diffusion process directly on the discrete state space. (This builds on prior work such as **multinomial diffusion** [Hoo+21], and the original diffusion paper of [SD+15b].)

The basic idea is illustrated in Figure 25.15 in the context of semantic segmentation, which associates a categorical label to each pixel in an image. On the right we illustrate some sample images, and the corresponding categorical distribution that they induce over a single pixel. We gradually transform these pixel-wise distributions to the uniform distribution, using a stochastic sampling process that we describe below. We then learn a neural network to invert this process, so it can generate discrete data from noise; in the diagram, this corresponds to moving from left to right.

To ensure efficient training, we require that we can efficiently sample from $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ for an abritrary timestep t, so we can randomly sample time steps when optimizing the variational bound in Equation (25.27). In addition, we require that $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ have a tractable form, so we can efficiently compute the KL terms

$$L_{t-1}(\boldsymbol{x}_0) = \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} D_{\mathbb{KL}} \left(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) \right) \tag{25.67}$$

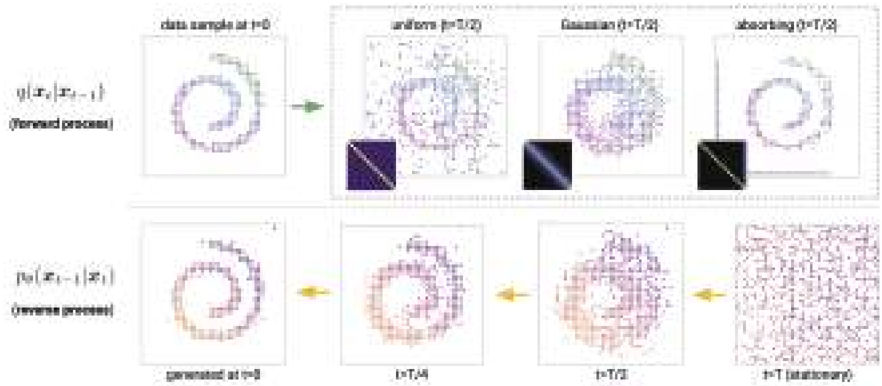


Figure 25.16: D3PM forward and (learned) reverse proccess applied to a quantized Swiss roll. Each dot represents a 2d categorical variable. Top: samples from the uniform, discretized Gaussian, and absorbing state models, along with corresponding transition matrices $\mathbf{Q}$*. Bottom: samples from a learned discretized Gaussian reverse process. From Figure 1 of [Aus+21]. Used with kind permission of Jacob Austin.*

Finally, it is useful if the forwards process converges to a known stationary distribution, $\pi(\boldsymbol{x}_T)$, which we can use for our generative prior $p(\boldsymbol{x}_T)$; this ensures $D_{\mathbb{KL}}\left(q(\boldsymbol{x}_T|\boldsymbol{x}_0) \parallel p(\boldsymbol{x}_T)\right) = 0$.

To satisfy the above criteria, we assume the state consists of D independent blocks, each representing a categorical variable, $x_t \in \{1, \ldots, K\}$; we represent this by the one-hot row vector $\boldsymbol{x}_0$. In general, this will represent a vector of probabilities. We then define the forwards diffusion kernel as follows:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \text{Cat}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}\mathbf{Q}_t) \tag{25.68}$$

where $[\mathbf{Q}_t]_{ij} = q(x_t = j|x_{t-1} = k)$ is a row stochastic transition matrix. (We discuss how to define $\mathbf{Q}_t$ in Section 25.7.2.)

We can derive the t-step marginal of the forwards process as follows:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \text{Cat}(\boldsymbol{x}_t|\boldsymbol{x}_0\overline{\mathbf{Q}}_t), \ \overline{\mathbf{Q}}_t = \mathbf{Q}_1\mathbf{Q}_2\cdots\mathbf{Q}_t \tag{25.69}$$

Similarly, we can reverse the forwards process as follows:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \cancel{\boldsymbol{x}_0})q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} = \text{Cat}\left(\boldsymbol{x}_{t-1}|\frac{\boldsymbol{x}_t\mathbf{Q}_t^{\mathsf{T}} \odot \boldsymbol{x}_0\overline{\mathbf{Q}}_{t-1}}{\boldsymbol{x}_0\overline{\mathbf{Q}}_t\boldsymbol{x}_t^{\mathsf{T}}}\right) \tag{25.70}$$

We discuss how to define the generative process $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ in Section 25.7.3. Since both distrbutions factorize, we can easily compute the KL distributions in Equation (25.67) by summing the KL for each dimension.

25.7.2 Choice of Markov transition matrices for the forward processes

In this section, we give some examples of how to represent the transition matrix $\mathbf{Q}_t$.

One simple approach is to use $\mathbf{Q}_t = (1-\beta_t)\mathbf{I} + \beta_t/K$, which we can write in scalar form as follows:

$$[\mathbf{Q}_t]_{ij} = \begin{cases} 1 - \frac{K-1}{K}\beta_t & \text{if } i = j \\ \frac{1}{K}\beta_t & \text{if } i \neq j \end{cases} \tag{25.71}$$

Intuïtively, this adds a little amount of uniform noise over the K classes, and with a large probability, $1-\beta_t$, we sample from $\boldsymbol{x}_{t-1}$. We call this the uniform kernel. Since this is a doubly stochastic matrix with strictly positive entries, the stationary distributon is uniform. See Figure 25.16 for an illustration.

In the case of the uniform kernel, one can show [Hoo+21] that the marginal distribution is given by

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \text{Cat}(\boldsymbol{x}_t|\overline{\alpha}_t \boldsymbol{x}_0 + (1-\overline{\alpha}_t)/K) \tag{25.72}$$

where $\alpha_t = 1 - \beta_t$ and $\overline{\alpha}_t = \prod_{\tau=1}^t \alpha_\tau$. This is similar to the Gaussian case discussed in Section 25.2. Furthermore, we can derive the posterior distribution as follows:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \text{Cat}(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}_{\text{post}}(\boldsymbol{x}_t, \boldsymbol{\theta}_0)),\ \boldsymbol{\theta}_{\text{post}}(\boldsymbol{x}_t, \boldsymbol{\theta}_0) = \tilde{\boldsymbol{\theta}} / \sum_{k=1}^K \tilde{\theta}_k \tag{25.73}$$

$$\tilde{\boldsymbol{\theta}} = [\alpha_t \boldsymbol{x}_t + (1-\alpha_t)/K] \odot [\overline{\alpha}_{t-1} \boldsymbol{x}_0 + (1-\overline{\alpha}_{t-1})/K] \tag{25.74}$$

Another option is to define a special **absorbing state** m, representing a MASK token, which we transition into with probability β_t. Formally, we have $\mathbf{Q}_t = (1-\beta_t)\mathbf{I} + \beta_t \mathbf{1} \boldsymbol{e}_m^\mathsf{T}$, or, in scalar form,

$$[\mathbf{Q}_t]_{ij} = \begin{cases} 1 & \text{if } i = j = m \\ 1-\beta_t & \text{if } i = j \neq m \\ \beta_t & \text{if } j = m, i \neq m \end{cases} \tag{25.75}$$

This converges to a point-mass distribution on state m. See Figure 25.16 for an illustration.

Another option, suitable for quantized ordinal values, is to use a **discretized Gaussian**, that transitions to other nearby states, with a probability that depends on how similar the states are in numerical value. If we ensure the transition matrix is doubly stochastic, the resulting stationary distribution will again be uniform. See Figure 25.16 for an illustration.

25.7.3 Parameterization of the reverse process

While it is possible to directly predict the logits $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ using a neural network $f_{\boldsymbol{\theta}}(\boldsymbol{x}_t)$, it is preferable to directly predict the logits of the output, using $\tilde{p}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_0|\boldsymbol{x}_t)$; we can then combine this with the analytical expression for $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ to get

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) \propto \sum_{\tilde{\boldsymbol{x}}_0} q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \tilde{\boldsymbol{x}}_0) \tilde{p}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_0|\boldsymbol{x}_t) \tag{25.76}$$

(The sum over $\tilde{\boldsymbol{x}}_0$ takes $O(DK)$ time, if there are D dimensions, each with K values.) One advantage of this approach, compared to directly learning $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, is that the model will automatically satisfy any sparsity constraints in $\mathbf{Q}_t$. In addition, we can perform inference with k steps at a time, by predicting

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-k}|\boldsymbol{x}_t) \propto \sum_{\tilde{\boldsymbol{x}}_0} q(\boldsymbol{x}_{t-k}|\boldsymbol{x}_t, \tilde{\boldsymbol{x}}_0)\tilde{p}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_0|\boldsymbol{x}_t) \tag{25.77}$$

Note that, in the multi-step Gaussian case, we require more complex models to handle multimodaility (see Section 25.5.2). By contrast, discrete distributions already have this flexibility built in.

25.7.4 Noise schedules

In this section we discuss how to choose the noise schedule for β_t. For discretized Gaussian diffusion, [Aus+21] propose to linearly increase the variance of the Gaussian noise before the discretization step. For uniform diffusion, we can use a cosine schedule of the form $\alpha_t = \cos(\frac{t/T+s}{1+s}\frac{\pi}{2})$, where $s = 0.08$, as proposed in [ND21]. (Recall that $\beta_t = 1 - \alpha_t$, so the noise increases over time.) For masked diffusion, we can use a schedule of the form $\beta_t = 1/(T - t + 1)$, as proposed in [SD+15b].

25.7.5 Connections to other probabilistic models for discrete sequences

There are interesting connections between D3PM and other probabilistic text models. For example, suppose we define the transition matrix as a combination of the unifrom transition matrix and an absorbing MASK state, i.e., $\mathbf{Q} = \alpha\mathbf{1}\boldsymbol{e}_m^\mathsf{T} + \beta\mathbf{1}\mathbf{1}^\mathsf{T}/K + (1-\alpha-\beta)\mathbf{I}$. For a one-step diffusion process in which $q(\boldsymbol{x}_1|\boldsymbol{x}_0)$ replaces $\alpha = 10\%$ of the tokens with MASK, and $\beta = 5\%$ uniformly at random, we recover the same objective that is used to train the **BERT** language model, namely

$$L_0(\boldsymbol{x}_0) = -\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1) \tag{25.78}$$

(This follows since $L_T = 0$, and there are no other time steps used in the variational bound in Equation (25.27).)

Now consider a diffusion process that deterministically masks tokens one-by-one. For a sequence of length $N = T$, we have $q([\boldsymbol{x}_t]_i|\boldsymbol{x}_0) = [\boldsymbol{x}_0]_i$ if $i < N - t$ (pass through), else $[\boldsymbol{x}_t]_i$ is set to MASK. Because this is a deterministic process, the posterior $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ is a delta function on the $\boldsymbol{x}_t$ with one fewer mask tokens. One can then show that the KL term becomes $D_{\mathbb{KL}}\left(q([\boldsymbol{x}_t]_i|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}([\boldsymbol{x}_{t-1}]_i|\boldsymbol{x}_t)\right) = -\log p_{\boldsymbol{\theta}}([\boldsymbol{x}_0]_i|\boldsymbol{x}_t)$, which is the standard cross-entropy loss for an autoregressive model.

Finally one can show that generative **masked language models**, such as [WC19; Gha+19], also correspond to discrete diffusion processes: the sequence starts will all locations masked out, and each step, a set of tokens are generated, given the previous sequence. The **MaskGIT** method of [Cha+22] uses a similar procedure in the image domain, after applying vector quantization to image patches. These parallel, iterative decoders are much faster than sequential autoregressive decoders. See Figure 25.17 for an illustration.

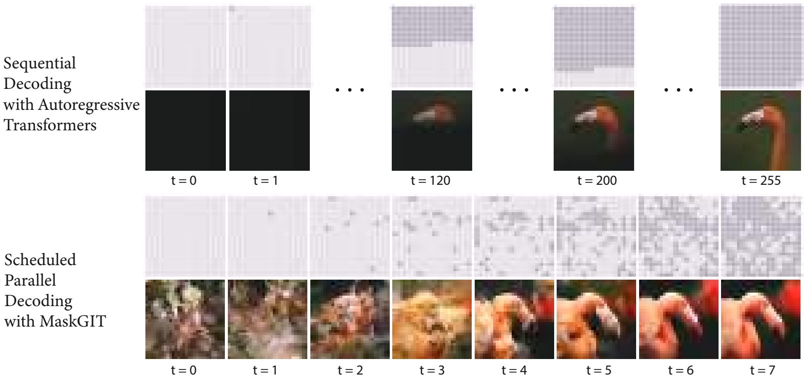


Figure 25.17: Comparison of sequential image generation with a transformer (top) vs parallel generation with MaskGIT (bottom). All pixels start out in the MASK state, denoted by light gray. In the transformer, we generate one pixel at a time, so it takes 256 steps for the whole image. In the MaskGIT method, multiple states are generated in parallel, which only takes 8 steps. From Figure 2 of [Cha+22]. Used with kind permission of Huiwen Chang.

26 Generative adversarial networks

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

26.1 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.

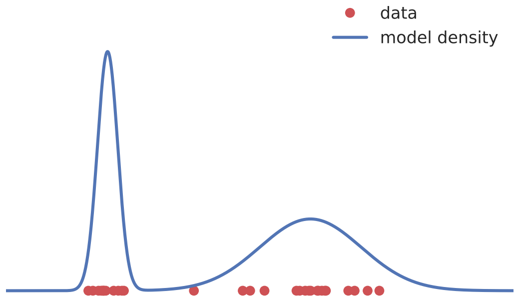
To develop a probabilistic formulation for GANs, it is useful to first distinguish between two types of probabilistic models: "**prescribed probabilistic models**" and "**implicit probabilistic models**" [DG84]. Prescribed probabilistic models, which we will call **explicit probabilistic models**, provide an explicit parametric specification of the distribution of an observed random variable $\boldsymbol{x}$, specifying a log-likelihood function $\log q_\theta(\boldsymbol{x})$ with parameters $\boldsymbol{\theta}$. 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. We illustrate the difference between implicit and explicit models in Figure 26.1.

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 21). Implicit generative models use a latent variable $\boldsymbol{z}$ and transform it using a deterministic function $G_{\boldsymbol{\theta}}$ that maps from $\mathbb{R}^m \rightarrow \mathbb{R}^d$ using parameters $\boldsymbol{\theta}$. 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:

$$\boldsymbol{x} = G_{\boldsymbol{\theta}}(\boldsymbol{z}'); \qquad \boldsymbol{z}' \sim q(\boldsymbol{z}) \tag{26.1}$$

$$q_\theta(\boldsymbol{x}) = \frac{\partial}{\partial x_1} \cdots \frac{\partial}{\partial x_d} \int_{\{G_{\boldsymbol{\theta}}(\boldsymbol{z}) \leq \boldsymbol{x}\}} q(\boldsymbol{z}) d\boldsymbol{z}, \tag{26.2}$$

where $q(\boldsymbol{z})$ is a distribution over latent variables that provides the external source of randomness. Equation (26.2) is the definition of the transformed density $q_\theta(\boldsymbol{x})$ defined as the derivative of a cumulative distribution function, and hence integrates the distribution $q(\boldsymbol{z})$ over all events defined



(a) Prescribed generative model.

(b) Implicit generative model.

Figure 26.1: 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 $\{G_{\boldsymbol{\theta}}(\boldsymbol{z}) \leq \boldsymbol{x}\}$. When the latent and data dimension are equal ($m = d$) and the function $G_{\boldsymbol{\theta}}(\boldsymbol{z})$ 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 23). In diffusion models (Chapter 25), 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 by a deep network. Such models are sometimes called **generator networks** or **generative neural samplers**; they can also be throught of as **differentiable simulators**. Unfortunately the integral (26.2) is intractable in these kinds of models, and we may not even be able to determine the set $\{G_{\boldsymbol{\theta}}(\boldsymbol{z}) \leq \boldsymbol{x}\}$. Of course, intractability is also a challenge for explicit latent variable models such as VAEs (Chapter 21), but in the GAN case, the lack of a likelihood term makes the learning problem even harder. Therefore this problem is called **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. (See also `https://poloclub.github.io/ganlab/` for an interactive tutorial.)

26.2 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_θ (with parameters $\boldsymbol{\theta}$)

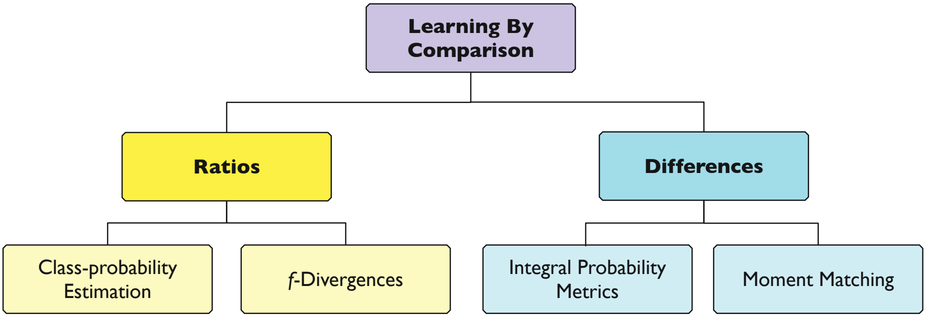


Figure 26.2: Overview of approaches for learning in implicit generative models

and the unknown true data distribution p^*. Recalling equation (26.2), in implicit models we cannot evaluate $q_\theta(\boldsymbol{x})$, 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.*

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 26.2 shows an overview of different approaches for learning by comparison.

26.2.1 Guiding principles

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

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

2. Can be evaluated only using samples from the data and model distribution.

3. Are computationally cheap to evaluate.

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

$$\mathcal{D}(p^*, q) \geq 0; \qquad \mathcal{D}(p^*, q) = 0 \iff p^* = q \tag{26.3}$$

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

$$\mathcal{D}(p^*, q) = \underset{D}{\operatorname{argmax}}\, \mathcal{F}(D, p^*, q) \tag{26.4}$$

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

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

$$\mathcal{F}(D_{\boldsymbol{\phi}}, p^*, q_\theta) = \mathbb{E}_{p^*(\boldsymbol{x})} f(\boldsymbol{x}, \boldsymbol{\phi}) + \mathbb{E}_{q_\theta(\boldsymbol{x})} g(\boldsymbol{x}, \boldsymbol{\phi}) \tag{26.5}$$

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

$$\mathcal{F}(D_{\boldsymbol{\phi}}, p^*, q_\theta) = \mathbb{E}_{p^*(\boldsymbol{x})} f(\boldsymbol{x}, \boldsymbol{\phi}) + \mathbb{E}_{q(\boldsymbol{z})} g(G_{\boldsymbol{\theta}}(\boldsymbol{z}), \boldsymbol{\phi}) \tag{26.6}$$

which can be estimated using Monte Carlo estimation

$$\mathcal{F}(D_{\boldsymbol{\phi}}, p^*, q_\theta) \approx \frac{1}{N}\sum_{i=1}^{N} f(\widehat{\boldsymbol{x}}_i, \boldsymbol{\phi}) + \frac{1}{M}\sum_{i=1}^{M} g(G_{\boldsymbol{\theta}}(\widehat{\boldsymbol{z}}_i), \boldsymbol{\phi}); \qquad \widehat{\boldsymbol{x}}_i \sim p^*(\boldsymbol{x}); \qquad \widehat{\boldsymbol{z}}_i \sim q(\boldsymbol{z}) \tag{26.7}$$

Next, we will see how to instantiate these guiding principles in order to find the functions f and g and thus the objective $\mathcal{F}$ which can be used to train implicit models: class probability estimation (Section 26.2.2), bounds on f-divergences (Section 26.2.3), integral probability metrics (Section 26.2.4), and moment matching (Section 26.2.5).

26.2.2 Density ratio estimation using binary classifiers

One way to compare two distributions p^* and q_θ is to compute their density ratio $r(\boldsymbol{x}) = \frac{p^*(\boldsymbol{x})}{q_\theta(\boldsymbol{x})}$. The distributions are the same if and only if the ratio is 1 everywhere in the support of $q_{\boldsymbol{\theta}}$. Since we cannot evaluate the densities of implicit models, we must instead develop techniques to compute the density ratio from samples alone, following the guiding principles established above.

Fortunately, we can use the trick from Section 2.7.5 which converts density estimation into a binary classification problem to write

$$\frac{p^*(\boldsymbol{x})}{q_\theta(\boldsymbol{x})} = \frac{D(\boldsymbol{x})}{1 - D(\boldsymbol{x})} \tag{26.8}$$

where $D(\boldsymbol{x})$ is the discriminator or critic which is trained to distinguish samples coming from p^* vs $q_{\boldsymbol{\theta}}$.

For parametric classification, we can learn discriminators $D_{\boldsymbol{\phi}}(\boldsymbol{x}) \in [0, 1]$ with parameters $\boldsymbol{\phi}$. Using knowledge and insight about probabilistic classification, we can learn the parameters by minimizing any proper scoring rule [GR07] (see also Section 14.2.1). For the familiar Bernoulli log-loss (or binary cross entropy loss), we obtain the objective:

$$\begin{aligned}
V(q_\theta, p^*) &= \arg\max_{\boldsymbol{\phi}} \mathbb{E}_{p(\boldsymbol{x}|y)p(y)}[y \log D_{\boldsymbol{\phi}}(\boldsymbol{x}) + (1-y)\log(1 - D_{\boldsymbol{\phi}}(\boldsymbol{x}))] \\
&= \arg\max_{\boldsymbol{\phi}} \mathbb{E}_{p(\boldsymbol{x}|y=1)p(y=1)} \log D_{\boldsymbol{\phi}}(\boldsymbol{x}) + \mathbb{E}_{p(\boldsymbol{x}|y=0)p(y=0)} \log(1 - D_{\boldsymbol{\phi}}(\boldsymbol{x})) &(26.9)\\
&= \arg\max_{\boldsymbol{\phi}} \frac{1}{2}\mathbb{E}_{p^*(\boldsymbol{x})} \log D_{\boldsymbol{\phi}}(\boldsymbol{x}) + \frac{1}{2}\mathbb{E}_{q_\theta(\boldsymbol{x})} \log(1 - D_{\boldsymbol{\phi}}(\boldsymbol{x})). &(26.10)
\end{aligned}$$

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

Table 26.1: Proper scoring rules that can be maximized in class probability-based learning of implicit generative models. Based on [ML16].

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 26.1, adapted from [ML16]. The optimal discriminator D is $\frac{p^*(\boldsymbol{x})}{p^*(\boldsymbol{x})+q_\theta(\boldsymbol{x})}$, since:

$$\frac{p^*(\boldsymbol{x})}{q_\theta(\boldsymbol{x})} = \frac{D^*(\boldsymbol{x})}{1-D^*(\boldsymbol{x})} \implies D^*(\boldsymbol{x}) = \frac{p^*(\boldsymbol{x})}{p^*(\boldsymbol{x})+q_\theta(\boldsymbol{x})} \tag{26.11}$$

By substituting the optimal discriminator into the scoring rule (26.10), we can show that the objective V can also be interpreted as the minimization of the Jensen-Shannon divergence.

$$V^*(q_\theta, p^*) = \frac{1}{2}\mathbb{E}_{p^*(\boldsymbol{x})}[\log \frac{p^*(\boldsymbol{x})}{p^*(\boldsymbol{x})+q_\theta(\boldsymbol{x})}] + \frac{1}{2}\mathbb{E}_{q_\theta(\boldsymbol{x})}[\log(1 - \frac{p^*(\boldsymbol{x})}{p^*(\boldsymbol{x})+q_\theta(\boldsymbol{x})})] \tag{26.12}$$

$$= \frac{1}{2}\mathbb{E}_{p^*(\boldsymbol{x})}[\log \frac{p^*(\boldsymbol{x})}{\frac{p^*(\boldsymbol{x})+q_\theta(\boldsymbol{x})}{2}}] + \frac{1}{2}\mathbb{E}_{q_\theta(\boldsymbol{x})}[\log(\frac{q_\theta(\boldsymbol{x})}{\frac{p^*(\boldsymbol{x})+q_\theta(\boldsymbol{x})}{2}})] - \log 2 \tag{26.13}$$

$$= \frac{1}{2}D_{\mathbb{KL}}\left(p^* \parallel \frac{p^*+q_\theta}{2}\right) + \frac{1}{2}D_{\mathbb{KL}}\left(q_\theta \parallel \frac{p^*+q_\theta}{2}\right) - \log 2 \tag{26.14}$$

$$= JSD(p^*, q_\theta) - \log 2 \tag{26.15}$$

where JSD denotes the Jensen-Shannon divergence:

$$JSD(p^*, q_\theta) = \frac{1}{2}D_{\mathbb{KL}}\left(p^* \parallel \frac{p^*+q_\theta}{2}\right) + \frac{1}{2}D_{\mathbb{KL}}\left(q_\theta \parallel \frac{p^*+q_\theta}{2}\right) \tag{26.16}$$

This establishes a connection between *optimal* binary classification and distributional divergences. By using binary classification, we were able to compute the distributional divergence using only samples, which is the important property needed for learning implicit generative models; as expressed in the guiding principles (Section 26.2.1), we have turned an intractable estimation problem — how to estimate the JSD divergence, into an optimization problem — how to learn a classifier which can be used to approximate that divergence.

We would like to train the parameters $\boldsymbol{\theta}$ of generative model to minimize the divergence:

$$\min_{\boldsymbol{\theta}} JSD(p^*, q_\theta) = \min_{\boldsymbol{\theta}} V^*(q_\theta, p^*) + \log 2 \tag{26.17}$$

$$= \min_{\boldsymbol{\theta}} \frac{1}{2}\mathbb{E}_{p^*(\boldsymbol{x})} \log D^*(\boldsymbol{x}) + \frac{1}{2}\mathbb{E}_{q_\theta(\boldsymbol{x})} \log(1 - D^*(\boldsymbol{x})) + \log 2 \tag{26.18}$$

Since we do not have access to the optimal classifier D^* but only to the neural approximation $D_{\boldsymbol{\phi}}$ obtained using the optimization in (26.10) , this results in a min-max optimization problem:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \frac{1}{2}\mathbb{E}_{p^*(\boldsymbol{x})}[\log D_{\boldsymbol{\phi}}(\boldsymbol{x})] + \frac{1}{2}\mathbb{E}_{q_\theta(\boldsymbol{x})}[\log(1 - D_{\boldsymbol{\phi}}(\boldsymbol{x}))] \tag{26.19}$$

By replacing the generating procedure (26.1) in (26.19) we obtain the objective in terms of the latent variables $\boldsymbol{z}$ of the implicit generative model:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \frac{1}{2}\mathbb{E}_{p^*(\boldsymbol{x})}[\log D_{\boldsymbol{\phi}}(\boldsymbol{x})] + \frac{1}{2}\mathbb{E}_{q(\boldsymbol{z})}[\log(1 - D_{\boldsymbol{\phi}}(G_{\boldsymbol{\theta}}(\boldsymbol{z})))], \tag{26.20}$$

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

Beyond the use of the Bernoulli scoring rule used above, other scoring rules have been used to train generative models via min-max optimization. The Brier scoring rule, which under discriminator optimality conditions can be shown to correspond to minimizing the Pearson χ^2 divergence via similar arguments as the ones shown above has lead to LS-GAN [Mao+17]. The hinge scoring rule has become popular [Miy+18b; BDS18], and under discriminator optimality conditions corresponds to minimizing the total variational distance [NWJ+09].

The connection between proper scoring rules and distributional divergences allows the construction of convergence guarantees for the learning criteria above, under infinite capacity of the discriminator and generator: since the minimizer of distributional divergence is the true data distribution (Equation 26.3), if the discriminator is optimal and the generator has enough capacity, it will learn the data distribution. In practice however, this assumption will not hold, as discriminators are rarely optimal; we will discuss this at length in Section 26.3.

26.2.3 Bounds on f-divergences

As we saw with the appearance of the Jensen-Shannon divergence in the previous section, we can consider directly using a measure of distributional divergence to derive methods for learning in implicit models. One general class of divergences are the f-divergences (Section 2.7.1) defined as:

$$\mathcal{D}_f\left[p^*(\boldsymbol{x}) \| q_\theta(\boldsymbol{x})\right] = \int q_\theta(\boldsymbol{x}) f\left(\frac{p^*(\boldsymbol{x})}{q_\theta(\boldsymbol{x})}\right) d\boldsymbol{x} \tag{26.21}$$

where f is a convex function such that $f(1) = 0$. For different choices of f, we can recover known distributional divergences such as the KL, reverse KL, and Jensen-Shannon divergence. We discuss such connections in Section 2.7.1, and provide a summary in Table 26.2.

To evaluate Equation (26.21) we will need to evaluate the density of the data $p^*(\boldsymbol{x})$ and the model $q_\theta(\boldsymbol{x})$, neither of which are available. In the previous section we overcame the challenge of evaluating the density ratio by transforming it into a problem of binary classification. In this section, we will instead look towards the role of lower bounds on f-divergences, which is an approach for tractability that is also used for variational inference (Chapter 10).

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

Table 26.2: Standard divergences as f divergences for various choices of f. The optimal critic is written as a function of the density ratio $r(\boldsymbol{x}) = \frac{p^*(\boldsymbol{x})}{q_\theta(\boldsymbol{x})}$.

f-divergences have a widely-developed theory in convex analysis and information theory. Since the function f in Equation (26.21) is convex, we know that we can find a tangent that bounds it from below. The variational formulation of the f-divergence is [NWJ10b; NCT16c]:

$$\mathcal{D}_f \left[p^*(\boldsymbol{x}) \| q_\theta(\boldsymbol{x})\right] = \int q_\theta(\boldsymbol{x}) f\left(\frac{p^*(\boldsymbol{x})}{q_\theta(\boldsymbol{x})}\right) d\boldsymbol{x} \tag{26.22}$$

$$= \int q_\theta(\boldsymbol{x}) \sup_{t:\mathcal{X}\to\mathbb{R}} \left[t(\boldsymbol{x}) \frac{p^*(\boldsymbol{x})}{q_\theta(\boldsymbol{x})} - f^\dagger(t(\boldsymbol{x}))\right] d\boldsymbol{x} \tag{26.23}$$

$$= \int \sup_{t:\mathcal{X}\to\mathbb{R}} p^*(\boldsymbol{x}) t(\boldsymbol{x}) - q_\theta(\boldsymbol{x}) f^\dagger(t(\boldsymbol{x})) d\boldsymbol{x} \tag{26.24}$$

$$\geq \sup_{t\in\mathcal{T}} \mathbb{E}_{p^*(\boldsymbol{x})}[t(\boldsymbol{x})] - \mathbb{E}_{q_\theta(\boldsymbol{x})}[f^\dagger(t(\boldsymbol{x}))]. \tag{26.25}$$

In the second line we use the result from convex analysis, discussed Supplementary Section 6.3, that re-expresses the convex function f using $f(u) = \sup_t ut - f^\dagger(t)$, where $f^\dagger$ is the convex conjugate of the function f, and t is a parameter we optimize over. Since we apply f at $u = \frac{p^*(\boldsymbol{x})}{q_\theta(\boldsymbol{x})}$ for all $\boldsymbol{x} \in \mathcal{X}$, we make the parameter t be a function $t(\boldsymbol{x})$. The final inequality comes from replacing the supremum over all functions from the data domain $\mathcal{X}$ to $\mathbb{R}$ with the supremum over a family of functions $\mathcal{T}$ (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 (26.25) follows the general desired form of Equation 26.5: it is the difference of two expectations, and these expectations can be computed by Monte Carlo estimation using only samples, as in Equation (26.7); despite starting with an objective (Equation 26.21) 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 (26.25) that allows learning both the generator and critic parameters. We use a critic D with parameters $\boldsymbol{\phi}$ to estimate the bound, and then optimize the parameters $\boldsymbol{\theta}$ of the generator to minimize the approximation of the f-divergence provided by the critic (we replace t above with $D_{\boldsymbol{\phi}}$, to retain standard GAN notation):

$$\min_{\boldsymbol{\theta}} \mathcal{D}_f(p^*, q_\theta) \geq \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \mathbb{E}_{p^*(\boldsymbol{x})}[D_{\boldsymbol{\phi}}(\boldsymbol{x})] - \mathbb{E}_{q_\theta(\boldsymbol{x})}[f^\dagger(D_{\boldsymbol{\phi}}(\boldsymbol{x}))] \tag{26.26}$$

$$= \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \mathbb{E}_{p^*(\boldsymbol{x})}[D_{\boldsymbol{\phi}}(\boldsymbol{x})] - \mathbb{E}_{q(\boldsymbol{z})}[f^\dagger(D_{\boldsymbol{\phi}}(G_{\boldsymbol{\theta}}(\boldsymbol{z})))] \tag{26.27}$$

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 26.27. 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 26.2 for f-divergences and Equation (26.11) for scoring rules).

26.2.4 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 (Section 2.7.2) defined as:

$$I_{\mathcal{F}}(p^*(\boldsymbol{x}), q_\theta(\boldsymbol{x})) = \sup_{f \in \mathcal{F}} \left| \mathbb{E}_{p^*(\boldsymbol{x})} f(\boldsymbol{x}) - \mathbb{E}_{q_\theta(\boldsymbol{x})} f(\boldsymbol{x}) \right|. \tag{26.28}$$

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 satisfying the conditions for distributional divergences $\mathcal{D}(p^*, q) \geq 0; \quad \mathcal{D}(p^*, q) = 0 \iff p^* = q$ (Equation (26.3)), 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 $z(\boldsymbol{x}) = 0$. This choice of $\mathcal{F}$ entails that regardless of the two distributions chosen, the value in Equation 26.28 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]:

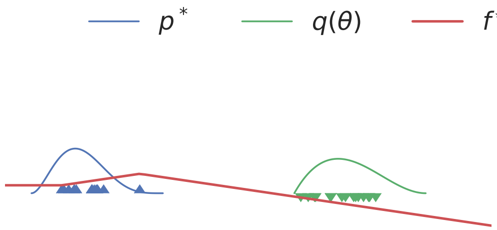
$$W_1(p^*(\boldsymbol{x}), q_\theta(\boldsymbol{x})) = \sup_{f : \|f\|_{\text{Lip}} \leq 1} \mathbb{E}_{p^*(\boldsymbol{x})} f(\boldsymbol{x}) - \mathbb{E}_{q_\theta(\boldsymbol{x})} f(\boldsymbol{x}) \tag{26.29}$$

We show an example of a Wasserstein critic in Figure 26.3a. The supremum over the set of 1-Lipschitz functions is intractable for most cases, which again suggests the introduction of a learned critic:

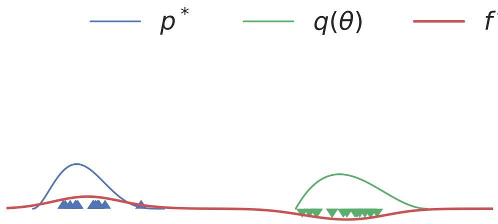
$$W_1(p^*(\boldsymbol{x}), q_\theta(\boldsymbol{x})) = \sup_{f : \|f\|_{\text{Lip}} \leq 1} \mathbb{E}_{p^*(\boldsymbol{x})} f(\boldsymbol{x}) - \mathbb{E}_{q_\theta(\boldsymbol{x})} f(\boldsymbol{x}) \tag{26.30}$$

$$\geq \max_{\boldsymbol{\phi} : \|D_{\boldsymbol{\phi}}\|_{\text{Lip}} \leq 1} \mathbb{E}_{p^*(\boldsymbol{x})} D_{\boldsymbol{\phi}}(\boldsymbol{x}) - \mathbb{E}_{q_\theta(\boldsymbol{x})} D_{\boldsymbol{\phi}}(\boldsymbol{x}), \tag{26.31}$$

where the critic $D_{\boldsymbol{\phi}}$ 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.



(a) Optimal Wasserstein critic.



(b) Optimal MMD critic.

Figure 26.3: Optimal critics in Integral Probability Metrics (IPMs). Generated by ipm_divergences.ipynb

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

$$\min_{\boldsymbol{\theta}} W_1(p^*(\boldsymbol{x}), q_\theta(\boldsymbol{x})) \geq \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}:\|D_{\boldsymbol{\phi}}\|_{\text{Lip}}\leq 1} \mathbb{E}_{p^*(\boldsymbol{x})} D_{\boldsymbol{\phi}}(\boldsymbol{x}) - \mathbb{E}_{q_\theta(\boldsymbol{x})} D_{\boldsymbol{\phi}}(\boldsymbol{x}) \tag{26.32}$$

$$= \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}:\|D_{\boldsymbol{\phi}}\|_{\text{Lip}}\leq 1} \mathbb{E}_{p^*(\boldsymbol{x})} D_{\boldsymbol{\phi}}(\boldsymbol{x}) - \mathbb{E}_{q(\boldsymbol{z})} D_{\boldsymbol{\phi}}(G_{\boldsymbol{\theta}}(\boldsymbol{z})) \tag{26.33}$$

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 obtain the **maximum mean discrepancy** (**MMD**) discussed in Section 2.7.3:

$$\text{MMD}(p^*(\boldsymbol{x}), q_\theta(\boldsymbol{x})) = \sup_{f:\|f\|_{RKHS}=1} \mathbb{E}_{p^*(\boldsymbol{x})} f(\boldsymbol{x}) - \mathbb{E}_{q_\theta(\boldsymbol{x})} f(\boldsymbol{x}). \tag{26.34}$$

We show an example of an MMD critic in Figure 26.3b. 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):

$$\text{MMD}^2(p^*, q_\theta) = \mathbb{E}_{p^*(\boldsymbol{x})}\mathbb{E}_{p^*(\boldsymbol{x}')}\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') - 2\mathbb{E}_{p^*(\boldsymbol{x})}\mathbb{E}_{q_\theta(\boldsymbol{y})}\mathcal{K}(\boldsymbol{x}, \boldsymbol{y}) + \mathbb{E}_{q_\theta(\boldsymbol{y})}\mathbb{E}_{q_\theta(\boldsymbol{y}')}\mathcal{K}(\boldsymbol{y}, \boldsymbol{y}') \tag{26.35}$$

$$= \mathbb{E}_{p^*(\boldsymbol{x})}\mathbb{E}_{p^*(\boldsymbol{x}')}\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') - 2\mathbb{E}_{p^*(\boldsymbol{x})}\mathbb{E}_{q(\boldsymbol{z})}\mathcal{K}(\boldsymbol{x}, G_{\boldsymbol{\theta}}(\boldsymbol{z})) + \mathbb{E}_{q(\boldsymbol{z})}\mathbb{E}_{q(\boldsymbol{z}')}\mathcal{K}(G_{\boldsymbol{\theta}}(\boldsymbol{z}), G_{\boldsymbol{\theta}}(\boldsymbol{z}')) \tag{26.36}$$

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

$$\min_{\boldsymbol{\theta}} \text{MMD}^2(p^*, q_\theta) \tag{26.37}$$

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 ζ such that $\mathcal{K}(\zeta(\boldsymbol{x}), \zeta(\boldsymbol{x}'))$ is a valid kernel; luckily, we can use that for any characteristic kernel $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}')$ and injective function ζ, $\mathcal{K}(\zeta(\boldsymbol{x}), \zeta(\boldsymbol{x}')))$ 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 $p^*(\boldsymbol{x})$ and the model distribution $q_\theta(\boldsymbol{x})$, 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 ζ with parameters $\boldsymbol{\phi}$ [Li+17b]:

$$\min_{\boldsymbol{\theta}} \text{MMD}_{\zeta}{}^2(p_{\mathcal{D}}, q_\theta) \tag{26.38}$$

$$\begin{aligned} &= \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \mathbb{E}_{p^*(\boldsymbol{x})} \mathbb{E}_{p^*(\boldsymbol{x}')} \mathcal{K}(\zeta_{\boldsymbol{\phi}}(\boldsymbol{x}), \zeta_{\boldsymbol{\phi}}(\boldsymbol{x}')) \\ &\qquad - 2\mathbb{E}_{p^*(\boldsymbol{x})} \mathbb{E}_{q_\theta(\boldsymbol{y})} \mathcal{K}(\zeta_{\boldsymbol{\phi}}(\boldsymbol{x}), \zeta_{\boldsymbol{\phi}}(\boldsymbol{y})) \\ &\qquad + \mathbb{E}_{q_\theta(\boldsymbol{y})} \mathbb{E}_{q_\theta(\boldsymbol{y}')} \mathcal{K}(\zeta_{\boldsymbol{\phi}}(\boldsymbol{y}), \zeta_{\boldsymbol{\phi}}(\boldsymbol{y}')) \end{aligned} \tag{26.39}$$

where $\zeta_{\boldsymbol{\phi}}$ is regularized to be injective, though this is sometimes relaxed [Bin+18]. Unlike the Wasserstein distance and f-divergences, Equation (26.39) can be estimated using Monte Carlo estimation, without requiring a lower bound on the original objective.

26.2.5 Moment matching

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:

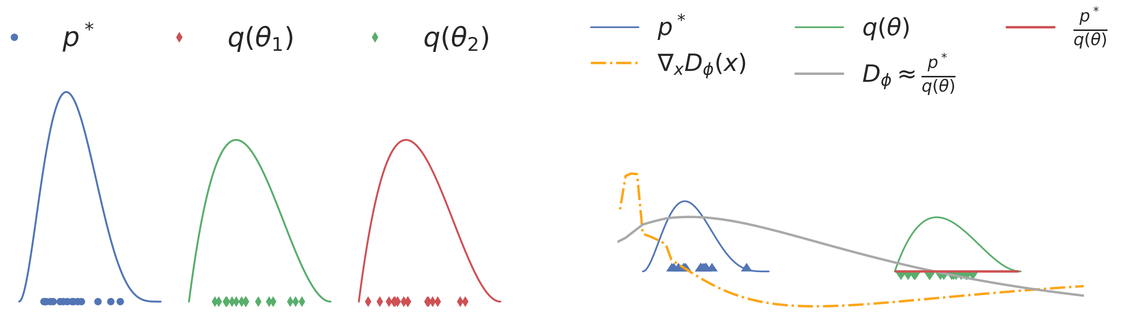
$$\min_{\boldsymbol{\theta}} \left\| \mathbb{E}_{p^*(\boldsymbol{x})} s(\boldsymbol{x}) - \mathbb{E}_{q_\theta(\boldsymbol{x})} s(\boldsymbol{x}) \right\|_2^2 \tag{26.40}$$

where $m(\boldsymbol{\theta}) = \mathbb{E}_{q_\theta(\boldsymbol{x})} s(\boldsymbol{x})$ is the *moment function*. The choice of statistic $s(\boldsymbol{x})$ 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 $p^*(\boldsymbol{x}) = q_\theta(\boldsymbol{x})$. To see that not all functions s satisfy this requirement consider the function $s(\boldsymbol{x}) = \boldsymbol{x}$: simply matching the means of two distributions is not sufficient to match higher moments (such as variance). For likelihood based models the score function $s(\boldsymbol{x}) = \log q_\theta(\boldsymbol{x})$ 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 an RHKS. 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 $D_{\boldsymbol{\phi}}$ together with the features of the learned critic: $s_{\boldsymbol{\phi}}(\boldsymbol{x}) = [\nabla_{\boldsymbol{\phi}} D_{\boldsymbol{\phi}}(\boldsymbol{x}), h_1(\boldsymbol{x}), \ldots, h_n(\boldsymbol{x})]$ where $h_1(\boldsymbol{x}), \ldots, h_n(\boldsymbol{x})$ are the hidden activations of the learned critic. Both features and gradients are needed: the gradients $\nabla_{\boldsymbol{\phi}} D_{\boldsymbol{\phi}}(\boldsymbol{x})$ are required to ensure the estimator for the parameters $\boldsymbol{\theta}$ is consistent, since the number of moments $s(\boldsymbol{x})$ 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 $h_i(\boldsymbol{x})$ 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.

26.2.6 On density ratios and differences

We have seen how density ratios (Sections 26.2.2 and 26.2.3) and density differences (Section 26.2.4) can be used to define training objectives for implicit generative models. We now explore some of the distinctions between using ratios and differences for learning by comparison, as well as explore the



(a) Failure of the KL divergence to distinguish between distributions with non-overlapping support: $D_{\mathbb{KL}}(p^* \parallel q_{\theta_1}) = D_{\mathbb{KL}}(p^* \parallel q_{\theta_2}) = \infty$, despite q_{θ_2} being closer to p^* than q_{θ_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 26.4: 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

effects of using approximations to these objectives using function classes such as neural networks has on these distinctions.

One often stated downside of using divergences that rely on density ratios (such as f-divergences) is their poor behavior when the distributions p^* and q_θ do not have overlapping support. For non-overlapping support, the density ratio $\frac{p^*}{q_\theta}$ will be ∞ in the parts of the space where $p^*(\boldsymbol{x}) > 0$ but $q_\theta(\boldsymbol{x}) = 0$, and 0 otherwise. In that case, the $D_{\mathbb{KL}}(p^* \parallel q_\theta) = \infty$ and the $JSD(p^*, q_\theta) = \log 2$, regardless of the value of $\boldsymbol{\theta}$. Thus *f-divergences cannot distinguish between different model distributions when they do not have overlapping support with the data distribution*, as visualized in Figure 26.4a. This is in contrast with difference based methods such as IPMs such as the Wasserstein distance and the MMD, which have smoothness requirements built in the *definition* of the method, by constraining the norm of the critic (Equations (26.29) and (26.34)). We can see the effect of these constraints in Figure 26.3: both the Wasserstein distance and the MMD provide useful signal in the case of distributions with non-overlapping support.

While the *definition* of f-divergences relies on density ratios (Equation (26.21)), we have seen that to train implicit generative models we use approximations to those divergences obtained using a parametric critic $D_{\boldsymbol{\phi}}$. If the function family of the critic used to approximate the divergence (via the bound or class probability estimation) contains only smooth functions, it will not be able to model the sharp true density ratio, which jumps from 0 to ∞, but it can provide a smooth approximation. We show an example in Figure 26.4b, where we show the density ratio for two distributions without overlapping support and an approximation provided by an MLP trained to approximate the KL divergence using Equation 26.25. Here, the smooth decision surface provided by the MLP can be used to train a generative model while the underlying KL divergence cannot be; the learned MLP provides the gradient signal on how to move distribution mass to areas with more density under the data distribution, while the KL divergence provides a zero gradient almost everywhere in the space. This ability of approximations to f-divergences to overcome non-overlapping support issues is a desirable property of generative modeling training criteria, as it allows models to learn the data distribution regardless of initialization [Fed+18]. Thus while the case of non-overlapping support provides an important theoretical difference between IPMs and f-divergences, it is less significant in practice since bounds on f-divergences or class probability estimation are used with smooth critics to approximate the underlying divergence.

Some density ratio and density difference based approaches also share commonalities: bounds are used both for f-divergences (variational bounds in Equation 26.25) and for the Wasserstein distance (Equation (26.31)). These bounds to distributional divergence and distances have their own set of challenges: since the generator minimizes a lower bound of the underlying divergence or distance, minimizing this objective provides no guarantees that the divergence will decrease in training. To see this, we can look at Equation 26.26: its RHS can get arbitrarily low without decreasing the LHS, the divergence we are interested in minimizing; this is unlike variational *upper* bound on the KL divergence used to train variational autoencoders Chapter 21.

26.3 Generative adversarial networks

We have looked at different learning principles that do not require the use of explicit likelihoods, and thus can be used to train implicit models. These learning principles specify training criteria, but do not tell us how to *train* models or parameterize 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 26.3.1); how to train models using gradient descent (Section 26.3.2); how to improve GAN optimization (Section 26.3.4) and how to assess GAN convergence (Section 26.3.5).

26.3.1 From learning principles to loss functions

In Section 26.2 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 $\boldsymbol{\theta}$ and the critic parameters $\boldsymbol{\phi}$. Many of these objectives 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:

$$\min \max V(\boldsymbol{\phi}, \boldsymbol{\theta}) \tag{26.41}$$

As an example, we recover the original GAN with the Bernoulli log-loss (Equation (26.19)) when

$$V(\boldsymbol{\phi}, \boldsymbol{\theta}) = \frac{1}{2}\mathbb{E}_{p^*(\boldsymbol{x})}[\log D_{\boldsymbol{\phi}}(\boldsymbol{x})] + \frac{1}{2}\mathbb{E}_{q_\theta(\boldsymbol{x})}[\log(1 - D_{\boldsymbol{\phi}}(\boldsymbol{x}))]. \tag{26.42}$$

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 26.42), 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: $\min_{\boldsymbol{\theta}} \mathbb{E}_{q_\theta(\boldsymbol{x})} \log(1 - D_{\boldsymbol{\phi}}(\boldsymbol{x}))$. 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 "nonsaturating loss" [Goo+14], defined as $\mathbb{E}_{q_\theta(\boldsymbol{x})} - \log D_{\boldsymbol{\phi}}(\boldsymbol{x})$, enjoys better gradient properties early in training, as shown in Figure 26.5: 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., $D(G(\boldsymbol{z}))$ 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:

$$\min_{\boldsymbol{\phi}} L_D(\boldsymbol{\phi}, \boldsymbol{\theta}); \qquad \min_{\boldsymbol{\theta}} L_G(\boldsymbol{\phi}, \boldsymbol{\theta}). \tag{26.43}$$

We recover the zero-sum formulations if $-L_D(\boldsymbol{\phi}, \boldsymbol{\theta}) = L_G(\boldsymbol{\phi}, \boldsymbol{\theta}) = V(\boldsymbol{\phi}, \boldsymbol{\theta})$. Despite departing from the zero-sum structure, the nested form of the optimization remains in the general formulation, as we will discuss in Section 26.3.2.

The loss functions for the discriminator and generator, L_D and L_G respectively, follow the general form in Equation 26.5, 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:

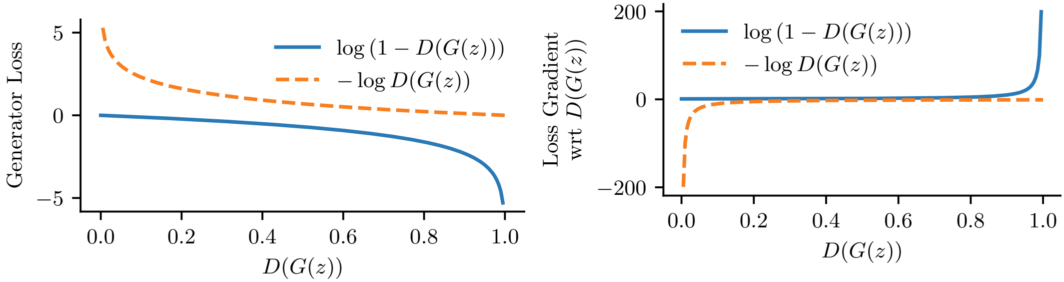
$$L_D(\boldsymbol{\phi}, \boldsymbol{\theta}) = \mathbb{E}_{p^*(\boldsymbol{x})} g(D_{\boldsymbol{\phi}}(\boldsymbol{x})) + \mathbb{E}_{q_\theta(\boldsymbol{x})} h(D_{\boldsymbol{\phi}}(\boldsymbol{x})) = \mathbb{E}_{p^*(\boldsymbol{x})} g(D_{\boldsymbol{\phi}}(\boldsymbol{x})) + \mathbb{E}_{q(\boldsymbol{z})} h(D_{\boldsymbol{\phi}}(G_{\boldsymbol{\theta}}(\boldsymbol{z}))) \tag{26.44}$$

$$L_G(\boldsymbol{\phi}, \boldsymbol{\theta}) = \mathbb{E}_{q_\theta(\boldsymbol{x})} l(D_{\boldsymbol{\phi}}(\boldsymbol{x})) = \mathbb{E}_{q(\boldsymbol{z})} l(D_{\boldsymbol{\phi}}(G_{\boldsymbol{\theta}}(\boldsymbol{z})) \tag{26.45}$$

where g, h, $l : \mathbb{R} \to \mathbb{R}$. 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 $l(t) = f^\dagger(t)$.

26.3.2 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 $L_D(\boldsymbol{\phi}, \boldsymbol{\theta})$ and a generator loss function $L_G(\boldsymbol{\phi}, \boldsymbol{\theta})$. Since the discriminator is often introduced to approximate a distance or divergence $D(p^*, q_\theta)$ (Section 26.2), 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 $\boldsymbol{\phi}^* = \operatorname{argmin}_{\boldsymbol{\phi}} L_D(\boldsymbol{\phi}, \boldsymbol{\theta})$ in order to perform a gradient update given by $\nabla_{\boldsymbol{\theta}} L_G(\boldsymbol{\phi}^*, \boldsymbol{\theta})$. Fully solving the inner optimization problem $\boldsymbol{\phi}^* = \operatorname{argmin}_{\boldsymbol{\phi}} L_D(\boldsymbol{\phi}, \boldsymbol{\theta})$



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

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

Figure 26.5: 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*

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 26.1.

Algorithm 26.1: General GAN training algorithm with alternating updates

Initialize $\boldsymbol{\phi}$, $\boldsymbol{\theta}$
for *each training iteration* **do**
  for K *steps* **do**
    Update the discriminator parameters $\boldsymbol{\phi}$ using the gradient $\nabla_{\boldsymbol{\phi}} L_D(\boldsymbol{\phi}, \boldsymbol{\theta})$;
  Update the generator parameters $\boldsymbol{\theta}$ using the gradient $\nabla_{\boldsymbol{\theta}} L_G(\boldsymbol{\phi}, \boldsymbol{\theta})$
Return $\boldsymbol{\phi}$, $\boldsymbol{\theta}$

We are thus interested in computing $\nabla_{\boldsymbol{\phi}} L_D(\boldsymbol{\phi}, \boldsymbol{\theta})$ and $\nabla_{\boldsymbol{\theta}} L_G(\boldsymbol{\phi}, \boldsymbol{\theta})$. Given the choice of loss functions follows the general form in Equations 26.44 and 26.45 both for the discriminator and generator, we can compute the gradients that can be used for training. To compute the discriminator gradients, we write:

$$\nabla_{\boldsymbol{\phi}} L_D(\boldsymbol{\phi}, \boldsymbol{\theta}) = \nabla_{\boldsymbol{\phi}} \left[\mathbb{E}_{p^*(\boldsymbol{x})} g(D_{\boldsymbol{\phi}}(\boldsymbol{x})) + \mathbb{E}_{q_\theta(\boldsymbol{x})} h(D_{\boldsymbol{\phi}}(\boldsymbol{x})) \right] \tag{26.46}$$

$$= \mathbb{E}_{p^*(\boldsymbol{x})} \nabla_{\boldsymbol{\phi}} g(D_{\boldsymbol{\phi}}(\boldsymbol{x})) + \mathbb{E}_{q_\theta(\boldsymbol{x})} \nabla_{\boldsymbol{\phi}} h(D_{\boldsymbol{\phi}}(\boldsymbol{x})) \tag{26.47}$$

where $\nabla_{\boldsymbol{\phi}} g(D_{\boldsymbol{\phi}}(\boldsymbol{x}))$ and $\nabla_{\boldsymbol{\phi}} h(D_{\boldsymbol{\phi}}(\boldsymbol{x}))$ 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:

$$L_G(\boldsymbol{\phi}, \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{q_\theta(\boldsymbol{x})} l(D_{\boldsymbol{\phi}}(\boldsymbol{x})) \tag{26.48}$$

Here we cannot change the order of differentiation and integration since the distribution under the integral depends on the differentiation parameter $\boldsymbol{\theta}$. Instead, we will use that $q_\theta(\boldsymbol{x})$ is the distribution induced by an implicit generative model (also known as the "reparameterization trick", see Section 6.3.5):

$$\nabla_{\boldsymbol{\theta}} L_G(\boldsymbol{\phi}, \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{q_\theta(\boldsymbol{x})} l(D_{\boldsymbol{\phi}}(\boldsymbol{x})) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{q(\boldsymbol{z})} l(D_{\boldsymbol{\phi}}(G_{\boldsymbol{\theta}}(\boldsymbol{z}))) = \mathbb{E}_{q(\boldsymbol{z})} \nabla_{\boldsymbol{\theta}} l(D_{\boldsymbol{\phi}}(G_{\boldsymbol{\theta}}(\boldsymbol{z}))) \tag{26.49}$$

and again use Monte Carlo estimation to approximate the gradient using samples from the prior $q(\boldsymbol{z})$. Replacing the choice of loss functions and Monte Carlo estimation in Algorithm 26.1 leads to Algorithm 26.2, which is often used to train GANs.

Algorithm 26.2: GAN training algorithm

```
Initialize φ, θ
for each training iteration do
    for K steps do
        Sample minibatch of M noise vectors z_m ~ q(z)
        Sample minibatch of M examples x_m ~ p*(x)
        Update the discriminator by performing stochastic gradient descent using this gradient:
          ∇_φ (1/M) Σ_{m=1}^M [g(D_φ(x_m)) + ∇_φ h(D_φ(G_θ(z_m)))].
    Sample minibatch of M noise vectors z_m ~ q(z)
    Update the generator by performing stochastic gradient descent using this gradient:
      ∇_θ (1/M) Σ_{m=1}^M l(D_φ(G_θ(z_m))).
Return φ, θ
```

26.3.3 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 phenomenon where the generator converges to a distribution which does not cover not all the modes (peaks) of the data distribution, thus the model underfits the distribution. We show an example in Figure 26.6: while the data is a mixture of Gaussians with 16 modes, the model converges only to a few modes. Alternatively, another problematic behavior is **mode hopping**, where the generator "hops" between generating different modes of the data distribution. An intuitive explanation for this behavior is as follows: if the generator becomes good at generating data from one mode, it will generate more from that mode. If the discriminator cannot learn to distinguish between real and generated data in this mode, the generator has no incentive to expand its support and generate data from other modes. On the other hand, if the discriminator eventually learns to distinguish between the real and generated data inside this mode, the generator can simply move (hop) to a new mode, and this game of cat and mouse can continue.

While mode collapse and mode hopping are often associated with GANs, many improvements have made GAN training more stable, and these behaviors more rare. These improvements include using large batch sizes, increasing the discriminator neural capacity, using discriminator and generator regularization, as well as more complex optimization methods.

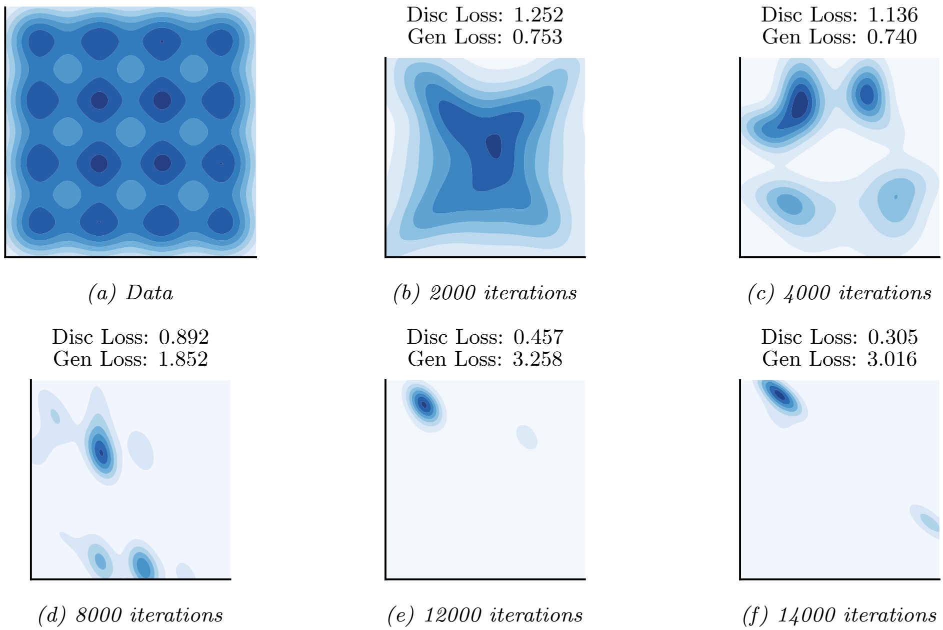


Figure 26.6: Illustration of mode collapse and mode hopping in GAN training. (a) The dataset, a mixture of 16 Gaussians in 2 dimensions. (b-f) Samples from the model after various amounts of training. Generated by gan_mixture_of_gaussians.ipynb.

26.3.4 Improving GAN optimization

Hyperparameter choices such as the choice of momentum can be crucial when training GANs, with lower momentum values being preferred compared to the usual high momentum used in supervised learning. Algorithms such as Adam [KB14a] provide a great boost in performance [RMC16a]. Many other optimization methods have been successfully applied to GANs, such as those which target variance reduction [Cha+19c]; those which backpropagate through gradient steps, thus ensuring that generator does well against the discriminator *after it has been updated* [Met+16]; or using a local bilinear approximation of the two player game [SA19]. While promising, these advanced optimization methods tend to have a higher computational cost, making them harder to scale to large models or large datasets compared to less efficient optimization methods.

26.3.5 Convergence of GAN training

The challenges with GAN optimization make it hard to quantify when convergence has occurred. In Section 26.2 we saw how global convergence guarantees can be provided under optimality conditions for multiple objectives constructed starting with different distributional divergences and distances: if the discriminator is optimal, the generator is minimizing a distributional divergence or distance between the data and model distribution, and thus under infinite capacity and perfect optimization can learn the data distribution. This type of argument has been used since the original GAN paper [Goo+14] to connect GANs to standard objectives in generative models, and obtain the associated theoretical guarantees. From a game theory perspective, this type of convergence guarantee provides an existence proof of a global Nash equilibrium for the GAN game, though under strong assumptions. A Nash equilibrium is achieved when both players (the discriminator and generator) would incur a loss if they decide to act by changing their parameters. Consider the original GAN defined by the objective in Equation 26.19; then $q_{\boldsymbol{\theta}} = p^*$ and $D_{\boldsymbol{\phi}}(\boldsymbol{x}) = \frac{p^*(\boldsymbol{x})}{p^*(\boldsymbol{x})+q_\theta(\boldsymbol{x})} = \frac{1}{2}$ is a global Nash equilibrium, since for a given $q_{\boldsymbol{\theta}}$, the ratio $\frac{p^*(\boldsymbol{x})}{p^*(\boldsymbol{x})+q_\theta(\boldsymbol{x})}$ is the optimal discriminator (Equation 26.11), and given an optimal discriminator, the data distribution is the optimal generator as it is the minimizer of the Jensen-Shannon divergence (Equation 26.15).

While these global theoretical guarantees provide useful insights about the GAN game, they do not account for optimization challenges that arise with accounting for the optimization trajectories of the two players, or for neural network parameterization since they assume infinite capacity both for the discriminator and generator. In practice GANs do not decrease a distance or divergence at every optimization step [Fed+18] and global guarantees are difficult to obtain when using optimization methods such as gradient descent. Instead, the focus shifts towards local convergence guarantees, such as reaching a local Nash equilibrium. A local Nash equilibrium requires that both players are at a local, not global minimum: a local Nash equilibrium is a stationary point (the gradients of the two loss functions are zero, i.e $\nabla_{\boldsymbol{\phi}} L_D(\boldsymbol{\phi}, \boldsymbol{\theta}) = \mathbf{0}$ and $\nabla_{\boldsymbol{\theta}} L_G(\boldsymbol{\phi}, \boldsymbol{\theta}) = \mathbf{0}$), and the eigenvalues of the Hessian of each player ($\nabla_{\boldsymbol{\phi}} \nabla_{\boldsymbol{\phi}} L_D(\boldsymbol{\phi}, \boldsymbol{\theta})$ and $\nabla_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} L_G(\boldsymbol{\phi}, \boldsymbol{\theta})$) are non-negative; for a longer discussion on Nash equilibria in continuous games, see [RBS16]. For the general GAN game, it is not guaranteed that a local Nash equilibrium always exists [FO20], and weaker conditions such as stationarity or locally stable stationarity have been studied [Ber+19]; other equilibrium definitions inspired by game theory have also been used [JNJ20; HLC19].

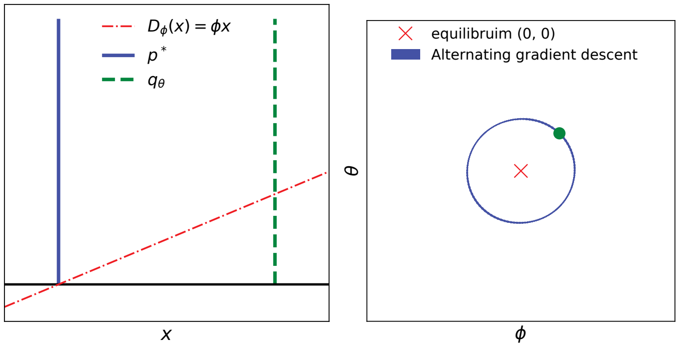
To motivate why convergence analysis is important in the case of GANs, we visualize an example of a GAN that does not converge trained with gradient descent. In DiracGAN [MGN18a] the data distribution $p^*(\boldsymbol{x})$ is the Dirac delta distribution with mass at zero. The generator is modeling a Dirac delta distribution with parameter θ: $G_\theta(z) = \theta$ and the discriminator is a linear function of the input with learned parameter ϕ: $D_\phi(x) = \phi x$. We also assume a GAN formulation where $g = h = -l$ in the general loss functions L_D and L_G defined above, see Equations (26.44) and (26.45). This results in the zero-sum game given by:

$$L_D = \mathbb{E}_{p^*(x)} - l(D_{\boldsymbol{\phi}}(x)) + \mathbb{E}_{q_\theta(x)} - l(D_{\boldsymbol{\phi}}(x)) = -l(0) - l(\theta\phi) \tag{26.50}$$

$$L_G = \mathbb{E}_{p^*(x)} l(D_{\boldsymbol{\phi}}(x)) + \mathbb{E}_{q_\theta(x)} l(D_{\boldsymbol{\phi}}(x)) = +l(0) + l(\theta\phi) \tag{26.51}$$

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 26.7 and show that DiracGANs with alternating gradient descent (Algorithm 26.1) 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



(a) *The DiracGAN problem.* (b) *Alternating gradient descent on DiracGAN.*

Figure 26.7: Visualizing divergence using a simple GAN: DiracGAN. Generated by dirac_gan.ipynb

with learning rates αh and λh, either via alternating updates (as we have seen in Algorithm 26.1):

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}_{t-1} - \alpha h \nabla_{\boldsymbol{\phi}} L_D(\boldsymbol{\phi}_{t-1}, \boldsymbol{\theta}_{t-1}), \tag{26.52}$$
$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \lambda h \nabla_{\boldsymbol{\theta}} L_G(\boldsymbol{\phi}_t, \boldsymbol{\theta}_{t-1}) \tag{26.53}$$

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

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}_{t-1} - \alpha h \nabla_{\boldsymbol{\phi}} L_D(\boldsymbol{\phi}_{t-1}, \boldsymbol{\theta}_{t-1}), \tag{26.54}$$
$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \lambda h \nabla_{\boldsymbol{\theta}} L_G(\boldsymbol{\phi}_{t-1}, \boldsymbol{\theta}_{t-1}) \tag{26.55}$$

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

$$\dot{\boldsymbol{\phi}} = -\nabla_{\boldsymbol{\phi}} L_D(\boldsymbol{\phi}, \boldsymbol{\theta}), \tag{26.56}$$
$$\dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}} L_G(\boldsymbol{\phi}, \boldsymbol{\theta}) \tag{26.57}$$

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

$$J = \begin{bmatrix} -\nabla_{\boldsymbol{\phi}} \nabla_{\boldsymbol{\phi}} L_D(\boldsymbol{\phi}, \boldsymbol{\theta}) & -\nabla_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\phi}} L_D(\boldsymbol{\phi}, \boldsymbol{\theta}) \\ -\nabla_{\boldsymbol{\phi}} \nabla_{\boldsymbol{\theta}} L_G(\boldsymbol{\phi}, \boldsymbol{\theta}) & -\nabla_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} L_G(\boldsymbol{\phi}, \boldsymbol{\theta}) \end{bmatrix} \tag{26.58}$$

evaluated at a stationary point (i.e., where $\nabla_{\boldsymbol{\phi}} L_D(\boldsymbol{\phi}, \boldsymbol{\theta}) = 0$ and $\nabla_{\boldsymbol{\theta}} L_G(\boldsymbol{\phi}, \boldsymbol{\theta}) = 0$). 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

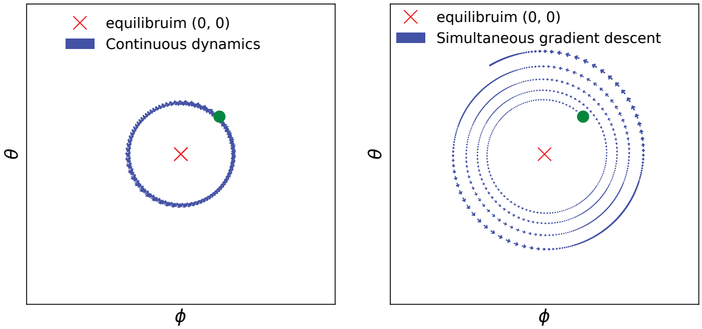


Figure 26.8: Continuous (left) and discrete dynamics (right) take different trajectories in DiracGAN. Generated by dirac_gan.ipynb

equilibrium. For the DiracGAN, the Jacobian evaluated at the equilibrium $\theta = \phi = 0$ is:

$$J = \begin{bmatrix} \nabla_{\boldsymbol{\phi}} \nabla_{\boldsymbol{\phi}} l(\theta\phi) + l(0) & \nabla_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\phi}} l(\theta\phi) + l(0) \\ -\nabla_{\boldsymbol{\phi}} \nabla_{\boldsymbol{\theta}} \left(l(\theta\phi) + l(0)\right) & -\nabla_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \left(l(\theta\phi) + l(0)\right) \end{bmatrix} = \begin{bmatrix} 0 & l'(0) \\ -l'(0) & 0 \end{bmatrix} \tag{26.59}$$

where eigenvalues of this Jacobian are $\lambda_{\pm} = \pm i l'(0)$. 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 26.8, left):

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

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 26.54 and 26.55. Under certain conditions, [MNG17b] prove that GANs trained with simultaneous gradient descent reach a local Nash equilibrium. Their approach relies on assessing the convergence of series of the form $F^k(\boldsymbol{x})$ resulting from the repeated application of gradient descent update of the form $F(\boldsymbol{x}) = \boldsymbol{x} + hG(\boldsymbol{x})$, 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 26.8, 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 26.56 and 26.57

and the gradient descent dynamics in Equations 26.54 and 26.55 — 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 parameterization 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.

26.4 Conditional GANs

We have thus far discussed how to use implicit generative models to learn a true unconditional distribution $p^*(\boldsymbol{x})$ from which we only have samples. It is often useful, however, to be able to learn *conditional distributions* of the from $p^*(\boldsymbol{x}|\boldsymbol{y})$. This requires having **paired data**, where each input $\boldsymbol{x}_n$ is paired with a corresponding set of covariates $\boldsymbol{y}_n$, such as a class label, or a set of attributes or words, so $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n) : n = 1 : N\}$, as in standard supervised learning. The conditioning variable can be discrete, like a class label, or continuous, such as an embedding which encodes other information. 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, as we discuss in Section 26.7.

To be able to learn implicit conditional distributions $q_\theta(\boldsymbol{x}|\boldsymbol{y})$, we require datasets that specify the conditioning information associated with data, and we have to adapt the model architectures and loss functions. 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]:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \frac{1}{2}\mathbb{E}_{p(\boldsymbol{y})}\mathbb{E}_{p^*(\boldsymbol{x}|\boldsymbol{y})}[\log D_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y})] + \frac{1}{2}\mathbb{E}_{p(\boldsymbol{y})}\mathbb{E}_{q_\theta(\boldsymbol{x}|\boldsymbol{y})}[\log(1 - D_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y}))] \tag{26.60}$$

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

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \frac{1}{2}\mathbb{E}_{p(\boldsymbol{y})}\mathbb{E}_{p^*(\boldsymbol{x}|\boldsymbol{y})}[\log D_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y})] + \frac{1}{2}\mathbb{E}_{p(\boldsymbol{y})}\mathbb{E}_{q(\boldsymbol{z})}[\log(1 - D_{\boldsymbol{\phi}}(\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{y}), \boldsymbol{y}))] \tag{26.61}$$

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

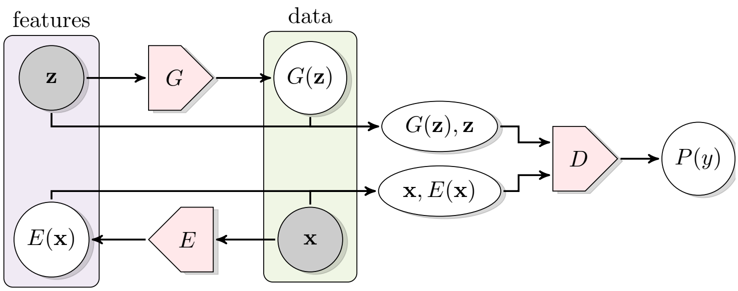


Figure 26.9: Learning an implicit posterior using an adversarial approach, as done in BiGAN. From Figure 1 of [DKD16]. Used with kind permission of Jeff Donahue.

dataset [OOS17]:

$$\mathcal{L}_c(\boldsymbol{\theta}, \boldsymbol{\phi}) = -\left[\frac{1}{2}\mathbb{E}_{p(\boldsymbol{y})}\mathbb{E}_{p^*(\boldsymbol{x}|\boldsymbol{y})}[\log D_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{x})] + \frac{1}{2}\mathbb{E}_{p(\boldsymbol{y})}\mathbb{E}_{q_\theta(\boldsymbol{x}|\boldsymbol{y})}[\log(D_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{x}))]\right] \tag{26.62}$$

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 minmize the classification loss $\mathcal{L}_c$. Thus together with the adversarial dynamics provided by $\mathcal{L}$, the two players are trained as follows:

$$\max_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) - \mathcal{L}_c(\boldsymbol{\theta}, \boldsymbol{\phi}) \qquad\qquad \min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) + \mathcal{L}_c(\boldsymbol{\theta}, \boldsymbol{\phi}) \tag{26.63}$$

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 $\boldsymbol{z}$ 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].

26.5 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 $p(\boldsymbol{z}|\boldsymbol{x})$, 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** $p(\boldsymbol{z}|\boldsymbol{x})$ introduce changes to the GAN algorithm to do so. An example of such a method is **BiGAN** (bidirectional GAN) [DKD16] or **ALI** (adversarialy learned inference) [Dum+16], which trains an implicit parameterized encoder $E_{\boldsymbol{\zeta}}$ to map input $\boldsymbol{x}$ to latent variables $\boldsymbol{z}$. To ensure consistency between the encoder $E_{\boldsymbol{\zeta}}$ and the generator $G_{\boldsymbol{\theta}}$, an adversarial approach is introduced with a discriminator $D_{\boldsymbol{\phi}}$ learning to distinguish between pairs of data and latent samples: $D_{\boldsymbol{\phi}}$ learns to consider pairs $(\boldsymbol{x}, E_{\boldsymbol{\zeta}}(\boldsymbol{x}))$ with $\boldsymbol{x} \sim p^*$ as real, while $(G_{\boldsymbol{\theta}}(\boldsymbol{z}), \boldsymbol{z})$ with $\boldsymbol{z} \sim q(\boldsymbol{z})$ is considered fake. This approach, shown in Figure 26.9, ensures that the joint distributions are matched, and thus the marginal distribution $q_\theta(\boldsymbol{x})$ given by $G_{\boldsymbol{\theta}}$ should learn $p^*(\boldsymbol{x})$, while the conditional distribution $p_{\boldsymbol{\zeta}}(\boldsymbol{z}|\boldsymbol{x})$ given by $E_{\boldsymbol{\zeta}}$ should learn $q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) = \frac{q_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\theta}}(\boldsymbol{x})} \propto q_\theta(\boldsymbol{x}|\boldsymbol{z})q(\boldsymbol{z})$. This joint GAN loss can be used both to train the generator $G_{\boldsymbol{\theta}}$ and the encoder $E_{\boldsymbol{\zeta}}$, 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: $E_{\boldsymbol{\theta}}(G_{\boldsymbol{\zeta}}(\boldsymbol{z})) = \boldsymbol{z}$ and $G_{\boldsymbol{\zeta}}(E_{\boldsymbol{\theta}}(\boldsymbol{x})) = \boldsymbol{x}$. (See also Section 21.2.4 for a discussion of how VAEs learn to ensure $p^*(\boldsymbol{x})p_{\boldsymbol{\zeta}}(\boldsymbol{z}|\boldsymbol{x})$ matches $p(\boldsymbol{z})p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ using an explicit model of the data.)

26.6 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 parameterized 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 26.6.1), employing the right inductive biases (Section 26.6.2), incorporating attention in GAN models (Section 26.6.3), progressive generation (Section 26.6.4), regularization (Section 26.6.5), and using large scale architectures (Section 26.6.6).

26.6.1 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 parameterization of the critic. Even within the same data modality, changes to critic architectures and regularization 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.

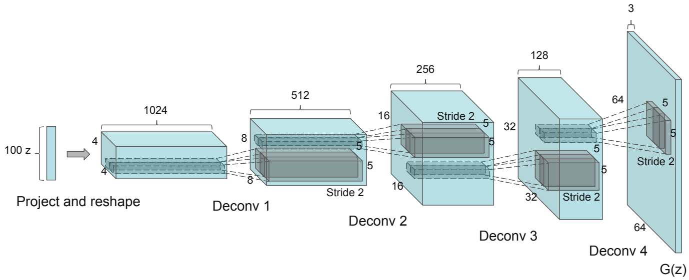


Figure 26.10: DCGAN convolutional generator. From Figure 1 of [RMC15]. Used with kind permission of Alec Radford.

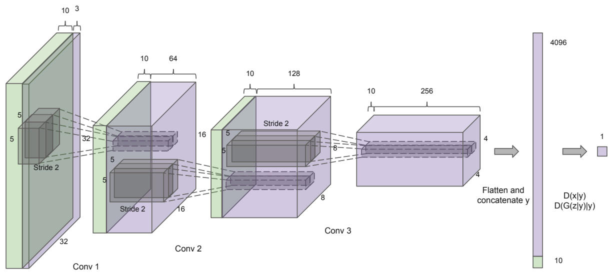


Figure 26.11: DCGAN convolutional discrimiantor. From Figure 1 of [RMC15]. Used with kind permission of Alec Radford.

26.6.2 Architectural inductive biases

While the original GAN paper used convolutions only sparingly, deep convolutional GAN (**DC-GAN**) [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 26.10) 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].

26.6.3 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+19c]. 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

Figure 26.12: 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+19c]. Used with kind permission of Han Zhang.

layers. This is illustrated in Figure 26.12, 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 $\boldsymbol{h} \in \mathbb{R}^{C \times N}$ is defined by $\boldsymbol{f} = W_f \boldsymbol{h}$, $\boldsymbol{g} = W_g \boldsymbol{h}$, $\mathbf{S} = \boldsymbol{f}^T \boldsymbol{g}$, where $W_f \in \mathbb{R}^{C' \times C}, W_g \in \mathbb{R}^{C' \times C}$, where $C' \leq C$ is a hyperparameter. From $\mathbf{S} \in \mathbb{R}^{N \times N}$, a probability row matrix β is obtained by applying the softmax operator for each row, which is then used to *attend* to a linear transformation of the features $\boldsymbol{o} = W_o(W_h \boldsymbol{h})\beta^T \in R^{C \times N}$, 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 $\boldsymbol{y} = \gamma \boldsymbol{o} + \boldsymbol{h}$, 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 γ 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.

26.6.4 Progressive generation

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

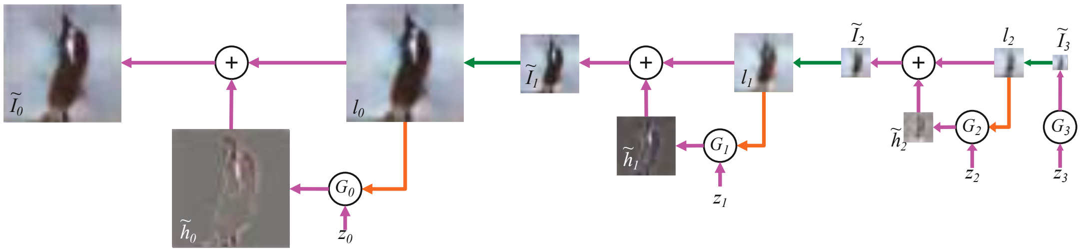


Figure 26.13: LapGAN generation algorithm: the generation process starts with a low dimension sample, which gets upscaled and residually added to the output of a generator at a higher resolution. The process gets repeated multiple times. From Figure 1 of [DCF+15]. Used with kind permission of Emily Denton.

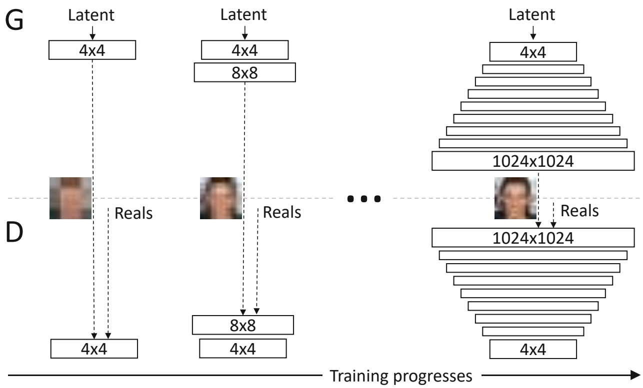


Figure 26.14: ProgressiveGAN training algorithm. The input to the discriminator at the bottom of the figure is either a generated image, or a real image (denotes as 'Reals' in the figure) at the corresponding resolution. From Figure 1 of [Kar+18]. Used with kind permission of Tero Karras.

sample, and so on — this process is shown in Figure 26.13. 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 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. Figure 26.14 shows the growing generator and discriminators in ProgressiveGAN training.

26.6.5 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+19c; BDS18]; and intuitively, as it can be used to avoid overfitting in the discriminator and generator. Regularization approaches include adding noise to the 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+19c; BDS18], and gradient penalties (penalizing the norm of the discriminator gradient with respect to its input $\|\nabla_{\boldsymbol{x}} D_{\boldsymbol{\phi}}(\boldsymbol{x})\|^2$) [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 the stability 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].

26.6.6 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 parameterization of the models, and the optimization method — have to be viewed collectively rather than in isolation.

26.7 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, such as images (Section 26.7.1), video (Section 26.7.2), audio (Section 26.7.3), and text (Section 26.7.4), and include applications such as imitation learning (Section 26.7.5), domain adapation (Section 26.7.6), and art (Section 26.7.7).

26.7.1 GANs for image generation

The most widely studied application area is in image generation. We focus on the translation of one image to another using either paired or unpaired datasets. 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. A JAX notebook which uses a small pretrained GAN to generate some face images can be found at GAN_JAX_CelebA_demo.ipynb. We show the progression of quality in sample generation of faces using GANs in Figure 26.15. 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].

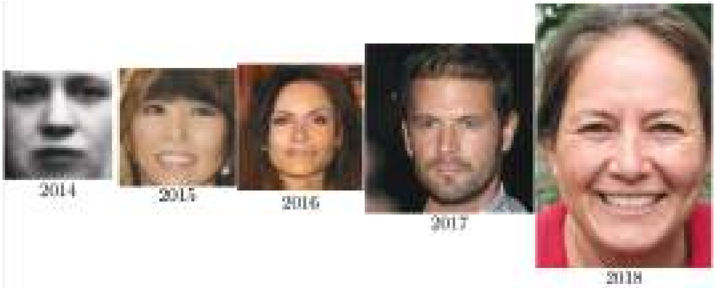


Figure 26.15: Increasingly realistic synthetic faces generated by different kinds of GAN, specifically (from left to right): original GAN [Goo+14], DCGAN [RMC15], CoupledGAN [LT16], ProgressiveGAN [Kar+18], StyleGAN [KLA19]. Used with kind permission of Ian Goodfellow. An online demo, which randomly generates face images using StyleGAN, can be found at https://thispersondoesnotexist.com.

26.7.1.1 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 and cats to 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 26.6.4. By learning the conditioning vector they are able to generate samples which interpolate 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.

26.7.1.2 Paired image-to-image generation

We have discussed in Section 26.4 how using paired data of the form $(\boldsymbol{x}_n, \boldsymbol{y}_n)$ can be used to build conditional generative models of $p(\boldsymbol{x}|\boldsymbol{y})$. In some cases, the conditioning variable $\boldsymbol{y}$ has the same size and shape as the output variable $\boldsymbol{x}$. The resulting model $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y})$ can then be used to perform **image to image translation**, as illustrated in Figure 26.16, where $\boldsymbol{y}$ is drawn from the **source domain**, and $\boldsymbol{x}$ 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 26.16

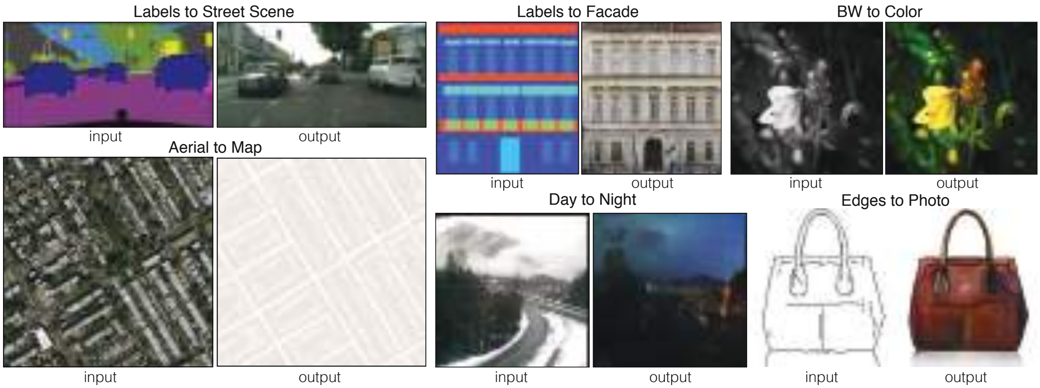


Figure 26.16: Example results on several image-to-image translation problems as generated by the pix2pix conditional GAN. From Figure 1 of [Iso+17]. Used with kind permission of Phillip Isola.

(depending on the dataset).

26.7.1.3 Unpaired image-to-image generation

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 $\mathcal{D}_x = \{\boldsymbol{x}_n : n = 1 : N_x\}$ and $\mathcal{D}_y = \{\boldsymbol{y}_n : n = 1 : N_y\}$. 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).

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

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 $p(\boldsymbol{x})$ and $p(\boldsymbol{y})$. We can try, however, to learn 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., $F(G(\boldsymbol{x})) = \boldsymbol{x}$ and $G(F(\boldsymbol{y})) = \boldsymbol{y}$. 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:

$$\mathcal{L}_{\text{cycle}} = \mathbb{E}_{p(\boldsymbol{x})}||F(G(\boldsymbol{x})) - \boldsymbol{x}||_1 + \mathbb{E}_{p(\boldsymbol{y})}||G(F(\boldsymbol{y})) - \boldsymbol{y}||_1 \tag{26.64}$$

To ensure that the outputs of G are samples from $p(\boldsymbol{y})$ and those of F are samples from $p(\boldsymbol{x})$, 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 26.17. Finally, we can optionally check that applying the conditional generator to images from its own domain does not change them:

$$\mathcal{L}_{\text{identity}} = \mathbb{E}_{p(\boldsymbol{x})}\|\boldsymbol{x} - F(\boldsymbol{x})\|_1 + \mathbb{E}_{p(\boldsymbol{y})}\|\boldsymbol{y} - G(\boldsymbol{y})\|_1 \tag{26.65}$$

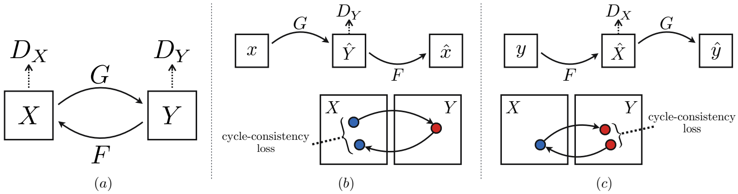


Figure 26.17: Illustration of the CycleGAN training scheme. (a) Illustration of the 4 functions that are trained. (b) Forwards cycle consistency from $\mathcal{X}$ back to $\mathcal{X}$. (c) Backwards cycle consistency from $\mathcal{Y}$ back to $\mathcal{Y}$. From Figure 3 of [Zhu+17]. Used with kind permission of Jun-Yan Zhu.

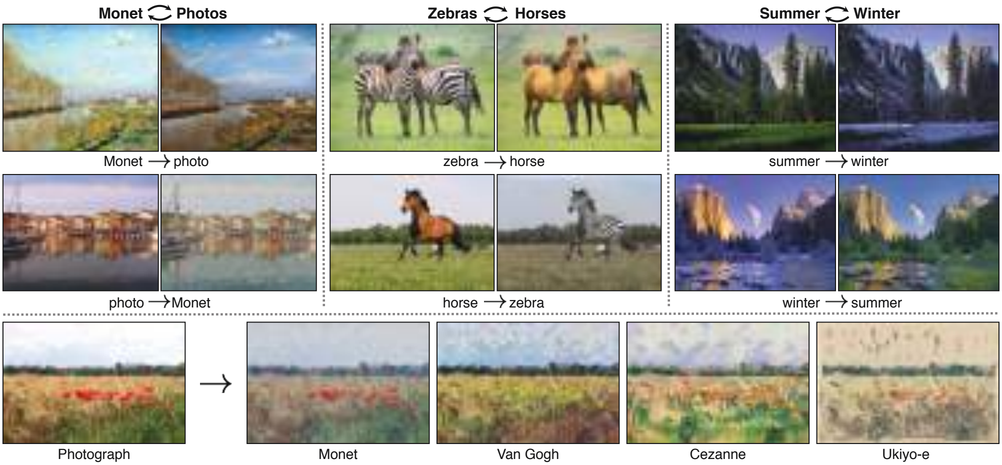


Figure 26.18: Some examples of unpaired image-to-image translation generated by the CycleGAN model. From Figure 1 of [Zhu+17]. Used with kind permission of Jun-Yan Zhu.

We can combine all three of these consistency losses to train the translation mappings F and G, using hyperparameters λ_1 and λ_2:

$$\mathcal{L} = \mathcal{L}_{\text{GAN}} + \lambda_1 \mathcal{L}_{\text{cycle}} + \lambda_2 \mathcal{L}_{\text{identity}} \tag{26.66}$$

CycleGAN results on various datasets are shown in Figure 26.18. The bottom row shows how CycleGAN can be used for **style transfer**.

26.7.2 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].

26.7.3 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+16a] 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 26.7.1.2), but uses 1d convolution instead of 2d. More recent GANs such as GAN-TTS [Bińi+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.

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

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

The difficulty of generating discrete data such as text using GANs can be seen looking at their loss function, such as in Equations (26.19), (26.21) and (26.28). GAN losses contain terms of the form $\mathbb{E}_{q_\theta(\boldsymbol{x})} f(\boldsymbol{x})$, which we not only need to evaluate, but also backpropagate through, by computing $\nabla_\theta \mathbb{E}_{q_\theta(\boldsymbol{x})} f(\boldsymbol{x})$. In the case of implicit distributions given by latent variable models, we used the reparameterization trick to compute this gradient (Equation 26.49). In the discrete case, the reparameterization trick is not available and we have to look for other ways to estimate the desired gradient. One approach is to use the score function estimator, discussed in Section 6.3.4. However, the score function estimator exhibits high gradient variance, which can destabilize training. One common approach to avoid this issue is to pre-train the language model generator using maximum likelihood, and then to fine-tune with a GAN loss which gets backpropagated into the generator using the score-function estimator, as done by Sequence GAN [Yu+17], MaliGAN [Che+17], and RankGAN [Lin+17a]. While these methods spearheaded the use of GANs for text, they do not address the inherent instabilities of score function estimation and thus have to limit the amount of adversarial fine tuning to a small number of epochs and often use a small learning rate, keeping their performance close to that of the maximum-likelihood solution [SSG18a; Cac+18].

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

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

26.7.5 Imitation learning

Imitation learning takes advantage of observations of expert demonstrations to learn action policies and reward functions of unknown environments by minimizing some form of discrepancy between the learned and the expert behaviors. There are many approaches available, including behavioral cloning [PPG91] that treats this problem as one of supervised learning, and inverse reinforcement learning [NR00b]. GANs are appealing for imitation learning since they provide a way to avoid the difficulty of designing good discrepancy functions for behaviors, and instead learn these discrepancy functions using a discriminator between trajectories generated by a learned agent and observed demonstrations.

This approach, known as generative adversarial imitation learning (**GAIL**) [HE16a] demonstrates the ability to use GANs for complex behaviors in high-dimensional environments. GAIL jointly learns a generator, which forms a stochastic policy, along with a discriminator that acts as a reward signal. Like we saw in the probabilistic development of GANs in the earlier sections, GAIL can also be generalized to multiple f-divergences, rather than the standard Jensen-Shannon divergence used as the standard loss in GANs. This has led to a family of other GAIL variants that use other f-divergences [Ke+19a; Fin+16; Bro+20c], including f-GAIL that aims to also learn the best f-divergence to use [Zha+20e], as well as new analytical insight into the computation and generalization of such approaches [Che+20b].

26.7.6 Domain adaptation

An important task in machine learning is to correct for shifts in the data distribution over time, minimizing some measure of domain shift, as we discuss in Section 19.5.3. Like with the other applications, GANs are popular as ways of avoiding the choice of distance or degree of shift. Both the supervised and unsupervised approaches for image generation we reviewed earlier looked at pixel-level domain adaptation models that perform distribution alignment in raw pixel space, translating source data to the style of a target domain, as with pix2pix and CycleGAN. Extensions of these approaches for the general problem of domain adaptation seek to do this not only in the observed data space (e.g., with pixels), but also at the feature level. One general approach is domain-adversarial training of neural networks [Gan+16b] or adversarial discriminative domain adaptation (ADDA) [Tze+17]. The CyCADA approach of [Hof+18] extends CycleGAN by enforcing both structural and semantic consistency during adaptation using a cycle-consistency loss and semantics losses based on a particular visual recognition task. There are also many extensions that include class conditional information [Tsa+18; Lon+18] or adaptation when the modes to be matched have different frequencies in the source and target domains [BHC19].

26.7.7 Design, art and creativity

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

Part V

Discovery

27 Discovery methods: an overview

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

$$\hat{\boldsymbol{z}}(\boldsymbol{x}) = \underset{\boldsymbol{z}}{\operatorname{argmax}}\, p(\boldsymbol{z}|\boldsymbol{x}) = \underset{\boldsymbol{z}}{\operatorname{argmax}} \log p(\boldsymbol{z}) + \log p(\boldsymbol{x}|\boldsymbol{z}) \tag{27.1}$$

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 27.1 for an illustration.

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

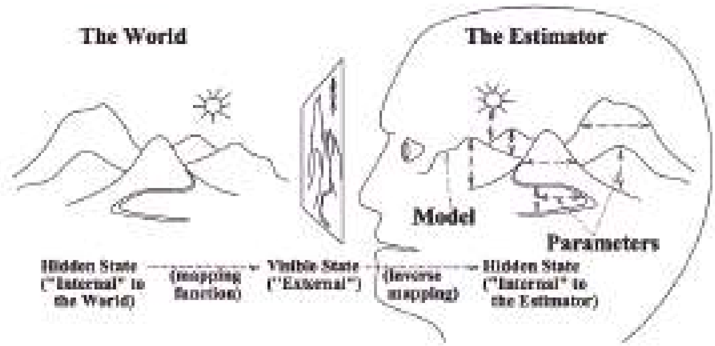


Figure 27.1: Vision as inverse graphics. The agent (here represented by a human head) has to infer the scene $\boldsymbol{z}$ given the image $\boldsymbol{x}$ using an estimator. From Figure 1 of [Rao99]. Used with kind permission of Rajesh Rao.

27.2 Overview of Part V

In Chapter 28, we discuss simple latent variable models where typically the observed data is a fixed-dimensional vector such as $\boldsymbol{x} \in \mathbb{R}^D$. In Chapter 29 we extend these models to work with sequences of correlated vectors, $\boldsymbol{x} = \boldsymbol{x}_{1:T}$, 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. In Chapter 30, we extend these models to work with general graphs.

In Chapter 31, 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 32, 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 33, we discuss how to interpret the behavior of a predictive model (typically a neural network).

28 Latent factor models

28.1 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 28.2), which has the form $p(\boldsymbol{x}) = \sum_k p(\boldsymbol{x}|z=k)p(z=k)$, where z is an indicator variable that specifies which mixture component to use for generating $\boldsymbol{x}$. 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:

$$\boldsymbol{z} \sim p(\boldsymbol{z}) \tag{28.1}$$

$$\boldsymbol{x}|\boldsymbol{z} \sim \text{Expfam}(\boldsymbol{x}|f(\boldsymbol{z})) \tag{28.2}$$

where $f(\boldsymbol{z})$ is known as the **decoder**, and $p(\boldsymbol{z})$ is some kind of prior. We assume that $\boldsymbol{z}$ is a single "layer" of hidden random variables, corresponding to a set of "latent factors". We call these **latent factor models**. In this chapter, we assume the decoder f is a simple linear model; we consider nonlinear extensions in Chapter 21. 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 $p(\boldsymbol{z})$ and/or the likelihood $p(\boldsymbol{x}|\boldsymbol{z})$, as we show in Table 28.1. 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.)

28.2 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

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p_k(\boldsymbol{x}) \tag{28.3}$$

where p_k is the k'th mixture component, and π_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** $z \in \{1, \ldots, K\}$, which specifies which distribution to use for generating the output $\boldsymbol{x}$. The

Model	$p(\boldsymbol{z})$	$p(\boldsymbol{x}\|\boldsymbol{z})$	Section
FA/PCA	$\mathcal{N}(\boldsymbol{z}\|\mathbf{0}, \mathbf{I})$	$\mathcal{N}(\boldsymbol{x}\|\mathbf{W}\boldsymbol{z}, \boldsymbol{\Psi})$	Section 28.3.1
GMM	$\sum_c \text{Cat}(c\|\boldsymbol{\pi})\mathcal{N}(\boldsymbol{z}\|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$	$\mathcal{N}(\boldsymbol{x}\|\mathbf{W}\boldsymbol{z}, \boldsymbol{\Psi})$	Section 28.2.4
MixFA	$\text{Cat}(c\|\boldsymbol{\pi})\mathcal{N}(\boldsymbol{z}\|\mathbf{0}, \mathbf{I})$	$\mathcal{N}(\boldsymbol{x}\|\mathbf{W}_c\boldsymbol{z} + \boldsymbol{\mu}_c, \boldsymbol{\Psi}_c)$	Section 28.3.3.5
NMF	$\prod_k \text{Ga}(z_k\|\alpha_k, \beta_k)$	$\prod_d \text{Poi}(x_d\| \exp(\boldsymbol{w}_d^\mathsf{T}\boldsymbol{z})))$	Section 28.4.1
Simplex FA (mPCA)	$\text{Dir}(\boldsymbol{z}\|\boldsymbol{\alpha})$	$\prod_d \text{Cat}(x_d\|\mathbf{W}_d\boldsymbol{z})$	Section 28.4.2
LDA	$\text{Dir}(\boldsymbol{z}\|\boldsymbol{\alpha})$	$\prod_d \text{Cat}(x_d\|\mathbf{W}\boldsymbol{z})$	Section 28.5
ICA	$\prod_d \text{Laplace}(z_d\|\lambda)$	$\prod_d \delta(x_d - \boldsymbol{w}_d^\mathsf{T}\boldsymbol{z})$	Section 28.6
Sparse coding	$\prod_k \text{Laplace}(z_k\|\lambda)$	$\prod_d \mathcal{N}(x_d\|\boldsymbol{w}_d^\mathsf{T}\boldsymbol{z}, \sigma^2)$	Section 28.6.5

Table 28.1: 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 $L = D$.)

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

$$p(z|\boldsymbol{\theta}) = \text{Cat}(z|\boldsymbol{\pi}) \tag{28.4}$$

$$p(\boldsymbol{x}|z = k, \boldsymbol{\theta}) = p(\boldsymbol{x}|\boldsymbol{\theta}_k) \tag{28.5}$$

The "generative story" for the data is that we first sample a specific component z, and then we generate the observations $\boldsymbol{x}$ using the parameters chosen according to the value of z. By marginalizing out z, we recover Equation (28.3):

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K} p(z = k|\boldsymbol{\theta})p(\boldsymbol{x}|z = k, \boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p(\boldsymbol{x}|\boldsymbol{\theta}_k) \tag{28.6}$$

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

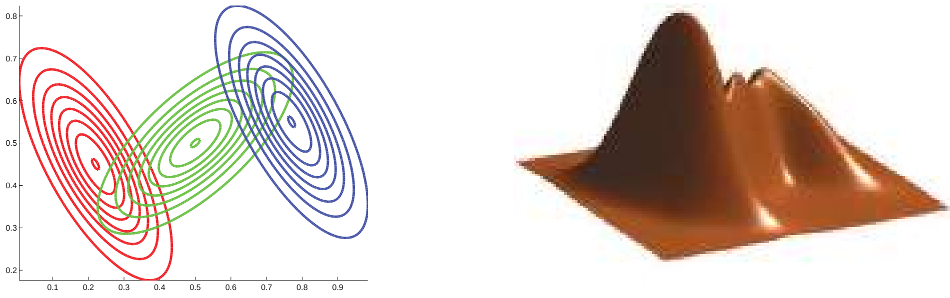
28.2.1 Gaussian mixture models (GMMs)

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

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{28.7}$$

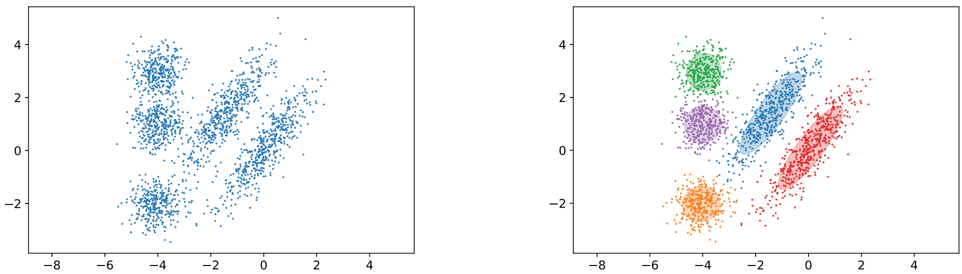
In Figure 28.1 we show the density defined by a mixture of 3 Gaussians in 2d. Each mixture component is represented by a different set of elliptical contours. If we let the number of mixture components grow sufficiently large, a GMM can approximate any smooth distribution over $\mathbb{R}^D$.

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



(a) (b)

Figure 28.1: 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.ipynb.



(a) (b)

Figure 28.2: (a) Some data in 2d. (b) A possible clustering using $K = 5$ clusters computed using a GMM. Generated by gmm_2d.ipynb.

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

$$r_{nk} \triangleq p(z_n = k|\boldsymbol{x}_n, \boldsymbol{\theta}) = \frac{p(z_n = k|\boldsymbol{\theta})p(\boldsymbol{x}_n|z_n = k, \boldsymbol{\theta})}{\sum_{k'=1}^{K} p(z_n = k'|\boldsymbol{\theta})p(\boldsymbol{x}_n|z_n = k', \boldsymbol{\theta})} \tag{28.8}$$

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

$$\hat{z}_n = \arg\max_k r_{nk} = \arg\max_k \left[\log p(\boldsymbol{x}_n|z_n = k, \boldsymbol{\theta}) + \log p(z_n = k|\boldsymbol{\theta})\right] \tag{28.9}$$

This is known as **hard clustering**. (If we use the responsibilities to fractionally assign each datapoint to different clusters, it is called **soft clustering**.) See Figure 28.2 for an example.

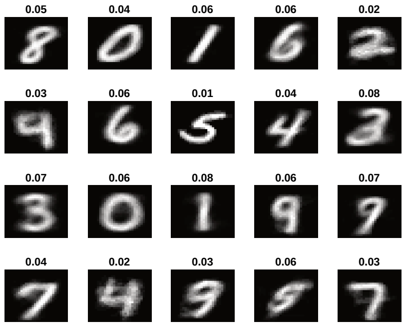


Figure 28.3: We fit a mixture of 20 Bernoullis to the binarized MNIST digit data. We visualize the estimated cluster means $\hat{\boldsymbol{\mu}}_k$. The numbers on top of each image represent the estimated mixing weights $\hat{\pi}_k$. No labels were used when training the model. Generated by mix_bernoulli_em_mnist.ipynb.

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

$$z_n = \operatorname*{argmin}_k ||\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k||_2^2 \tag{28.10}$$

In other words, we assign each datapoint 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).

28.2.2 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:

$$p(\boldsymbol{x}|z=k,\boldsymbol{\theta}) = \prod_{d=1}^{D} \text{Ber}(y_d|\mu_{dk}) = \prod_{d=1}^{D} \mu_{dk}^{y_d}(1-\mu_{dk})^{1-y_d} \tag{28.11}$$

Here μ_{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., $\boldsymbol{\mu}_k$ and π_k) are shown in Figure 28.3. 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 3.8.1 for information on how to choose the number K of mixture components.)

28.2.3 Gaussian scale mixtures (GSMs)

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

$$p(x) = \int \mathcal{N}(x|0, \sigma_0^2\epsilon^2)p(\epsilon)d\epsilon \tag{28.12}$$

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

The main advantage of this approach is that it is often computationally more convenient to work with the **expanded parameterization**, in which we explicitly include the scale term ϵ, since, conditional on that, the distribution is Gaussian. We use this formulation in Section 6.5.5, where we discuss robust regression.

28.2.3.1 Student-t distribution as a GSM

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

$$\mathcal{T}(x|0, \sigma^2, \nu) = \int_0^\infty \mathcal{N}(x|0, z\sigma^2)\mathrm{IG}(z|\frac{\nu}{2}, \frac{\nu}{2})dz = \int_0^\infty \mathcal{N}(x|0, z\sigma^2)\chi^{-2}(z|\nu, 1)dz \tag{28.13}$$

where IG is the inverse Gamma distribution (Section 2.2.3.4). Thus we can think of the Student as an infinite superposition of Gaussians of different widths; marginalizing this out induces a distribution with wider tails than a Gaussian with the same variance. This result also explains why the Student distribution approaches a Gaussian as the dof gets large, since when $\nu = \infty$, the inverse Gamma distribution becomes a delta function.

28.2.3.2 Laplace distribution as a GSM

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

$$\mathrm{Laplace}(x|0, \lambda) = \int \mathcal{N}(x|0, \tau^2)\mathrm{Ga}(\tau^2|1, \frac{\lambda^2}{2})d\tau^2 \tag{28.14}$$

28.2.3.3 Spike and slab distribution

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

$$x = \sum_{\epsilon \in \{0,1\}} \mathcal{N}(x|0, \sigma_0^2\epsilon)p(\epsilon) = \pi\mathcal{N}(x|0, \sigma_0^2) + (1-\pi)\delta_0(x) \tag{28.15}$$

This is known as the **spike and slab** distribution, since the $\delta_0(x)$ is a "spike" at 0, and the $\mathcal{N}(x|0, \sigma_0^2)$ acts like a uniform "slab" for large enough σ_0. This distribution is useful in sparse modeling.

28.2.3.4 Horseshoe distribution

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

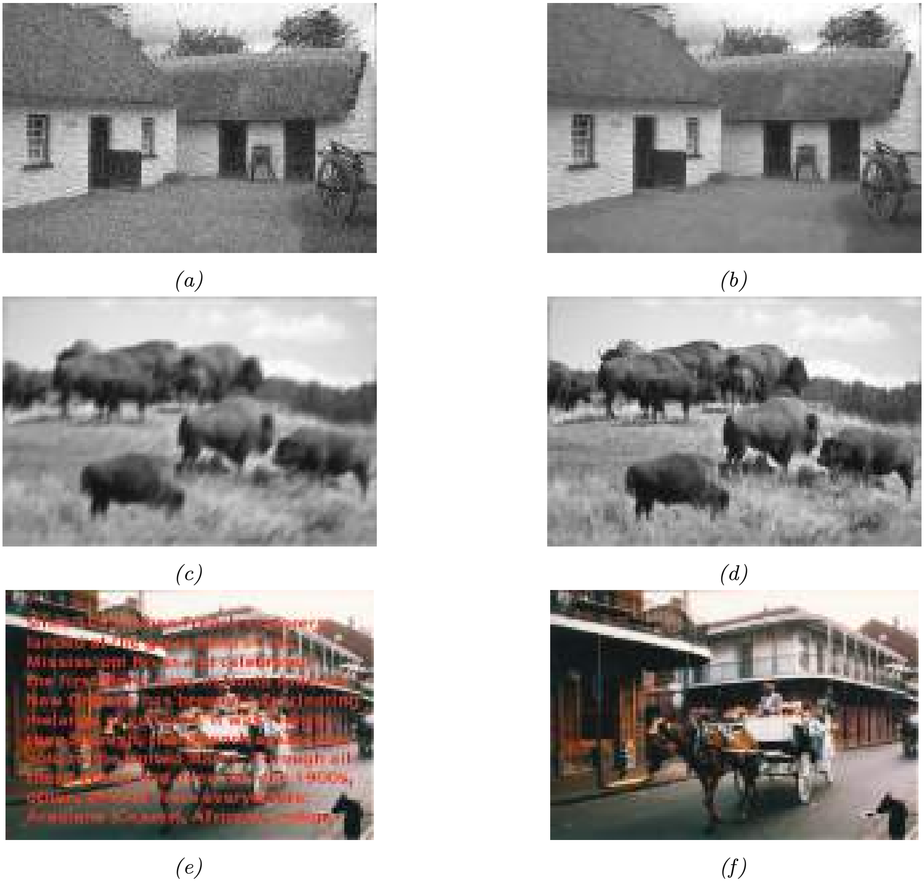


Figure 28.4: Example of recovering a clean image (right) from a corrupted version (left) using MAP estimation with a GMM patch prior and Gaussian likelihood. First row: image denoising. Second row: image deblurring. Third row: image inpainting. From [RW15] and [ZW11]. Used with kind permission of Dan 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.

28.2.4 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 $\boldsymbol{x}$ from a corrupted version $\boldsymbol{y}$. We use a linear-Gaussian forwards model of the form

$$p(\boldsymbol{y}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{y}|\mathbf{W}\boldsymbol{x}, \sigma^2\mathbf{I}) \tag{28.16}$$

where σ^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 28.4a), we can set $\mathbf{W} = \mathbf{I}$. The resulting MAP estimate can be used for **image denoising**, as in Figure 28.4b.
- If the corruption is due to blurring (as in Figure 28.4c), 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 28.4d.
- If the corruption is due to occlusion (as in Figure 28.4e), 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 28.4f.
- 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 $\hat{\boldsymbol{x}} = \operatorname{argmax} p(\boldsymbol{x}|\boldsymbol{y})$. However, the problem of inverting this model is ill-posed, since there are many possible latent images $\boldsymbol{x}$ that map to the same observed image $\boldsymbol{y}$. 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 $p(\boldsymbol{x}_i) = \sum_k p(c_i = k)\mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ 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 $\boldsymbol{x}_i$ and use Equation (2.129) to compute the marginal likelihood

$$c_i^* = \operatorname*{argmax}_c p(c)p(\boldsymbol{y}_i|c) = \operatorname*{argmax}_c p(c)\mathcal{N}(\boldsymbol{y}_i|\mathbf{W}\boldsymbol{\mu}_c, \sigma^2\mathbf{I} + \mathbf{W}\boldsymbol{\Sigma}_c\mathbf{W}^\mathsf{T}) \tag{28.17}$$

We can then approximate the MAP for the latent patch $\boldsymbol{x}_i$ by using the approximation

$$p(\boldsymbol{x}_i|\boldsymbol{y}_i) \approx p(\boldsymbol{x}_i|\boldsymbol{y}_i, c_i^*) \propto \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_{c_i^*}, \boldsymbol{\Sigma}_{c_i^*})\mathcal{N}(\boldsymbol{y}_i|\mathbf{W}\boldsymbol{x}_i, \sigma^2\mathbf{I}) \tag{28.18}$$

If we know c_i^*, we can compute the above using Bayes' rule for Gaussians in Equation (2.121).

To apply this method to full images, [ZW11] optimize the following objective

$$E(\boldsymbol{x}|\boldsymbol{y}) = \frac{1}{2\sigma^2}||\mathbf{W}\boldsymbol{x} - \boldsymbol{y}||^2 - \text{EPLL}(\boldsymbol{x}) \tag{28.19}$$

where **EPLL** is the "**expected patch log likelihood**", given by

$$\text{EPLL}(\boldsymbol{x}) = \sum_i \log p(\mathbf{P}_i\boldsymbol{x}) \tag{28.20}$$

where $\boldsymbol{x}_i = \mathbf{P}_i\boldsymbol{x}$ 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 28.4 for some examples of this process in action.

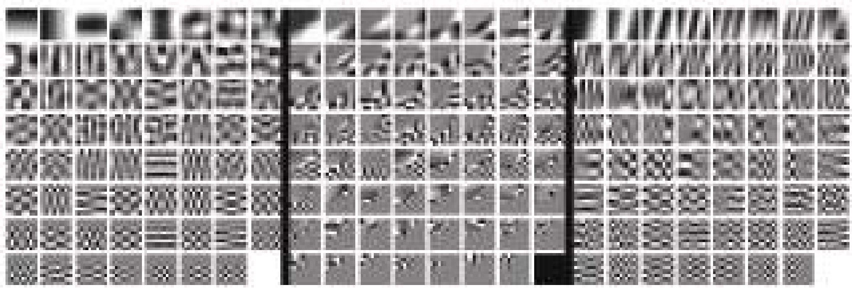


Figure 28.5: Illustration of the parameters learned by a GMM applied to image patches. Each of the 3 panels corresponds to a different mixture component k. Within each panel, we show the eigenvectors (reshaped as images) of the covariance matrix $\mathbf{\Sigma}_k$ in decreasing order of eigenvalue. We see various kinds of patterns, including ones that look like the ones learned from PCA (see Figure 28.34), but also ones that look like edges and texture. From Figure 6 of [ZW11]. Used with kind permission of Daniel Zoran.

A more principled solution to the overlapping patch problem is to use a multiscale model, as proposed in [PE16]. Another approach, proposed in [FW21], uses Gibbs sampling to combine samples from overlapping patches. This approach has the additional advantage of computing posterior samples from $p(\boldsymbol{x}|\boldsymbol{y})$, which can look much better than the posterior mean or mode computed by optimization methods. (For example, if the corruption process removes the color from the latent image $\boldsymbol{x}$ to create a gray scale $\boldsymbol{y}$, then the posterior MAP estimate of $\boldsymbol{x}$ will also be a gray scale image, whereas posterior samples will be color images.) See also Section 28.3.3.5 where we show how to extend the GMM model to a mixture of low rank Gaussians, which lets us directly model images instead of image patches.

28.2.4.1 Why does the method work?

To understand why such a simple model of image patches works so well, note that the log prior for a single latent image patch $\boldsymbol{x}_i$ using mixture component k can be written as follows:

$$\log p(\boldsymbol{x}_i|c_i = k) = \log \mathcal{N}(\boldsymbol{x}_i|\mathbf{0}, \mathbf{\Sigma}_k) = -\boldsymbol{x}_i^\mathsf{T} \mathbf{\Sigma}_k^{-1} \boldsymbol{x}_i + a_k \tag{28.21}$$

where a_k is a constant that depends on k but is independent of $\boldsymbol{x}_i$. Let $\mathbf{\Sigma}_k = \mathbf{V}_k \mathbf{\Lambda}_k \mathbf{V}_k^\mathsf{T}$ be an eigendecomposition of $\mathbf{\Sigma}_k$, where $\lambda_{k,d}$ is the d'th eigenvalue of $\mathbf{\Sigma}_k$, and $\boldsymbol{v}_{k,d}$ is the d'th eigenvector. Then we can rewrite the above as follows:

$$\log p(\boldsymbol{x}_i|c_i = k) = -\sum_{d=1}^{D} \frac{1}{\lambda_{k,d}} (\boldsymbol{v}_{k,d}^\mathsf{T} \boldsymbol{x}_i)^2 + a_k \tag{28.22}$$

Thus we see that the eigenvectors are acting like templates. Each mixture component has a different set of templates, each with their own weight (eigenvalue), as illustrated in Figure 28.5. By mixing these together, we get a powerful model for the statistics of natural image patches. (See [ZW12] for more analysis of why this simple model works so well, based on the "**dead leaves**" model of image formation.)

28.2.4.2 Speeding up inference using discriminative models

Although simple and effective, computing $f(\boldsymbol{y}) = \text{argmax}_{\boldsymbol{x}}\, p(\boldsymbol{x}|\boldsymbol{y})$ 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 $(\boldsymbol{y}, f(\boldsymbol{y}))$ pairs. We can then train an amortized inference network (Section 10.1.5) to learn this $\boldsymbol{y} \to f(\boldsymbol{y})$ 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 $(\boldsymbol{y}, \boldsymbol{x})$, where $\boldsymbol{x}$ is a clean natural image, and $\boldsymbol{y} = C(\boldsymbol{x})$ is an artificial corruption of it. We can then train a discriminative model $\hat{f}(\boldsymbol{y})$ directly from $(\boldsymbol{y}, \boldsymbol{x})$ 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 σ^2.

28.2.4.3 Blind inverse problems

In the discussion above, we assumed the forwards model had the form $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}|\mathbf{W}\boldsymbol{x}, \sigma^2\mathbf{I})$, where $\mathbf{W}$ is known. If $\mathbf{W}$ is not known, then computing $p(\boldsymbol{x}|\boldsymbol{y})$ 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, $\boldsymbol{x}$, using an EM-like method from a set of images coming from the same likelihood function. That is, we alternate between estimating $\hat{\boldsymbol{x}} = \text{argmax}_{\boldsymbol{x}}\, p(\boldsymbol{x}|\boldsymbol{y}, \hat{\mathbf{W}})$ in the E step, and estimating $\hat{\mathbf{W}} = \text{argmax}_{\mathbf{W}}\, p(\boldsymbol{y}|\hat{\boldsymbol{x}}, \mathbf{W})$ in the M step. Some encouraging results of this approach are shown in [Ani+18]. (They use a GAN prior for $p(\boldsymbol{x})$ rather than a GMM.)

In cases where we get two independent noisy samples, $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, generated from the same underlying image $\boldsymbol{x}$, then we can avoid having to explicitly learn an image prior $p(\boldsymbol{x})$, and can instead directly learn an estimator for the posterior mode, $f(\boldsymbol{y}) = \text{argmax}_{\boldsymbol{x}}\, p(\boldsymbol{x}|\boldsymbol{y})$, without needing access to the latent image $\boldsymbol{x}$, by exploiting a form of cycle consistency; see [XC19] for details.

28.2.5 Using mixture models for classification problems

It is possible to use mixture models to define the class-conditional density $p(\boldsymbol{x}|y = c)$ in a generative classifier. We can then derive the class posterior using Bayes' rule:

$$p(y = c|\boldsymbol{x}) = \frac{p(y = c)p(\boldsymbol{x}|y = c)}{\sum_{c'} p(y = c)p(\boldsymbol{x}|y = c)} = \frac{p(y = c)p(\boldsymbol{x}|y = c)}{Z} \tag{28.23}$$

where $p(y = c) = \pi_c$ is the prior on class label c, Z is the normalization constant, and the form of $p(\boldsymbol{x}|y = c)$ depends on the kind of data we have. For real-valued features, it is common to use a GMM:

$$p(\boldsymbol{x}|y = c) = \sum_{k=1}^{K_c} \alpha_{c,k} \mathcal{N}(\boldsymbol{x}|\mu_{c,k}, \boldsymbol{\Sigma}_{c,k}) \tag{28.24}$$

Using a generative model to perform classification can be useful when we have missing data, since we can compute $p(\boldsymbol{x}^v|y = c) = \sum_{\boldsymbol{x}^m} p(\boldsymbol{x}^m, \boldsymbol{x}^v|y = c)$ to compute the marginal likelihood of the visible features $\boldsymbol{x}^v$. It is also useful for semi-supervised learning, since we can optimize the model to fit $\sum_n \log p(\boldsymbol{x}_n^l, y_n^l)$ on the labeled data and $\sum_n \log p(\boldsymbol{x}_n^u)$ on the unlabeled data.

28.2.5.1 Hybrid generative/discriminative training

Unfortunately the classification accuracy of generative models of the form $p(\boldsymbol{x}, y)$ can be much worse than discriminative (conditional) models of the form $p(y|\boldsymbol{x})$, since the latter are directly optimized to predict the labels given the features, and don't "waste" capacity on modeling irrelevant details of the inputs. (For a more in-depth discussion of generative vs discriminative classifiers, see e.g., [Mur22, Sec 9.4].)

Fortunately it is possible to train generative models in a discriminative fashion, which can close the performance gap with conditional models, while maintaining the advantages of generative models. In particular, we can optimize the following hybrid objective, proposed in [BT04; Rot+18]:

$$\mathcal{L}(\boldsymbol{\theta}) = \underbrace{-\lambda \sum_{n=1}^{N} \log p(\boldsymbol{x}_n, y_n|\boldsymbol{\theta})}_{\mathcal{L}_{\text{gen}}(\boldsymbol{\theta})} \underbrace{-(1-\lambda) \sum_{n=1}^{N} \log p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta})}_{\mathcal{L}_{\text{dis}}(\boldsymbol{\theta})} \tag{28.25}$$

where $0 \leq \lambda \leq 1$ controls the tradeoff between generative and discriminative modeling.

If we have unlabeled data, we can modify the generative loss as shown below:

$$\mathcal{L}_{\text{gen}}(\boldsymbol{\theta}) = \kappa \sum_{n=1}^{N^l} \log p(\boldsymbol{x}_n^l, y_n^l|\boldsymbol{\theta}) + (1-\kappa) \sum_{n=1}^{N^u} \log p(\boldsymbol{x}_n^u|\boldsymbol{\theta}) \tag{28.26}$$

Here we have introduced an extra trade-off parameter $0 \leq \kappa \leq 1$ to prevent the unlabeled data from overwhelming the labeled data (if $N_u \gg N_l$), as proposed in [Nig+00].

An alternative to changing the objective function is to change the model itself, so that we parameterize the joint using $p(\boldsymbol{x}, y) = p(y|\boldsymbol{x}, \boldsymbol{\theta})p(\boldsymbol{x}|\tilde{\boldsymbol{\theta}})$, and then define different kinds of joint priors $p(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}})$; see [LBM06; BL07a] for details.

28.2.5.2 Optimization issues

To optimize the loss, we need to reparameterize the model into unconstrained form. For the class prior, we can use $\pi_{1:C} = \text{softmax}(\tilde{\pi}_{1:C})$, and optimize wrt the logits $\tilde{\pi}_{1:C}$. Similarly for the mixture weights $\alpha_{c,1:K}$. The means $\boldsymbol{\mu}_{ck}$ are already unconstrained. For the covariance matrices, we will use a diagonal plus low-rank representation, to reduce the number of parameters:

$$\boldsymbol{\Sigma}_{c,k} = \text{diag}(\boldsymbol{d}_{c,k}) + \mathbf{S}_{c,k}\mathbf{S}_{c,k}^{\mathsf{T}} \tag{28.27}$$

where $\mathbf{S}_{c,k}$ is an unconstrained $D \times R$ matrix, where $R \ll D$ is the rank of the approximation. (For numerical stability, we usually add $\epsilon\mathbf{I}$ to the above expression, to ensure $\boldsymbol{\Sigma}_{c,k}$ is positive definite for all parameter settings.) To ensure positivity of the diagonal term, we can use the softplus transform, $d_{c,k} = \log(1 + \exp(\tilde{d}_{c,k}))$, and optimize wrt the $\tilde{d}_{c,k}$ terms.

28.2.5.3 Numerical issues

To compute the class conditional log likelihood, $\ell_c = \log p(\boldsymbol{x}|y = c)$, we can use the **log-sum-exp trick** to avoid numerical underflow. Define $\tilde{\alpha}_{ck} = \log \alpha_{ck}$, and $\ell_{ck} = \log \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{ck}, \boldsymbol{\Sigma}_{ck})$ and let

$\beta_{ck} = \tilde{\alpha}_{ck} + \ell_{ck}$. Then we have

$$\ell_c = \log p(\boldsymbol{x}|y=c) = \log\left(\sum_k p(z_k|y=c)p(\boldsymbol{x}|y=c,z=k)\right) = \log\left(\sum_k \alpha_{ck}\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{ck},\boldsymbol{\Sigma}_{ck})\right) \tag{28.28}$$

$$= \log\left(\sum_k e^{\beta_{ck}}\right) = \log\left(e^M \sum_k e^{\beta_{ck}-M}\right) = M + \log\left(\sum_k e^{\beta'_{ck}}\right) \triangleq \text{logsumexp}(\{\beta_{ck}\}_k) \tag{28.29}$$

where $M = \max_k \beta_{ck}$ and $\beta'_{ck} = \beta_{ck} - M$. Note that we can safely compute each $e^{\beta'_{ck}}$ without underflow.

We can use a similar method to compute the posterior over classes. We have

$$p(y=c|\boldsymbol{x}) = \frac{\pi_c e^{l_c}}{Z} = \frac{\pi_c e^{l_c - L}}{e^{-L}Z} = \frac{\pi_c e^{\tilde{l}_c}}{\tilde{Z}} \tag{28.30}$$

where $L = \max_c l_c$, $\tilde{l}_c = l_c - L$, and $\tilde{Z} = \sum_c \pi_c e^{\tilde{l}_c}$. This lets us combine the class prior probability π_c with the scaled class conditional log likelihood $\tilde{l}_c$ to get the class posterior in a stable way. (We can also compute the log normalization constant, $\log p(\boldsymbol{x}) = \log Z = \log(\tilde{Z}) + L$.)

To compute a single Gaussian log density, $\ell_{ck} = \log \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{ck},\boldsymbol{\Sigma}_{ck})$, we need to evaluate $\log\det(\boldsymbol{\Sigma}_{ck})$ and $\boldsymbol{\Sigma}_{ck}^{-1}$. To make this efficient, we can use the matrix determinant lemma to compute

$$\det(\mathbf{A} + \mathbf{S}\mathbf{S}^\mathsf{T}) = \det(\mathbf{I} + \mathbf{S}^\mathsf{T}\mathbf{A}^{-1}\mathbf{S})\det(\mathbf{A}) \tag{28.31}$$

where $\mathbf{A} = \text{diag}(\boldsymbol{d}) + \epsilon\mathbf{I}$, and the matrix inversion lemma to compute

$$(\mathbf{A} + \mathbf{S}\mathbf{S}^\mathsf{T})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{S}(\mathbf{I} + \mathbf{S}^\mathsf{T}\mathbf{A}^{-1}\mathbf{S})^{-1}\mathbf{S}^\mathsf{T}\mathbf{A}^{-1} \tag{28.32}$$

(See also the discussion of mixture of factor analyzers in Section 28.3.3.)

28.2.6 Unidentifiability

The parameters of a mixture model are unidentiable, due to the **label switching** problem. To see this, consider fitting a GMM with 2 clusters (with parameters $\hat{\boldsymbol{\theta}}$) to a dataset which is generated from the true distribution p^* which we assume is also a GMM with 2 clusters (with parameters $\boldsymbol{\theta}^*$). The MLE will converge to the estimated parameters $\hat{\boldsymbol{\theta}}$ which minimizes $D_{\mathbb{KL}}\left(p(\boldsymbol{x}|\boldsymbol{\theta}^*) \parallel p(\boldsymbol{x}|\hat{\boldsymbol{\theta}})\right)$. However, there are 2 equally likely modes to the likelihood surface, $(\hat{\boldsymbol{\mu}}_1 = \boldsymbol{\mu}_1^*, \hat{\boldsymbol{\mu}}_2 = \boldsymbol{\mu}_2^*)$ and $(\hat{\boldsymbol{\mu}}_2 = \boldsymbol{\mu}_1^*, \hat{\boldsymbol{\mu}}_1 = \boldsymbol{\mu}_2^*)$, since the identify of the clusters is irrelevant. Hence computing the posterior mean of the cluster parameters $\boldsymbol{\mu}_k$ from some Bayesian inference procedure is meaningless. Instead, [Ste00] proposes a decision theoretic approach, in which the action space allows the user to ask questions about the clustering assignment (or parameters) after performing a suitable permutation of the labels. See also [Pap16] for an R library that implements this and other related algorithms.

28.3 Factor analysis

In this section, we discuss a simple latent factor model in which the prior $p(\boldsymbol{z})$ is Gaussian, and the likelihood $p(\boldsymbol{x}|\boldsymbol{z})$ is also Gaussian, using a linear decoder for the mean. This family includes many important special cases, such as PCA, as we discuss below. We also briefly discuss some simple extensions.

28.3.1 Factor analysis: the basics

Factor analysis corresponds to the following linear-Gaussian latent variable generative model:

$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \tag{28.33}$$
$$p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}|\mathbf{W}\boldsymbol{z} + \boldsymbol{\mu}, \boldsymbol{\Psi}) \tag{28.34}$$

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.

28.3.1.1 FA as a Gaussian with low-rank plus diagonal covariance

FA can be thought of as a low-rank version of a Gaussian distribution. To see this, note that the induced marginal distribution $p(\boldsymbol{x}|\boldsymbol{\theta})$ is a Gaussian (see Equation (2.129) for the derivation):

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \int \mathcal{N}(\boldsymbol{x}|\mathbf{W}\boldsymbol{z} + \boldsymbol{\mu}, \boldsymbol{\Psi})\mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)d\boldsymbol{z} \tag{28.35}$$
$$= \mathcal{N}(\boldsymbol{x}|\mathbf{W}\boldsymbol{\mu}_0 + \boldsymbol{\mu}, \boldsymbol{\Psi} + \mathbf{W}\boldsymbol{\Sigma}_0\mathbf{W}^{\mathsf{T}}) \tag{28.36}$$

The first and second moments can be derived as follows:

$$\begin{aligned} \mathbb{E}[\boldsymbol{x}] &= \mathbf{W}\boldsymbol{\mu}_0 + \boldsymbol{\mu} \\ \mathrm{Cov}[\boldsymbol{x}] &= \mathbf{W}\mathrm{Cov}[\boldsymbol{z}]\mathbf{W}^{\mathsf{T}} + \boldsymbol{\Psi} = \mathbf{W}\boldsymbol{\Sigma}_0\mathbf{W}^{\mathsf{T}} + \boldsymbol{\Psi} \end{aligned} \tag{28.37}$$

From this, we see that we can set $\boldsymbol{\mu}_0 = \mathbf{0}$ without loss of generality, since we can always absorb $\mathbf{W}\boldsymbol{\mu}_0$ into $\boldsymbol{\mu}$. Similarly, we can set $\boldsymbol{\Sigma}_0 = \mathbf{I}$ without loss of generality, since we can always absorb a correlated prior by using a new weight matrix, $\tilde{\mathbf{W}} = \mathbf{W}\boldsymbol{\Sigma}_0^{-\frac{1}{2}}$, since then

$$\mathrm{Cov}[\boldsymbol{x}] = \mathbf{W}\Sigma_0\mathbf{W}^{\mathsf{T}} + \boldsymbol{\Psi} = \tilde{\mathbf{W}}\tilde{\mathbf{W}}^{\mathsf{T}} + \boldsymbol{\Psi} \tag{28.38}$$

Finally, we see that we should restrict $\boldsymbol{\Psi}$ to be diagonal, otherwise we could set $\tilde{\mathbf{W}} = \mathbf{0}$, thus ignoring the latent factors, while still being able to model any covariance. After these simplifications we have the final model:

$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\mathbf{0}, \mathbf{I}) \tag{28.39}$$
$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\mathbf{W}\boldsymbol{z} + \boldsymbol{\mu}, \boldsymbol{\Psi}) \tag{28.40}$$

from which we get

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^{\mathsf{T}} + \boldsymbol{\Psi}) \tag{28.41}$$

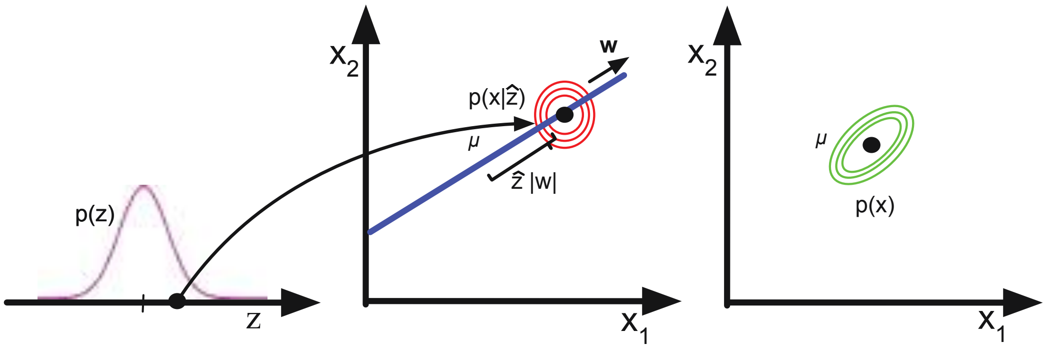


Figure 28.6: Illustration of the FA generative process, where we have $L = 1$ latent dimension generating $D = 2$ observed dimensions; we assume $\boldsymbol{\Psi} = \sigma^2\mathbf{I}$. The latent factor has value $z \in \mathbb{R}$, sampled from $p(z)$; this gets mapped to a 2d offset $\boldsymbol{\delta} = z\boldsymbol{w}$, where $\boldsymbol{w} \in \mathbb{R}^2$, which gets added to $\boldsymbol{\mu}$ to define a Gaussian $p(\boldsymbol{x}|z) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu} + \boldsymbol{\delta}, \sigma^2\mathbf{I})$. By integrating over z, we "slide" this circular Gaussian "spray can" along the principal component axis $\boldsymbol{w}$, which induces elliptical Gaussian contours in $\boldsymbol{x}$ space centered on $\boldsymbol{\mu}$. Adapted from Figure 12.9 of [Bis06].

For example, suppose where $L = 1$, $D = 2$ and $\boldsymbol{\Psi} = \sigma^2\mathbf{I}$. We illustrate the generative process in this case in Figure 28.6. We can think of this as taking an isotropic Gaussian "spray can", representing the likelihood $p(\boldsymbol{x}|\boldsymbol{z})$, and "sliding it along" the 1d line defined by $\boldsymbol{w}z + \boldsymbol{\mu}$ as we vary the 1d latent prior z. This induces an elongated (and hence correlated) Gaussian in 2d. That is, the induced distribution has the form $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{w}\boldsymbol{w}^\mathsf{T} + \sigma^2\mathbf{I})$.

In general, FA approximates the covariance matrix of the visible vector using a low-rank decomposition:

$$\mathbf{C} = \text{Cov}[\boldsymbol{x}] = \mathbf{W}\mathbf{W}^\mathsf{T} + \boldsymbol{\Psi} \tag{28.42}$$

This only uses $O(LD)$ parameters, which allows a flexible compromise between a full covariance Gaussian, with $O(D^2)$ parameters, and a diagonal covariance, with $O(D)$ parameters.

28.3.1.2 Computing the posterior

We can compute the posterior over the latent codes, $p(\boldsymbol{z}|\boldsymbol{x})$, using Bayes' rule for Gaussians. In particular, from Equation (2.121), we have

$$p(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_{z|x}, \boldsymbol{\Sigma}_{z|x}) \tag{28.43}$$

$$\boldsymbol{\Sigma}_{z|x} = (\mathbf{I} + \mathbf{W}^\mathsf{T}\boldsymbol{\Psi}^{-1}\mathbf{W})^{-1} = \mathbf{I} - \mathbf{W}^\mathsf{T}(\mathbf{W}\mathbf{W}^\mathsf{T} + \boldsymbol{\Psi})^{-1}\mathbf{W} \tag{28.44}$$

$$\boldsymbol{\mu}_{z|x} = \boldsymbol{\Sigma}_{z|x}[\mathbf{W}^\mathsf{T}\boldsymbol{\Psi}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})] = \mathbf{W}^\mathsf{T}(\mathbf{W}\mathbf{W}^\mathsf{T} + \boldsymbol{\Psi})^{-1}(\boldsymbol{x} - \boldsymbol{\mu}) \tag{28.45}$$

We can avoid inverting the $D \times D$ matrix $\mathbf{C} = \mathbf{W}\mathbf{W}^\mathsf{T} + \mathbf{\Psi}$ by using the matrix inversion lemma:

$$\mathbf{C}^{-1} = (\mathbf{W}\mathbf{W}^\mathsf{T} + \mathbf{\Psi})^{-1} \tag{28.46}$$

$$= \mathbf{\Psi}^{-1} - \mathbf{\Psi}^{-1}\mathbf{W}\underbrace{(\mathbf{I} + \mathbf{W}^\mathsf{T}\mathbf{\Psi}^{-1}\mathbf{W})^{-1}}_{\mathbf{L}^{-1}}\mathbf{W}^\mathsf{T}\mathbf{\Psi}^{-1} \tag{28.47}$$

where $\mathbf{L} = \mathbf{I} + \mathbf{W}^\mathsf{T}\mathbf{\Psi}^{-1}\mathbf{W}$ is $L \times L$.

28.3.1.3 Computing the likelihood

In this section, we discuss how to efficiently compute the log (marginal) likelihood, which is given by

$$\log p(\boldsymbol{x}|\boldsymbol{\mu}, \mathbf{C}) = -\frac{1}{2}\left[D\log(2\pi) + \log\det(\mathbf{C}) + \tilde{\boldsymbol{x}}^\mathsf{T}\mathbf{C}^{-1}\tilde{\boldsymbol{x}}\right] \tag{28.48}$$

where $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{\mu}$, and $\mathbf{C} = \mathbf{W}\mathbf{W}^\mathsf{T} + \mathbf{\Psi}$. Using Equation (28.47), the Mahalanobis distance can be computed using

$$\tilde{\boldsymbol{x}}^\mathsf{T}\mathbf{C}^{-1}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{x}}^\mathsf{T}\left[\mathbf{\Psi}^{-1}\tilde{\boldsymbol{x}} - \mathbf{\Psi}^{-1}\mathbf{W}\mathbf{L}^{-1}(\mathbf{W}^\mathsf{T}\mathbf{\Psi}^{-1}\tilde{\boldsymbol{x}})\right] \tag{28.49}$$

which takes $O(L^3 + LD)$ to compute. From the matrix determinant lemma, the log determinant is given by

$$\log\det(\mathbf{C}) = \log\det(\mathbf{L}) + \log\det(\mathbf{\Psi}) \tag{28.50}$$

which takes $O(L^3 + D)$ to compute. (See also Section 28.2.5, where we discuss fitting low-rank GMM classifiers discriminatively, which requires similar computations.)

28.3.1.4 Model fitting using EM

We can compute the MLE for the FA model either by performing gradient ascent on the log likelihood in Equation (28.48), or by using the EM algorithm [RT82; GH96b]. The latter can converge faster, and automatically satisfies positivity constraints on $\mathbf{\Psi}$. We give the details below, assuming that the observed data is standardized, so $\boldsymbol{\mu} = \mathbf{0}$ for notational simplicity.

In the E step, we compute the following expected sufficient statistics:

$$\mathbf{E}_{\boldsymbol{x},\boldsymbol{z}} = \sum_{n=1}^{N} \boldsymbol{x}_n \mathbb{E}\left[\boldsymbol{z}|\boldsymbol{x}_n\right]^\mathsf{T} \tag{28.51}$$

$$\mathbf{E}_{\boldsymbol{z},\boldsymbol{z}} = \sum_{n=1}^{N} \mathbb{E}\left[\boldsymbol{z}\boldsymbol{z}^\mathsf{T}|\boldsymbol{x}_n\right] \tag{28.52}$$

$$\mathbf{E}_{\boldsymbol{x},\boldsymbol{x}} = \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\mathsf{T} \tag{28.53}$$

where

$$\mathbb{E}\left[\boldsymbol{z}|\boldsymbol{x}\right] = \mathbf{B}\boldsymbol{x} \tag{28.54}$$

$$\mathbb{E}\left[\boldsymbol{z}\boldsymbol{z}^\mathsf{T}|\boldsymbol{x}\right] = \mathrm{Cov}\left[\boldsymbol{z}|\boldsymbol{x}\right] + \mathbb{E}\left[\boldsymbol{z}|\boldsymbol{x}\right]\mathbb{E}\left[\boldsymbol{z}|\boldsymbol{x}\right]^\mathsf{T} = \mathbf{I} - \mathbf{B}\mathbf{W} + \mathbf{B}\boldsymbol{x}\boldsymbol{x}^\mathsf{T}\mathbf{B}^\mathsf{T} \tag{28.55}$$

$$\mathbf{B} \triangleq \mathbf{W}^\mathsf{T}(\mathbf{\Psi} + \mathbf{W}\mathbf{W}^\mathsf{T})^{-1} = \mathbf{W}^\mathsf{T}\mathbf{C}^{-1} \tag{28.56}$$

In the M step, we have

$$\mathbf{W}^{\text{new}} = \mathbf{E}_{\boldsymbol{x},\boldsymbol{z}} \mathbf{E}_{\boldsymbol{z},\boldsymbol{z}}^{-1} \tag{28.57}$$

$$\boldsymbol{\Psi}^{\text{new}} = \frac{1}{N} \text{diag}\left(\mathbf{E}_{\boldsymbol{x},\boldsymbol{x}} - \mathbf{W}^{\text{new}} \mathbf{E}_{\boldsymbol{x},\boldsymbol{z}}^{\mathsf{T}}\right) \tag{28.58}$$

28.3.1.5 Handling missing data

We can also perform posterior inference in the presence of missing data (if we make the missing at random assumption — see Section 3.11 for discussion). In particular, let us partition $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$, $\mathbf{W} = [\mathbf{W}_1, \mathbf{W}_2]$, and $\boldsymbol{\mu} = [\boldsymbol{\mu}_1, \boldsymbol{\mu}_2]$, and suppose $\boldsymbol{x}_2$ is missing (unknown). From Supplementary Section 2.1.1, we have

$$p(\boldsymbol{z}|\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_{z|1}, \boldsymbol{\Sigma}_{z|1}) \tag{28.59}$$

$$\boldsymbol{\Sigma}_{z|1}^{-1} = \mathbf{I} + \mathbf{W}_1^{\mathsf{T}} \boldsymbol{\Sigma}_{11}^{-1} \mathbf{W}_1 \tag{28.60}$$

$$\boldsymbol{\mu}_{z|1} = \boldsymbol{\Sigma}_{z|1} [\mathbf{W}_1^{\mathsf{T}} \boldsymbol{\Sigma}_{11}^{-1} (\boldsymbol{x}_1 - \boldsymbol{\mu}_1)] \tag{28.61}$$

where $\boldsymbol{\Sigma}_{11}$ is the top left block of $\boldsymbol{\Psi}$.

We can modify the EM algorithm to fit the model in the presence of missing data in the obvious way.

28.3.1.6 Unidentifiability of the parameters

The parameters of a FA model are unidentifiable. To see this, consider a model with weights $\mathbf{W}$ and observation covariance $\boldsymbol{\Psi}$. We have

$$\text{Cov}[\boldsymbol{x}] = \mathbf{W}\mathbb{E}\left[\boldsymbol{z}\boldsymbol{z}^{\mathsf{T}}\right]\mathbf{W}^{\mathsf{T}} + \mathbb{E}\left[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^{\mathsf{T}}\right] = \mathbf{W}\mathbf{W}^{\mathsf{T}} + \boldsymbol{\Psi} \tag{28.62}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Psi})$ is the observation noise. Now consider a different model with weights $\tilde{\mathbf{W}} = \mathbf{W}\mathbf{R}$, where $\mathbf{R}$ is an arbitrary orthogonal rotation matrix, satisfying $\mathbf{R}\mathbf{R}^{\mathsf{T}} = \mathbf{I}$. This has the same likelihood, since

$$\text{Cov}[\boldsymbol{x}] = \tilde{\mathbf{W}}\mathbb{E}\left[\boldsymbol{z}\boldsymbol{z}^{\mathsf{T}}\right]\tilde{\mathbf{W}}^{\mathsf{T}} + \mathbb{E}\left[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^{\mathsf{T}}\right] = \mathbf{W}\mathbf{R}\mathbf{R}^{\mathsf{T}}\mathbf{W}^{\mathsf{T}} + \boldsymbol{\Psi} = \mathbf{W}\mathbf{W}^{\mathsf{T}} + \boldsymbol{\Psi} \tag{28.63}$$

Geometrically, multiplying $\mathbf{W}$ by an orthogonal matrix is like rotating $\boldsymbol{z}$ before generating $\boldsymbol{x}$; but since $\boldsymbol{z}$ is drawn from an isotropic Gaussian, this makes no difference to the likelihood. Consequently, we cannot uniquely identify $\mathbf{W}$, and therefore cannot uniquely identify the latent factors, either. This is called the "**factor rotations problem**" (see e.g., [Dar80]).

To break this symmetry, several solutions can be used, as we discuss below.

- **Forcing W to have orthogonal columns.**. In (P)PCA, we force $\mathbf{W}$ to have orthogonal columns. and to sort the dimensions in order of decreasing eigenvalue (of $\mathbf{W}\mathbf{W}^{\mathsf{T}}$). However, this still does not ensure identifiability, since we can always multiply $\mathbf{W}$ by another orthogonal matrix without changing the likelihood.
- **Forcing W to be lower triangular**. One way to resolve permutation unidentifiability, which is popular in the Bayesian community (e.g., [LW04]), is to ensure that the first visible feature is only generated by the first latent factor, the second visible feature is only generated by the first

two latent factors, and so on. For example, if $L = 3$ and $D = 4$, the correspond factor loading matrix is given by

$$\mathbf{W} = \begin{pmatrix} w_{11} & 0 & 0 \\ w_{21} & w_{22} & 0 \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{pmatrix} \tag{28.64}$$

We also require that $w_{kk} > 0$ for $k = 1 : L$. The total number of parameters in this constrained matrix is $D + DL - L(L-1)/2$, which is equal to the number of uniquely identifiable parameters in FA (excluding the mean).[1] The disadvantage of this method is that the first L visible variables, known as the **founder variables**, affect the interpretation of the latent factors, and so must be chosen carefully.

- **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 ℓ_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, $p(\boldsymbol{z})$, with a non-Gaussian distribution, we can sometimes uniquely identify $\mathbf{W}$, as well as the latent factors. See e.g., [KKH20] for details.

28.3.2 Probabilistic PCA

In this section, we consider a special case of the factor analysis model in which $\mathbf{W}$ has orthogonal columns and $\boldsymbol{\Psi} = \sigma^2 \mathbf{I}$, so $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \mathbf{C})$ where $\mathbf{C} = \mathbf{W}\mathbf{W}^\mathsf{T} + \sigma^2 \mathbf{I}$. This model is called **probabilistic principal components analysis** (**PPCA**) [TB99], or **sensible PCA** [Row97].

The advantage of PPCA over factor analysis is that the MLE has a closed form solution, as we show in Section 28.3.2.2. The advantage of PPCA over non-probabilistic PCA is that the model defines a proper likelihood function, which makes it easier to extend in various ways e.g., by creating mixtures of PPCA models (see Section 28.3.3).

28.3.2.1 Derivation of the MLE

The log likelihood for PPCA is given by

$$\log p(\mathbf{X}|\boldsymbol{\mu}, \mathbf{W}, \sigma^2) = -\frac{ND}{2}\log(2\pi) - \frac{N}{2}\log|\mathbf{C}| - \frac{1}{2}\sum_{n=1}^{N}(\boldsymbol{x}_n - \boldsymbol{\mu})^\mathsf{T}\mathbf{C}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}) \tag{28.65}$$

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 $\mathbf{R}$, since that is the dimensionality of the space of orthogonal matrices of size $L \times L$. To see this, note that there are $L-1$ free parameters in $\mathbf{R}$ in the first column (since the column vector must be normalized to unit length), there are $L-2$ free parameters in the second column (which must be orthogonal to the first), and so on.

The MLE for $\boldsymbol{\mu}$ is $\overline{\boldsymbol{x}}$. Plugging in gives

$$\log p(\mathbf{X}|\boldsymbol{\mu}, \mathbf{W}, \sigma^2) = -\frac{N}{2}\left[D\log(2\pi) + \log|\mathbf{C}| + \operatorname{tr}(\mathbf{C}^{-1}\mathbf{S})\right] \tag{28.66}$$

where $\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n - \overline{\boldsymbol{x}})(\boldsymbol{x}_n - \overline{\boldsymbol{x}})^\mathsf{T}$ is the empirical covariance matrix.

In [TB99; Row97] they show that the maximum of this objective must satisfy

$$\hat{\mathbf{W}} = \mathbf{U}_L(\boldsymbol{\Lambda}_L - \sigma^2\mathbf{I})^{\frac{1}{2}}\mathbf{R} \tag{28.67}$$

where $\mathbf{U}_L$ is a $D \times L$ matrix whose columns are given by the L eigenvectors of $\mathbf{S}$ with largest eigenvalues, $\boldsymbol{\Lambda}_L$ 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

$$\mathbf{C} = \mathbf{W}\mathbf{W}^\mathsf{T} + \sigma^2\mathbf{I} = \mathbf{U}_L(\boldsymbol{\Lambda}_L - \sigma^2\mathbf{I})\mathbf{U}_L^\mathsf{T} + \sigma^2\mathbf{I} \tag{28.68}$$

The MLE for the observation variance is

$$\sigma^2 = \frac{1}{D-L}\sum_{i=L+1}^{D}\lambda_i \tag{28.69}$$

which is the average distortion associated with the discarded dimensions. If $L = D$, then the estimated noise is 0, since the model collapses to $\boldsymbol{z} = \boldsymbol{x}$.

28.3.2.2 PCA is recovered in the noise-free limit

In the noise-free limit, where $\sigma^2 = 0$, we see that the MLE (for $\mathbf{R} = \mathbf{I}$) is

$$\hat{\mathbf{W}} = \mathbf{U}_L\boldsymbol{\Lambda}_L^{\frac{1}{2}} \tag{28.70}$$

so

$$\hat{\mathbf{C}} = \hat{\mathbf{W}}\hat{\mathbf{W}}^\mathsf{T} = \mathbf{U}_L\boldsymbol{\Lambda}_L^{\frac{1}{2}}\boldsymbol{\Lambda}_L^{\frac{1}{2}}\mathbf{U}_L^\mathsf{T} = \mathbf{S}_L \tag{28.71}$$

where $\mathbf{S}_L$ is the rank L approximation to $\mathbf{S}$. This is the same as standard PCA.

28.3.2.3 Computing the posterior

To use PPCA as an alternative to PCA, we need to compute the posterior mean $\mathbb{E}[\boldsymbol{z}|\boldsymbol{x}]$, which is the equivalent of the PCA encoder model. Using the factor analysis results from Section 28.3.1.2, we have

$$p(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\sigma^{-2}\boldsymbol{\Sigma}\mathbf{W}^\mathsf{T}(\boldsymbol{x} - \boldsymbol{\mu}), \boldsymbol{\Sigma}) \tag{28.72}$$

where

$$\boldsymbol{\Sigma}^{-1} = \mathbf{I} + \sigma^{-2}\mathbf{W}^\mathsf{T}\mathbf{W} = \frac{1}{\sigma^2}\underbrace{(\sigma^2\mathbf{I} + \mathbf{W}^\mathsf{T}\mathbf{W})}_{\mathbf{M}} \tag{28.73}$$

Hence

$$p(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\mathbf{M}^{-1}\mathbf{W}^{\mathsf{T}}(\boldsymbol{x}-\boldsymbol{\mu}), \sigma^2\mathbf{M}^{-1}) \tag{28.74}$$

In the $\sigma^2 = 0$ limit, we have $\mathbf{M} = \mathbf{W}^{\mathsf{T}}\mathbf{W}$ and so

$$\mathbb{E}\left[\boldsymbol{z}|\boldsymbol{x}\right] = (\mathbf{W}^{\mathsf{T}}\mathbf{W})^{-1}\mathbf{W}^{\mathsf{T}}(\boldsymbol{x}-\overline{\boldsymbol{x}}) \tag{28.75}$$

This is the orthogonal projection of the data into the latent space, as in standard PCA.

28.3.2.4 Model fitting using EM

In Section 28.3.2.2, we showed how to fit the PCA model using an eigenvector method. We can also use EM, by leveraging the probabilistic formulation of PPCA in the zero noise limit, $\sigma^2 = 0$, as shown by [Row97].

In particular, let $\tilde{\mathbf{Z}} = \mathbf{Z}^{\mathsf{T}}$ be an $L \times N$ matrix storing the posterior means (low-dimensional representations) along its columns. Similarly, let $\tilde{\boldsymbol{x}}_n = \boldsymbol{x}_n - \hat{\boldsymbol{\mu}}$ be the centered examples stored along the columns of $\tilde{\mathbf{X}}$. From Equation (28.75), when $\sigma^2 = 0$, we have

$$\tilde{\mathbf{Z}} = (\mathbf{W}^{\mathsf{T}}\mathbf{W})^{-1}\mathbf{W}^{\mathsf{T}}\tilde{\mathbf{X}} \tag{28.76}$$

This constitutes the E step. Notice that this is just an orthogonal projection of the data.

From Equation (28.57), the M step is given by

$$\hat{\mathbf{W}} = \left[\sum_n \tilde{\boldsymbol{x}}_n \mathbb{E}\left[\boldsymbol{z}_n|\tilde{\boldsymbol{x}}_n\right]^{\mathsf{T}}\right]\left[\sum_n \mathbb{E}\left[\boldsymbol{z}_n|\tilde{\boldsymbol{x}}_n\right]\mathbb{E}\left[\boldsymbol{z}_n|\tilde{\boldsymbol{x}}_n\right]^{\mathsf{T}}\right]^{-1} \tag{28.77}$$

where we exploited the fact that $\boldsymbol{\Sigma} = \mathrm{Cov}\left[\boldsymbol{z}|\tilde{\boldsymbol{x}}\right] = 0\mathbf{I}$ when $\sigma^2 = 0$.

In summary, here is the entire algorithm:

$$\tilde{\mathbf{Z}} = (\mathbf{W}^{\mathsf{T}}\mathbf{W})^{-1}\mathbf{W}^{\mathsf{T}}\tilde{\mathbf{X}} \text{ (E step)} \tag{28.78}$$

$$\mathbf{W} = \tilde{\mathbf{X}}\tilde{\mathbf{Z}}^{\mathsf{T}}(\tilde{\mathbf{Z}}\tilde{\mathbf{Z}}^{\mathsf{T}})^{-1} \text{ (M step)} \tag{28.79}$$

It is worth comparing this expression to the MLE for multi-output linear regression, which has the form $\mathbf{W} = (\sum_n \boldsymbol{y}_n\boldsymbol{x}_n^{\mathsf{T}})(\sum_n \boldsymbol{x}_n\boldsymbol{x}_n^{\mathsf{T}})^{-1}$. 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.

[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 of $\mathbf{S}$. 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].

28.3.3 Mixture of factor analyzers

The factor analysis model (Section 28.3.1) 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

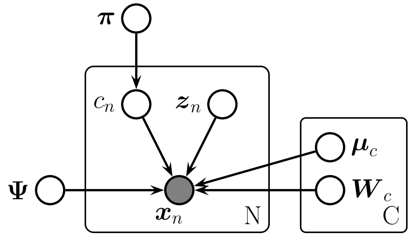


Figure 28.7: Mixture of factor analyzers as a PGM.

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 analyzers** or MFA [GH96b]. 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 covariance matrix which is diagonal plus low-rank.

28.3.3.1 Model definition

The generative story is as follows. First we sample a discrete latent indicator $m_n \in \{1, \dots, K\}$ from discrete distribution $\text{Cat}(\cdot|\boldsymbol{\pi})$ to specify which subspace (cluster) we should use to generate the data. If $m_n = k$, we sample $\boldsymbol{z}_n$ from a Gaussian prior and pass it through the $\mathbf{W}_k$ matrix, where $\mathbf{W}_k$ maps from the L-dimensional subspace to the D-dimensional visible space.[2] Finally we add Gaussian observation noise sampled from $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Psi})$. Thus the model is as follows:

$$p(\boldsymbol{x}_n|\boldsymbol{z}_n, m_n = k, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k + \mathbf{W}_k\boldsymbol{z}_n, \boldsymbol{\Psi}) \tag{28.80}$$

$$p(\boldsymbol{z}_n|\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{z}_n|\mathbf{0}, \mathbf{I}) \tag{28.81}$$

$$p(m_n|\boldsymbol{\theta}) = \text{Cat}(m_n|\boldsymbol{\pi}) \tag{28.82}$$

The corresponding distribution in the visible space is given by

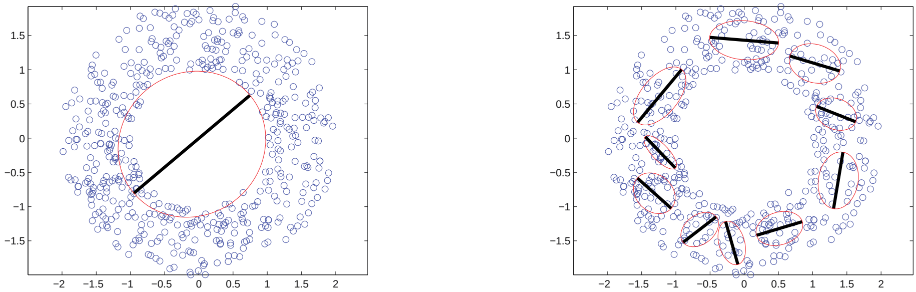
$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_k p(m = k) \int p(\boldsymbol{z}|c)p(\boldsymbol{x}|\boldsymbol{z}, m)\, d\boldsymbol{z} \tag{28.83}$$

$$= \sum_k \pi_k \int \mathcal{N}(\boldsymbol{z}|\mathbf{0}, \mathbf{I})\mathcal{N}(\boldsymbol{x}|\mathbf{W}_k\boldsymbol{z} + \boldsymbol{\mu}_k, \boldsymbol{\Psi})\, d\boldsymbol{z} \tag{28.84}$$

$$= \sum_k \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Psi} + \mathbf{W}_k\mathbf{W}_k^\mathsf{T}) \tag{28.85}$$

In the special case that $\boldsymbol{\Psi} = \sigma^2\mathbf{I}$, we get a mixture of PPCA models. See Figure 28.8 for an example of the method applied to some 2d data.

2. If we allow $\boldsymbol{z}_n$ to depend on m_n, we can let each subspace have a different dimensionality, as suggested in [KS15].



(a) *(b)*

Figure 28.8: 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 mix_ppca_demo.ipynb.

We can think of this as a low-rank version of a mixture of Gaussians. In particular, this model needs $O(KLD)$ parameters instead of the $O(KD^2)$ parameters needed for a mixture of full covariance Gaussians. This can reduce overfitting.

28.3.3.2 Model fitting using EM

We can fit this model using EM, extending the results of Section 28.3.1.4 (see [GH96b] for the derivation, and [ZY08] for a faster ECM version). In the E step, we compute the posterior responsibility of cluster j for datapoint i using

$$r_{ij} \triangleq p(m_i = j|\boldsymbol{x}_i, \boldsymbol{\theta}) \propto \pi_j \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \mathbf{W}_j\mathbf{W}_j^{\mathsf{T}} + \boldsymbol{\Psi}) \tag{28.86}$$

We also compute the following expected sufficient statistics, where we define $w_j = \mathbb{I}(m = j)$ and $\mathbf{B}_j = \mathbf{W}_j^{\mathsf{T}}(\boldsymbol{\Psi} + \mathbf{W}_j\mathbf{W}_j^{\mathsf{T}})^{-1}$:

$$\mathbb{E}[w_j \boldsymbol{z}|\boldsymbol{x}_i] = \mathbb{E}[w_j|\boldsymbol{x}_i]\,\mathbb{E}[\boldsymbol{z}|w_j, \boldsymbol{x}_i] = r_{ij}\mathbf{B}_j(\boldsymbol{x}_i - \boldsymbol{\mu}_j) \tag{28.87}$$

$$\mathbb{E}\left[w_j \boldsymbol{z}\boldsymbol{z}^{\mathsf{T}}|\boldsymbol{x}_i\right] = \mathbb{E}[w_j|\boldsymbol{x}_i]\,\mathbb{E}\left[\boldsymbol{z}\boldsymbol{z}^{\mathsf{T}}|w_j, \boldsymbol{x}_i\right] = r_{ij}(\mathbf{I} - \mathbf{B}_j\mathbf{W}_j + \mathbf{B}_j(\boldsymbol{x} - \boldsymbol{\mu}_j)(\boldsymbol{x} - \boldsymbol{\mu}_j)^{\mathsf{T}}\mathbf{B}_j^{\mathsf{T}}) \tag{28.88}$$

In the M step, we compute the following parameter update for the augmented factor loading matrix:

$$[\mathbf{W}_j^{\text{new}}\ \boldsymbol{\mu}_j^{\text{new}}] \triangleq \tilde{\mathbf{W}}_j^{\text{new}} = (\sum_i r_{ij}\boldsymbol{x}_i\mathbb{E}[\tilde{\boldsymbol{z}}|\boldsymbol{x}_i, w_j]^{\mathsf{T}})(\sum_i r_{ij}\mathbb{E}\left[\tilde{\boldsymbol{z}}\tilde{\boldsymbol{z}}^{\mathsf{T}}|\boldsymbol{x}_i, w_j\right])^{-1} \tag{28.89}$$

where $\tilde{\boldsymbol{z}} = [\boldsymbol{z}; 1]$,

$$\mathbb{E}[\tilde{\boldsymbol{z}}|\boldsymbol{x}_i, w_j] = \begin{pmatrix} \mathbb{E}[\boldsymbol{z}|\boldsymbol{x}_i, w_j] \\ 1 \end{pmatrix} \tag{28.90}$$

$$\mathbb{E}\left[\tilde{\boldsymbol{z}}\tilde{\boldsymbol{z}}^{\mathsf{T}}|\boldsymbol{x}_i, w_j\right] = \begin{pmatrix} \mathbb{E}\left[\tilde{\boldsymbol{z}}\tilde{\boldsymbol{z}}^{\mathsf{T}}|\boldsymbol{x}_i, w_j\right] & \mathbb{E}[\tilde{\boldsymbol{z}}|\boldsymbol{x}_i, w_j] \\ \mathbb{E}[\tilde{\boldsymbol{z}}|\boldsymbol{x}_i, w_j]^{\mathsf{T}} & 1 \end{pmatrix} \tag{28.91}$$

The new covariance matrix is given by

$$\boldsymbol{\Psi}^{\text{new}} = \frac{1}{N} \text{diag} \left(\sum_{ij} r_{ij} (\boldsymbol{x}_i - \tilde{\mathbf{W}}_j^{\text{new}} \mathbb{E}\left[\tilde{\boldsymbol{z}} | \boldsymbol{x}_i, w_j\right]) \boldsymbol{x}_i^{\mathsf{T}} \right) \tag{28.92}$$

And the new mixing weights are given by

$$\pi_j^{\text{new}} = \frac{1}{N} \sum_{i=1}^{N} r_{ij} \tag{28.93}$$

28.3.3.3 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 10.1.5) 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.

28.3.3.4 Model selection

To choose the number of mixture components K, 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 N_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 15.2.8). This can be implemented using variational Bayes EM (Section 10.3.5); for details, see [Bis99; GB00].

Figure 28.9 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 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 28.10 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 N_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].

28.3.3.5 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

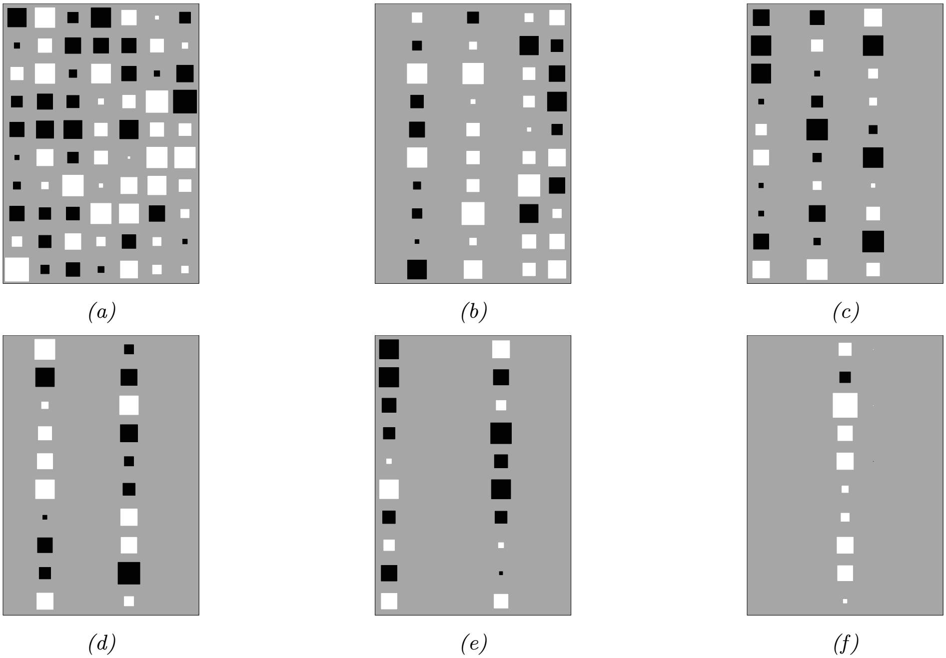


Figure 28.9: Illustration of estimating the effective dimensionalities in a mixture of factor analysers using variational Bayes EM with an ARD prior. Black are negative values, white are positive, gray is 0. The blank columns have been forced to 0 via the ARD mechanism, reducing the effective dimensionality. The data was generated from 6 clusters with intrinsic dimensionalities of 7, 4, 3, 2, 2, 1, which the method has successfully estimated. From Figure 4.4 of [Bea03]. Used with kind permission of Matt Beal.

<table>
<tr><th rowspan="2">number of points per cluster</th><th colspan="6">intrinsic dimensionalities</th></tr>
<tr><th>1</th><th>7</th><th>4</th><th>3</th><th>2</th><th>2</th></tr>
<tr><td>8</td><td colspan="4">2</td><td colspan="2">1</td></tr>
<tr><td>8</td><td colspan="2">1</td><td colspan="4">2</td></tr>
<tr><td>16</td><td>1</td><td colspan="4">4</td><td>2</td></tr>
<tr><td>32</td><td>1</td><td>6</td><td>3</td><td>3</td><td>2</td><td>2</td></tr>
<tr><td>64</td><td>1</td><td>7</td><td>4</td><td>3</td><td>2</td><td>2</td></tr>
<tr><td>128</td><td>1</td><td>7</td><td>4</td><td>3</td><td>2</td><td>2</td></tr>
</table>

Figure 28.10: 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.

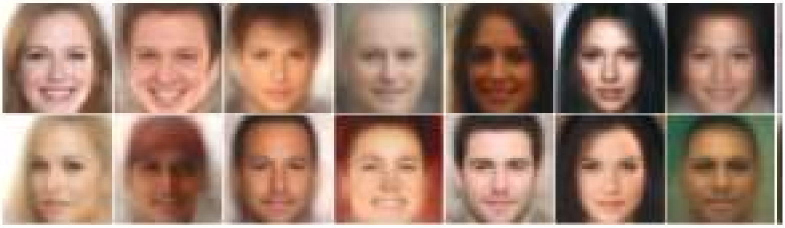
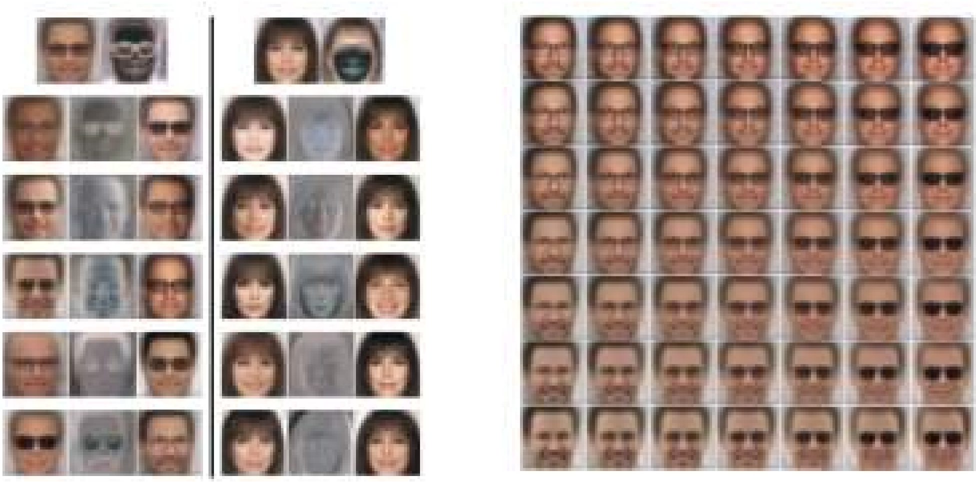


Figure 28.11: Random samples from the MixFA model fit to CelebA. Generated by mix_ppca_celebA.ipynb. Adapted from Figure 4 of [RW18]. Used with kind permission of Yair Weiss.



(a) (b)

Figure 28.12: (a) Visualization of the parameters learned by the MFA model. The top row shows the mean $\boldsymbol{\mu}_k$ and noise variance $\boldsymbol{\Psi}_k$, 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 $\boldsymbol{\mu}_k - \mathbf{W}_k[:, i]$; in the middle, we show $0.5 + \mathbf{W}_k[:, i]$, and on the right we show $\boldsymbol{\mu}_k + \mathbf{W}_k[:, i]$. (b) Images generated by computing $\boldsymbol{\mu}_k + z_1\mathbf{W}_k[:, i] + z_2\mathbf{W}_k[:, j]$, 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 $\boldsymbol{\mu}_k$. 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 28.3.1.3 to efficiently compute the log likelihood, despite the high dimensionality. The $\boldsymbol{\mu}_k$ 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.

Figure 28.13: Samples from the 100 CelebA images with lowest likelihood under the MFA model. Generated by mix_ppca_celebA.ipynb. Adapted from Figure 7a of [RW18]. Used with kind permission of Yair Weiss.

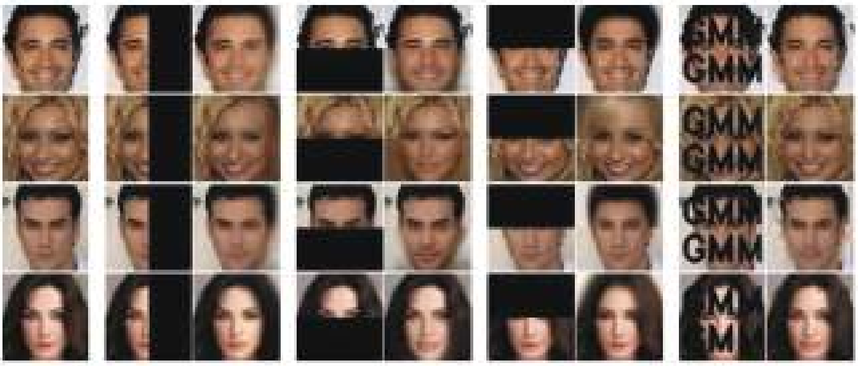


Figure 28.14: Illustration of image imputation using an MFA. Left column shows 4 original images. Subsequent pairs of columns show an occluded input, and a predicted output. Generated by mix_ppca_celebA.ipynb. Adapted from Figure 7b of [RW18]. Used with kind permission of Yair Weiss.

Figure 28.11 shows some images generated from the fitted model. The results are suprisingly good for such a simple locally linear model. The reason the method works is similar to the discussion in Section 28.2.4.1: essentially the $\mathbf{W}_k$ matrix learns a set of L-dimensional basis functions for the subset of face images that get mapped to cluster k. See Figure 28.12 for an illustration.

There are several advantages to this model compared to VAEs and GANs. First, [RW18], showed that this MixFA model captures more of the modes of the data distribution than more sophisticated generative models, such as VAEs (Section 21.2) and GANs (Chapter 26). Second, we can compute the exact likelihood $p(\boldsymbol{x})$, so we can compute outliers or unusual images. This is illustrated in Figure 28.13.

Third, we can perform image imputation from partially observed images given arbitrary missingness patterns. To see this, let us partition $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$, where $\boldsymbol{x}_1$ (of size D_1) is observed and $\boldsymbol{x}_2$ (of size

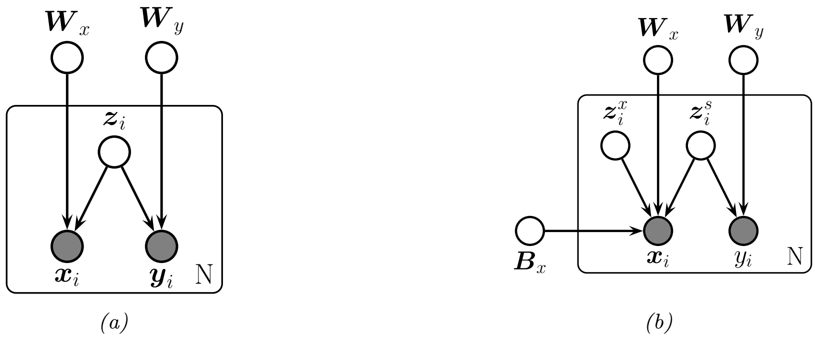


Figure 28.15: Gaussian latent factor models for paired data. (a) Supervised PCA. (b) Partial least squares.

$D_2 = D - D_1$) is missing. We can compute the most probable cluster using

$$k^* = \operatorname*{argmax}_{k=1}^{K} p(c = k)p(\boldsymbol{x}_1|c = k) \tag{28.94}$$

where

$$\log p(\boldsymbol{x}_1|\boldsymbol{\mu}_k, \mathbf{C}_k) = -\frac{1}{2}\left[D_1 \log(2\pi) + \log\det(\mathbf{C}_{k,11}) + \tilde{\boldsymbol{x}}_1^\mathsf{T}\mathbf{C}_{k,11}^{-1}\tilde{\boldsymbol{x}}_1\right] \tag{28.95}$$

where $\mathbf{C}_{k,11}$ is the top left $D_1 \times D_1$ block of $\mathbf{W}_k\mathbf{W}_k^\mathsf{T} + \boldsymbol{\Psi}_k$, and $\tilde{\boldsymbol{x}}_1 = \boldsymbol{x}_1 - \boldsymbol{\mu}_k[1 : D_1]$. Once we know which discrete mixture component to use, we can compute the Gaussian posterior $p(\boldsymbol{z}|\boldsymbol{x}_1, k^*)$ using Equation (28.59). Let $\hat{\boldsymbol{z}} = \mathbb{E}[\boldsymbol{z}|\boldsymbol{x}_1, k^*]$. Given this, we can compute the predicted output for the full image:

$$\hat{\boldsymbol{x}} = \mathbf{W}_{k^*}\hat{\boldsymbol{z}} + \boldsymbol{\mu}_{k^*} \tag{28.96}$$

We then use the estimate $\boldsymbol{x}' = [\boldsymbol{x}_1, \hat{\boldsymbol{x}}_2]$, so the observed pixels are not changed. This is an example of **image imputation**, and is illustrated in Figure 28.14. Note that we can condition on an arbitrary subset of pixels, and fill in the rest, whereas some other models (e.g., autoregressive models) can only predict the bottom right given the top left (since they assume a generative model which works in raster-scan order).

28.3.4 Factor analysis models for paired data

In this section, we discuss linear-Gaussian factor analysis models when we have two kinds of observed variables, $\boldsymbol{x} \in \mathbb{R}^{D_x}$ and $\boldsymbol{y} \in \mathbb{R}^{D_y}$, which are paired. These often correspond to different sensors or modalities (e.g., images and sound). We follow the presentation of [Vir10].

28.3.4.1 Supervised PCA

If we have two observed signals, we can model the joint $p(\boldsymbol{x}, \boldsymbol{y})$ using a shared low-dimensional representation using the following linear Gaussian model:

$$p(\boldsymbol{z}_n) = \mathcal{N}(\boldsymbol{z}_n|\mathbf{0}, \mathbf{I}_L) \tag{28.97}$$

$$p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}_n|\mathbf{W}_x\boldsymbol{z}_n, \sigma_x^2\mathbf{I}_{D_x}) \tag{28.98}$$

$$p(\boldsymbol{y}_n|\boldsymbol{z}_n, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}_n|\mathbf{W}_y\boldsymbol{z}_n, \sigma_y^2\mathbf{I}_{D_y}) \tag{28.99}$$

This is illustrated as a graphical model in Figure 28.15a. The intuition is that $\boldsymbol{z}_n$ is a shared latent subspace, that captures features that $\boldsymbol{x}_n$ and $\boldsymbol{y}_n$ have in common. The variance terms σ_x and σ_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** [Wes03].

We can marginalize out $\boldsymbol{z}_n$ to get $p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{\theta})$. If $\boldsymbol{y}_n$ is a scalar, this becomes

$$p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}) = \mathcal{N}(y_n|\boldsymbol{x}_n^\mathsf{T}\boldsymbol{v}, \boldsymbol{w}_y^\mathsf{T}\mathbf{C}\boldsymbol{w}_y + \sigma_y^2) \tag{28.100}$$

$$\mathbf{C} = (\mathbf{I} + \sigma_x^{-2}\mathbf{W}_x^\mathsf{T}\mathbf{W}_x)^{-1} \tag{28.101}$$

$$\boldsymbol{v} = \sigma_x^{-2}\mathbf{C}\mathbf{W}_x\boldsymbol{w}_y \tag{28.102}$$

To apply this to the classification setting, we can replace the Gaussian $p(\boldsymbol{y}|\boldsymbol{z})$ with a logistic regression model:

$$p(y_n|\boldsymbol{z}_n, \boldsymbol{\theta}) = \text{Ber}(y_n|\sigma(\boldsymbol{w}_y^\mathsf{T}\boldsymbol{z}_n)) \tag{28.103}$$

In this case, we can no longer compute the marginal posterior predictive $p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta})$ in closed form, but we can use techniques similar to exponential family PCA (see [Guo09] for details).

The above model is completely symmetric in $\boldsymbol{x}$ and $\boldsymbol{y}$. If our goal is to predict $\boldsymbol{y}$ from $\boldsymbol{x}$ via the latent bottleneck $\boldsymbol{z}$, then we might want to upweight the likelihood term for $\boldsymbol{y}$, as proposed in [Ris+08]. This gives

$$p(\mathbf{X}, \mathbf{Y}, \mathbf{Z}|\boldsymbol{\theta}) = p(\mathbf{Y}|\mathbf{Z}, \mathbf{W}_y)p(\mathbf{X}|\mathbf{Z}, \mathbf{W}_x)^\alpha p(\mathbf{Z}) \tag{28.104}$$

where $\alpha \leq 1$ controls the relative importance of modeling the two sources. The value of α can be chosen by cross-validation.

28.3.4.2 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, $\boldsymbol{z}_i^x$, and to let the rest of the subspace, $\boldsymbol{z}_i^s$, be shared between input and output. The model has the form

$$p(\boldsymbol{z}_i) = \mathcal{N}(\boldsymbol{z}_i^s|\mathbf{0}, \mathbf{I}_{L_s})\mathcal{N}(\boldsymbol{z}_i^x|\mathbf{0}, \mathbf{I}_{L_x}) \tag{28.105}$$

$$p(\boldsymbol{y}_i|\boldsymbol{z}_i) = \mathcal{N}(\mathbf{W}_y\boldsymbol{z}_i^s + \boldsymbol{\mu}_y, \sigma^2\mathbf{I}_{D_y}) \tag{28.106}$$

$$p(\boldsymbol{x}_i|\boldsymbol{z}_i) = \mathcal{N}(\mathbf{W}_x\boldsymbol{z}_i^s + \mathbf{B}_x\boldsymbol{z}_i^x + \boldsymbol{\mu}_x, \sigma^2\mathbf{I}_{D_x}) \tag{28.107}$$

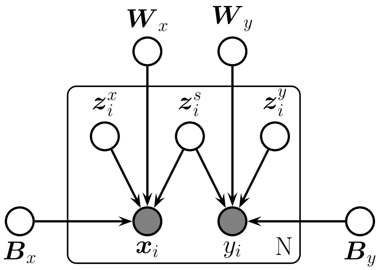


Figure 28.16: Canonical correlation analysis as a PGM.

See Figure 28.15b. The corresponding induced distribution on the visible variables has the form

$$p(\boldsymbol{v}_i|\boldsymbol{\theta}) = \int \mathcal{N}(\boldsymbol{v}_i|\mathbf{W}\boldsymbol{z}_i + \boldsymbol{\mu}, \sigma^2\mathbf{I})\mathcal{N}(\boldsymbol{z}_i|\mathbf{0}, \mathbf{I})d\boldsymbol{z}_i = \mathcal{N}(\boldsymbol{v}_i|\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^\mathsf{T} + \sigma^2\mathbf{I}) \tag{28.108}$$

where $\boldsymbol{v}_i = (\boldsymbol{y}_i; \boldsymbol{x}_i)$, $\boldsymbol{\mu} = (\boldsymbol{\mu}_y; \boldsymbol{\mu}_x)$ and

$$\mathbf{W} = \begin{pmatrix} \mathbf{W}_y & \mathbf{0} \\ \mathbf{W}_x & \mathbf{B}_x \end{pmatrix} \tag{28.109}$$

$$\mathbf{W}\mathbf{W}^\mathsf{T} = \begin{pmatrix} \mathbf{W}_y\mathbf{W}_y^\mathsf{T} & \mathbf{W}_y\mathbf{W}_x^\mathsf{T} \\ \mathbf{W}_x\mathbf{W}_y^\mathsf{T} & \mathbf{W}_x\mathbf{W}_x^\mathsf{T} + \mathbf{B}_x\mathbf{B}_x^\mathsf{T} \end{pmatrix} \tag{28.110}$$

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].

28.3.4.3 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, $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$, then we have three latent variables, $\boldsymbol{z}_i^s \in \mathbb{R}^{L_s}$ which is shared, $\boldsymbol{z}_i^x \in \mathbb{R}^{L_x}$ and $\boldsymbol{z}_i^y \in \mathbb{R}^{L_y}$ which are private. We can write the model as follows [BJ05]:

$$p(\boldsymbol{z}_i) = \mathcal{N}(\boldsymbol{z}_i^s|\mathbf{0}, \mathbf{I}_{L_s})\mathcal{N}(\boldsymbol{z}_i^x|\mathbf{0}, \mathbf{I}_{L_x})\mathcal{N}(\boldsymbol{z}_i^y|\mathbf{0}, \mathbf{I}_{L_y}) \tag{28.111}$$

$$p(\boldsymbol{x}_i|\boldsymbol{z}_i) = \mathcal{N}(\boldsymbol{x}_i|\mathbf{B}_x\boldsymbol{z}_i^x + \mathbf{W}_x\boldsymbol{z}_i^s + \boldsymbol{\mu}_x, \sigma^2\mathbf{I}_{D_x}) \tag{28.112}$$

$$p(\boldsymbol{y}_i|\boldsymbol{z}_i) = \mathcal{N}(\boldsymbol{y}_i|\mathbf{B}_y\boldsymbol{z}_i^y + \mathbf{W}_y\boldsymbol{z}_i^s + \boldsymbol{\mu}_y, \sigma^2\mathbf{I}_{D_y}) \tag{28.113}$$

See Figure 28.16 The corresponding observed joint distribution has the form

$$p(\boldsymbol{v}_i|\boldsymbol{\theta}) = \int \mathcal{N}(\boldsymbol{v}_i|\mathbf{W}\boldsymbol{z}_i + \boldsymbol{\mu}, \sigma^2\mathbf{I})\mathcal{N}(\boldsymbol{z}_i|\mathbf{0}, \mathbf{I})d\boldsymbol{z}_i = \mathcal{N}(\boldsymbol{v}_i|\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^\mathsf{T} + \sigma^2\mathbf{I}_D) \tag{28.114}$$

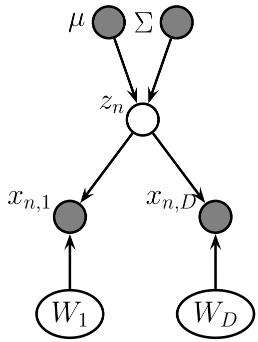


Figure 28.17: Exponential family PCA model as a DPGM.

where

$$\mathbf{W} = \begin{pmatrix} \mathbf{W}_x & \mathbf{B}_x & \mathbf{0} \\ \mathbf{W}_y & \mathbf{0} & \mathbf{B}_y \end{pmatrix} \tag{28.115}$$

$$\mathbf{W}\mathbf{W}^\mathsf{T} = \begin{pmatrix} \mathbf{W}_x\mathbf{W}_x^\mathsf{T} + \mathbf{B}_x\mathbf{B}_x^\mathsf{T} & \mathbf{W}_x\mathbf{W}_y^\mathsf{T} \\ \mathbf{W}_y\mathbf{W}_x^\mathsf{T} & \mathbf{W}_y\mathbf{W}_y^\mathsf{T} + \mathbf{B}_y\mathbf{B}_y^\mathsf{T} \end{pmatrix} \tag{28.116}$$

[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, and Section 32.2.2.2 for more details.

28.3.5 Factor analysis with exponential family likelihoods

So far we have assumed the observed data is real-valued, so $\boldsymbol{x}_n \in \mathbb{R}^D$. 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 $\boldsymbol{z}_n$. That is, we use

$$p(\boldsymbol{x}_n|\boldsymbol{z}_n) = \exp(\mathcal{T}(\boldsymbol{x})^\mathsf{T}\boldsymbol{\theta} + h(\boldsymbol{x}) - g(\boldsymbol{\theta})) \tag{28.117}$$

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 $p(\boldsymbol{z}_n|\boldsymbol{x}_n, \mathbf{W})$ 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 $\boldsymbol{z}_n$ and $\mathbf{W}$. This can be regarded as a degenerate version of variational EM, where the E step uses a delta function posterior for $\boldsymbol{z}_n$. [GS08] present an improved algorithm that finds the global optimum, and [Ude+16] presents an extension called **generalized low rank models**, that covers many different kinds of loss function.

However, it is often preferable to use a probabilistic version of the model, rather than computing point estimates of the latent factors. In this case, we must represent the posterior using a non-degenerate distribution to avoid overfitting, since the number of latent variables is proportional to the number of data cases [WCS08]. Fortunately, we can use a non-degenerate posterior, such as a Gaussian, by optimizing the variational lower bound. We give some examples of this below.

28.3.5.1 Example: binary PCA

Consider a factored Bernoulli likelihood:

$$p(\boldsymbol{x}|\boldsymbol{z}) = \prod_d \text{Ber}(x_d|\sigma(\boldsymbol{w}_d^\mathsf{T}\boldsymbol{z})) \tag{28.118}$$

Suppose we observe $N = 150$ bit vectors of length $D = 16$. Each example is generated by choosing one of three binary prototype vectors, and then by flipping bits at random. See Figure 28.18(a) for the data. We can fit this using the variational EM algorithm (see [Tip98] for details). We use $L = 2$ latent dimensions to allow us to visualize the latent space. In Figure 28.18(b), we plot $\mathbb{E}\left[\boldsymbol{z}_n|\boldsymbol{x}_n, \hat{\mathbf{W}}\right]$. We see that the projected points group into three distinct clusters, as is to be expected. In Figure 28.18(c), we plot the reconstructed version of the data, which is computed as follows:

$$p(\hat{x}_{nd} = 1|\boldsymbol{x}_n) = \int d\boldsymbol{z}_n \; p(\boldsymbol{z}_n|\boldsymbol{x}_n)p(\hat{x}_{nd}|\boldsymbol{z}_n) \tag{28.119}$$

If we threshold these probabilities at 0.5 (corresponding to a MAP estimate), we get the "denoised" version of the data in Figure 28.18(d).

28.3.5.2 Example: categorical PCA

We can generalize the model in Section 28.3.5.1 to handle categorical data by using the following likelihood:

$$p(\boldsymbol{x}|\boldsymbol{z}) = \prod_d \text{Cat}(x_d|\text{softmax}(\mathbf{W}_d\boldsymbol{z})) \tag{28.120}$$

We call this **categorical PCA** (**CatPCA**). A variational EM algorithm for fitting this is described in [Kha+10].

28.3.6 Factor analysis with DNN likelihoods (VAEs)

The FA model 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 let the mapping from $\boldsymbol{z}$ to $\boldsymbol{x}$ be a nonlinear model, such as a neural network. That is, the likelihood becomes

$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|f(\boldsymbol{w};\boldsymbol{\theta}), \sigma^2\mathbf{I}) \tag{28.121}$$

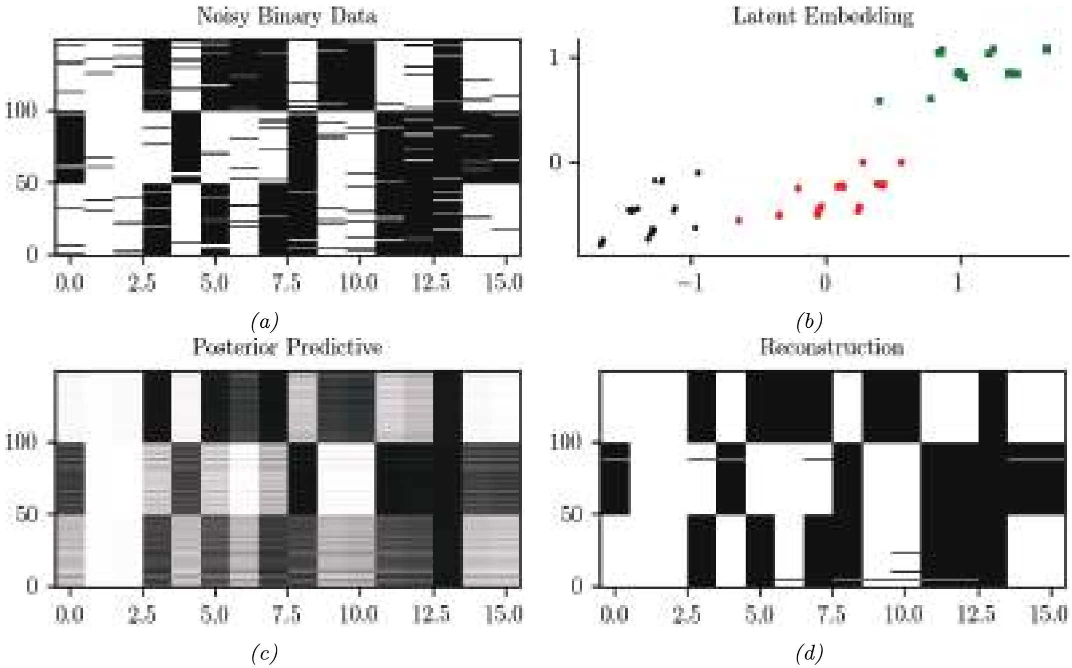


Figure 28.18: (a) 150 synthetic 16 dimensional bit vectors. (b) The 2d embedding learned by binary PCA, fit using variational EM. We have color coded points by the identity of the true "prototype" that generated them. (c) Predicted probability of being on. (d) Thresholded predictions. Generated by binary_fa_demo.ipynb.

We call this "**nonlinear factor analysis**". (We can of course replace the Gaussian likelihood with other distributions, such as categorical, in which case we get nonlinear exponential family factor analysis.) Unfortunately we can no longer compute the posterior or the MLE exactly, so we need to use approximate methods. In Chapter 21, we discuss the variational autoencoder, which fits this nonlinear FA 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]).

28.3.7 Factor analysis with GP likelihoods (GP-LVM)

In this section we discuss a nonlinear version of factor analysis in which we replace the linear decoder $f(\boldsymbol{z}) = \mathbf{W}\boldsymbol{z}$ used in the likelihood $p(\boldsymbol{y}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{y}|f(\boldsymbol{z}), \sigma^2\mathbf{I})$ with a nonlinear function, represented by a Gaussian process (Chapter 18), one per output dimension. This is known as a **GP-LVM**, which stands for "Gaussian process latent variable model" [Law05]. (Note that we switch notation a bit from standard FA and define the observed output variable by $\boldsymbol{y}$, to be consistent with standard supervised GP notation; the inputs to the GP will be latent variables $\boldsymbol{z}$.)

To explain the method in more detail, we start with PPCA (Section 28.3.2). Recall that the PPCA model is as follows:

$$p(\boldsymbol{z}_i) = \mathcal{N}(\boldsymbol{z}_i|\mathbf{0}, \mathbf{I}) \tag{28.122}$$

$$p(\boldsymbol{y}_i|\boldsymbol{z}_i, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}_i|\mathbf{W}\boldsymbol{z}_i, \sigma^2\mathbf{I}) \tag{28.123}$$

We can fit this model by maximum likelihood, by integrating out the $\boldsymbol{z}_i$ and maximizing wrt $\mathbf{W}$ (and σ^2). The objective is given by

$$p(\mathbf{Y}|\mathbf{W}, \sigma^2) = (2\pi)^{-DN/2}|\mathbf{C}|^{-N/2} \exp\left(-\frac{1}{2}\text{tr}(\mathbf{C}^{-1}\mathbf{Y}^\mathsf{T}\mathbf{Y})\right) \tag{28.124}$$

where $\mathbf{C} = \mathbf{W}\mathbf{W}^\mathsf{T} + \sigma^2\mathbf{I}$. As we showed in Section 28.3.2, the MLE for $\mathbf{W}$ can be computed in terms of the eigenvectors of $\mathbf{Y}^\mathsf{T}\mathbf{Y}$.

Now we consider the dual problem, whereby we maximize wrt $\mathbf{Z}$ and integrate out $\mathbf{W}$. We will use a prior of the form $p(\mathbf{W}) = \prod_j \mathcal{N}(\boldsymbol{w}_j|\mathbf{0}, \mathbf{I})$. The corresponding likelihood becomes

$$p(\mathbf{Y}|\mathbf{Z}, \sigma^2) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{Y}_{:,d}|\mathbf{0}, \mathbf{Z}\mathbf{Z}^\mathsf{T} + \sigma^2\mathbf{I}) \tag{28.125}$$

$$= (2\pi)^{-DN/2}|\mathbf{K}_z|^{-D/2} \exp\left(-\frac{1}{2}\text{tr}(\mathbf{K}_\sigma^{-1}\mathbf{Y}\mathbf{Y}^\mathsf{T})\right) \tag{28.126}$$

where $\mathbf{K}_\sigma = \mathbf{K} + \sigma^2\mathbf{I}$, and $\mathbf{K} = \mathbf{Z}\mathbf{Z}^\mathsf{T}$. The MLE for $\mathbf{Z}$ can be computed in terms of the eigenvectors of $\mathbf{K}_\sigma$, and gives the same results as PPCA (see [Law05] for the details).

To understand what this process is doing, consider modeling the prior on $f : \mathcal{Z} \to \mathcal{Y}$ with a GP with a linear kernel:

$$\mathcal{K}(\boldsymbol{z}_i, \boldsymbol{z}_j) = \boldsymbol{z}_i^\mathsf{T}\boldsymbol{z}_j + \sigma^2\delta_{ij} \tag{28.127}$$

The corresponding covariance matrix has the form $\mathbf{K} = \mathbf{Z}\mathbf{Z}^\mathsf{T} + \sigma^2\mathbf{I}$. Thus Equation (28.126) is equivalent to the likelihood of a product of independent GPs. Just as factor analysis is like linear regression with unknown inputs, so GP-LVM is like GP regression with unknown inputs. The goal is then to compute a point estimate of these unknown inputs, i.e., $\hat{\mathbf{Z}}$. (We can also use Bayesian inference.)

The advantage of the dual formulation is that we can use a more general kernel for $\mathbf{K}$ instead of the linear kernel. That is, we can set $K_{ij} = \mathcal{K}(\boldsymbol{z}_i, \boldsymbol{z}_j)$ for any Mercer kernel. The MLE for $\mathbf{Z}$ is no longer be available via eigenvalue methods, but can be computed using gradient-based optimization.

In Figure 28.19, 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 datapoint corresponds to a different body pose. When projected to 2d, we see that similar poses are clustered nearby.

28.4 LFMs with non-Gaussian priors

In this section, we discuss (linear) latent factor models with non-Gaussian priors. See Table 28.1 for a summary of the models we will discuss.

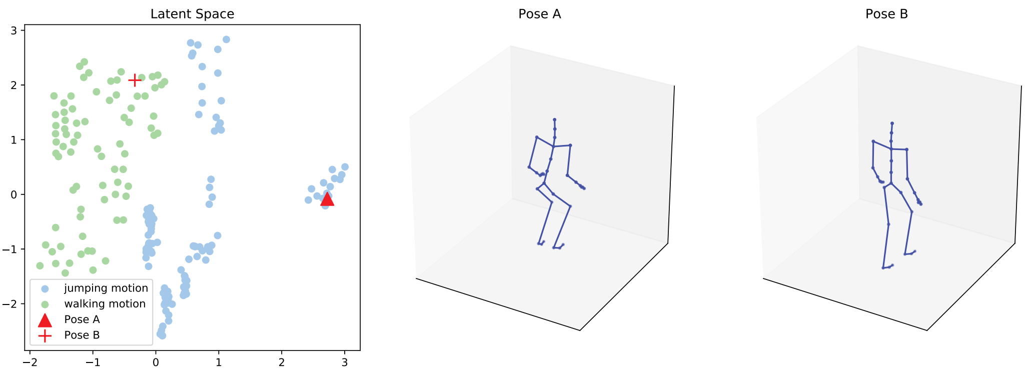
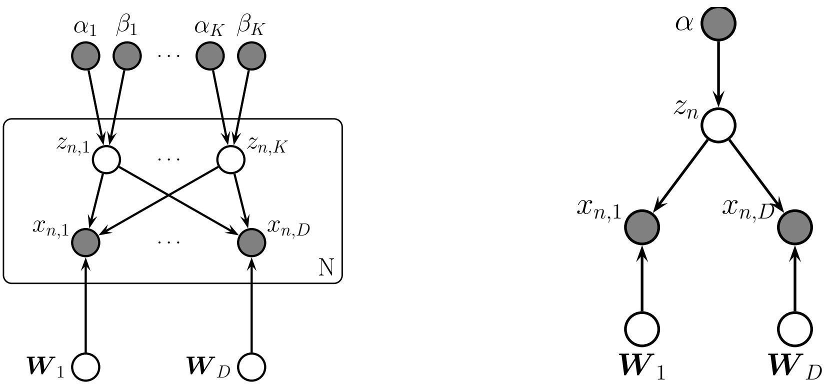


Figure 28.19: Illustration of a 2d embedding of human motion-capture data using a GP-LVM. We show two poses and their corresponding embeddings. Generated by gplvm_mocap.ipynb. Used with kind permission of Aditya Ravuri.



(a) (b)

Figure 28.20: (a) Gaussian-Poisson (GAP) model as a DPGM. Here $z_{n,k} \in \mathbb{R}^+$ and $x_{n,d} \in \mathbb{Z}_{\geq 0}$. (b) Simplex FA model as a DPGM. Here $\boldsymbol{z}_n \in \mathbb{S}_K$ and $x_{n,d} \in \{1, \ldots, V\}$.

28.4.1 Non-negative matrix factorization (NMF)

Suppose that we use a gamma distribution for the latents: $p(\boldsymbol{z}) = \prod_k \text{Ga}(z_k|\alpha_k, \beta_k)$. This results in a sparse, non-negative hidden representation, which can help interpretability. This is particularly useful when the data is also sparse and non-negative, such as word counts. In this case, it makes sense to use a Poisson likelihood: $p(\boldsymbol{x}|\boldsymbol{z}) = \prod_{d=1}^{D} \text{Poi}(x_d|\boldsymbol{w}_d^\mathsf{T}\boldsymbol{z})$. The overall model has the form

$$p(\boldsymbol{z}, \boldsymbol{x}) = p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z}) = \left[\prod_k \text{Ga}(z_k|\alpha_k, \beta_k)\right] \left[\prod_{d=1}^{D} \text{Poi}(x_d|\boldsymbol{w}_d^\mathsf{T}\boldsymbol{z})\right] \tag{28.128}$$

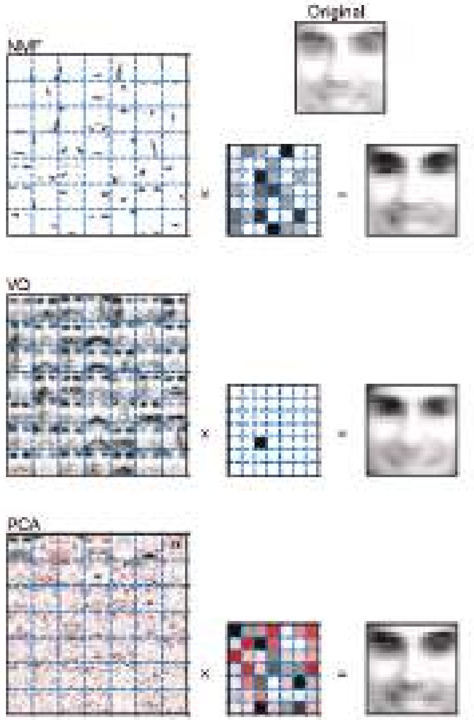
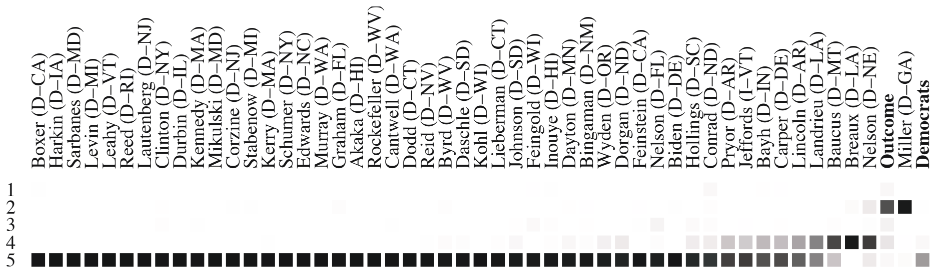


Figure 28.21: Illustrating the difference between non-negative matrix factorization (NMF), vector quantization (VQ), and principal components analysis (PCA). Left column: filters (columns of $\mathbf{W}$*) learned from a set of 2429 faces images, each of size* 19×19*. There are 49 basis functions in total, shown in a* 7×7 *montage; each filter is reshaped to a* 19×19 *image for display purposes. (For PCA, negative weights are red, positive weights are black.) Middle column: the 49 latent factors* $\boldsymbol{z}$ *when the model is applied to the original face image shown at the top. Right column: reconstructed face image. From Figure 1 of [LS99].*

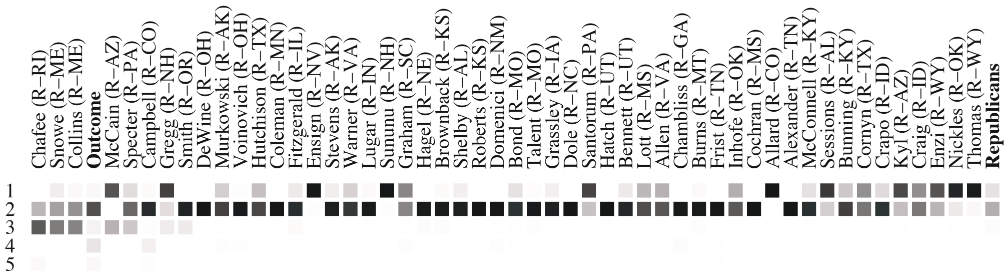
The resulting model is called the **GaP** (gamma-Poisson) model [Can04]. See Figure 28.20a for the graphical model.

The parameters α_k and β_k control the sparsity of the latent representation $\boldsymbol{z}_n$. If we set $\alpha_k = \beta_k = 0$, and compute the MLE for $\mathbf{W}$, we recover **non-negative matrix factorization** (**NMF**) [PT94; LS99; LS01], as shown in [BJ06].

Figure 28.21 illustrates the result of applying NMF to a dataset of image patches of faces, where the data correspond to non-negative pixel intensities. We see that the learned basis functions are small localized **parts** of faces. Also, the coefficient vector $\boldsymbol{z}$ is sparse and positive. For PCA, the coefficient vector has negative values, and the resulting basis functions are global, not local. For vector quantization (i.e., GMM model), $\boldsymbol{z}$ 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.



(a) Democrats.



(b) Republicans.

Figure 28.22: 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.

28.4.2 Multinomial PCA

Suppose we use a Dirichlet prior for the latents, $p(\boldsymbol{z}) = \mathrm{Dir}(\boldsymbol{z}|\boldsymbol{\alpha})$, so $\boldsymbol{z} \in \mathbb{S}_K$, which is the K-dimensional probability simplex. As in Section 28.4.1, the vector $\boldsymbol{z}$ will be sparse and non-negative, but in addition it will satsify the constraint $\sum_{k=1}^{K} z_k = 1$, so the components are not independent. Now suppose our data is categorical, $x_d \in \{1, \ldots, V\}$, so our likelihood has the form $p(\boldsymbol{x}|\boldsymbol{z}) = \prod_d \mathrm{Cat}(x_d|\mathbf{W}_d\boldsymbol{z})$. The overall model is therefore

$$p(\boldsymbol{z}, \boldsymbol{x}) = \mathrm{Dir}(\boldsymbol{z}|\boldsymbol{\alpha}) \prod_{d=1}^{D} \mathrm{Cat}(x_d|\mathbf{W}_d\boldsymbol{z}) \tag{28.129}$$

See Figure 28.20b for the DPGM. 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].

28.4.2.1 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 28.22 plots $\mathbb{E}[z_{nk}|\boldsymbol{x}_n] \in [0,1]$, which is the degree to which senator n belongs to latent component or "bloc" k. We see that component 5 is the Democractic majority, and block 2 is the Republican majority. See [BJ06] for further details.

28.4.2.2 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 $\boldsymbol{\mu}_{nd} = \mathbf{W}^d \boldsymbol{z}_n$, and hence

$$p(x_{nd} = v|\boldsymbol{z}_n) = \sum_k z_{nk} w_{vk}^d \tag{28.130}$$

Thus the latent variables can be additively combined to compute the mean parameters, aiding interpretability. By contrast, the CatPCA model in Section 28.3.5.2 uses a Gaussian prior, so $\mathbf{W}^d \boldsymbol{z}_n$ can be negative; consequently it must pass this vector through a softmax, to convert from natural parameters to mean parameters; this makes $\boldsymbol{z}_n$ harder to interpret.

28.4.2.3 Connection to mixture models

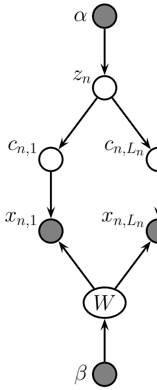
If $\boldsymbol{z}_n$ were a one-hot vector, rather than any point in the probability simplex, then the mPCA model would be equivalent to selecting a single column from $\mathbf{W}_d$ corresponding to the discrete hidden state. This is equivalent to a finite mixture of categorical distributions (see Section 28.2.2), and corresponds to the assumption that $\boldsymbol{x}$ is generated by a single cluster. However, the mPCA model does not require that $\boldsymbol{z}_n$ be one-hot, and instead allows $\boldsymbol{x}_n$ to partially belong to multiple clusters. For this reason, this model is also known as an **admixture mixture** or **mixed membership model** [EFL04].

28.5 Topic models

In this section, we show how to modify the multinomial PCA model of Section 28.4.2 to create latent variable models for sequences of discrete tokens, such as words in text documents, or genes in a DNA sequence. The basic idea is to assume that the words are conditionally independent given a latent **topic vector** $\boldsymbol{z}$. Rather than being a single discrete cluster label, $\boldsymbol{z}$ is a probability distribution over clusters, and each word is sampled from its own "local" cluster. In the NLP community, this kind of model is called a **topic model** (see e.g., [BGHM17]).

28.5.1 Latent Dirichlet allocation (LDA)

In this section, we discuss the most common kind of topic model known as **latent Dirichlet allocation** or **LDA** [BNJ03a; Ble12]. (This usage of the term "LDA" is not to be confused with linear discriminant analysis.) In the genetic community, this model is known as an **admixture model** [PSD00].



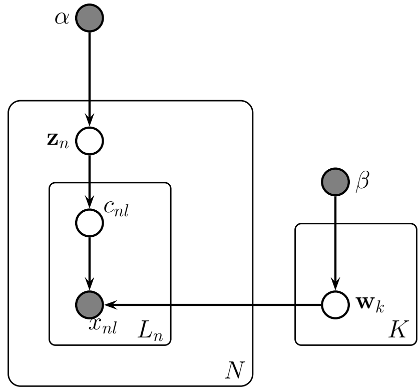


Figure 28.23: Latent Dirichlet allocation (LDA) as a DPGM. (a) Unrolled form. (b) Plate form.

28.5.1.1 Model definition

We can define the LDA model as follows. Let $x_{nl} \in \{1, \ldots, V\}$ be the identity of the l'th word in document n, where l can now range from 1 to L_n, the length of the document, and V is the size of the vocabulary. The probability of word v at location l is given by

$$p(x_{nl} = v|\boldsymbol{z}_n) = \sum_k z_{nk} w_{kv} \tag{28.131}$$

where $0 \leq z_{nk} \leq 1$ is the proportion of "topic" k in document n, and $\boldsymbol{z}_n \sim \text{Dir}(\boldsymbol{\alpha})$.

We can rewrite this model by associating a discrete latent variable $m_{nl} \in \{1, \ldots, N_z\}$ with each word in each document, with distribution $p(m_{nl}|\boldsymbol{z}_n) = \text{Cat}(m_{nl}|\boldsymbol{z}_n)$. Thus m_{nl} specifies the topic to use for word l in document n. The full joint model becomes

$$p(\boldsymbol{x}_n, \boldsymbol{z}_n, \boldsymbol{m}_n) = \text{Dir}(\boldsymbol{z}_n|\boldsymbol{\alpha}) \prod_{l=1}^{L_n} \text{Cat}(m_{nl}|\boldsymbol{z}_n)\text{Cat}(x_{nl}|\mathbf{W}[m_{nl}, :]) \tag{28.132}$$

where $\mathbf{W}[k, :] = \boldsymbol{w}_k$ is the distribution over words for the k'th topic. See Figure 28.23 for the corresponding DPGM.

We typically use a Dirichlet prior the topic parameters, $p(\boldsymbol{w}_k) = \text{Dir}(\boldsymbol{w}_k|\beta \mathbf{1}_V)$; by setting β small enough, we can encourage these topics to be sparse, so that each topic only predicts a subset of the words. In addition, we use a Dirichlet prior on the latent factors, $p(\boldsymbol{z}_n) = \text{Dir}(\boldsymbol{z}_n|\alpha \mathbf{1}_{N_z})$. If we set α small enough, we can encourage the topic distribution for each document to be sparse, so that each document only contains a subset of the topics. See Figure 28.24 for an illustration.

Note that an earlier version of LDA, known as **probabilistic LSA**, was proposed in [Hof99]. (LSA stands for "**latent semantic analysis**", and refers to the application of PCA to text data; see [Mur22, Sec 20.5.1.2] for details.) The likelihood function, $p(\boldsymbol{x}|\boldsymbol{z})$, is the same as in LDA, but pLSA does not specify a prior for $\boldsymbol{z}$, since it is designed for posterior analysis of a fixed corpus (similar to LSA), rather than being a true generative model.

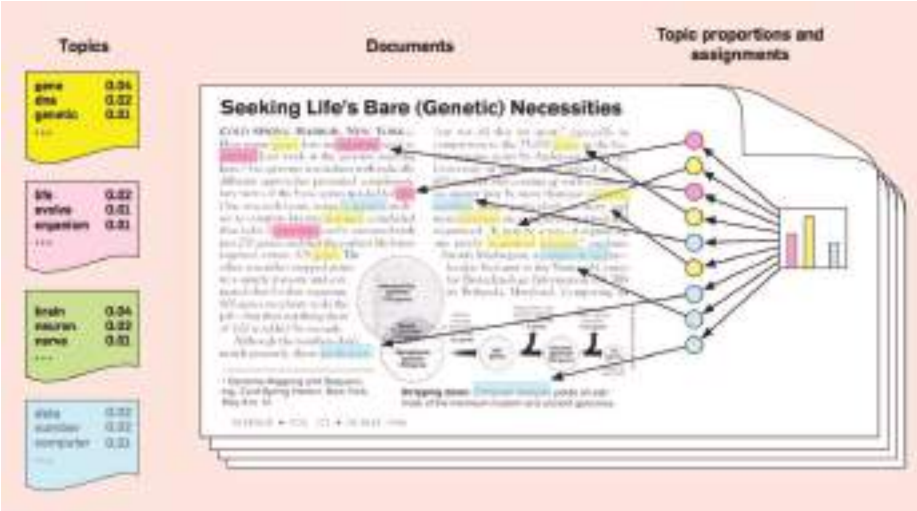


Figure 28.24: Illustration of latent Dirichlet allocation (LDA). We have color coded certain words by the topic they have been assigned to: yellow represents the genetics cluster, pink represents the evolution cluster, blue represent the data analysis cluster, and green represents the neuroscience cluster. Each topic is in turn defined as a sparse distribution over words. This article is not related to neuroscience, so no words are assigned to the green topic. The overall distribution over topic assignments for this document is shown in the right as a sparse histogram. Adapted from Figure 1 of [Ble12]. Used with kind permission of David Blei.

28.5.1.2 Polysemy

Each topic is a distribution over words that co-occur, and which are therefore semantically related. For example, Figure 28.25 shows 3 topics which were learned from an LDA model fit to the **TASA corpus**[3]. These seem to correspond to 3 different senses of the word "play": playing an instrument, a theatrical play, and playing a sports game.

We can use the inferred document-level topic distribution to overcome **polysemy**, i.e., to disambiguate the meaning of a particular word. This is illustrated in Figure 28.26, where a subset of the words are annotated with the topic to which they were assigned (i.e., we show $\text{argmax}_k \, p(m_{nl} = k|\boldsymbol{x}_n)$). In the first document, the word "music" makes it clear that the musical topic (number 77) is present in the document, which in turn makes it more likely that $m_{nl} = 77$ where l is the index corresponding to the word "play".

3. The TASA corpus is an untagged collection of educational materials consisting of 37,651 documents and 12,190,931 word tokens. Words appearing in fewer than 5 documents were replaced with an asterisk, but punctuation was included. The combined vocabulary was of size 37,202 unique words.

Topic 77

word	prob.
MUSIC	.090
DANCE	.034
SONG	.033
PLAY	.030
SING	.026
SINGING	.026
BAND	.026
PLAYED	.023
SANG	.022
SONGS	.021
DANCING	.020
PIANO	.017
PLAYING	.016
RHYTHM	.015
ALBERT	.013
MUSICAL	.013

Topic 82

word	prob.
LITERATURE	.031
POEM	.028
POETRY	.027
POET	.020
PLAYS	.019
POEMS	.019
PLAY	.015
LITERARY	.013
WRITERS	.013
DRAMA	.012
WROTE	.012
POETS	.011
WRITER	.011
SHAKESPEARE	.010
WRITTEN	.009
STAGE	.009

Topic 166

word	prob.
PLAY	.136
BALL	.129
GAME	.065
PLAYING	.042
HIT	.032
PLAYED	.031
BASEBALL	.027
GAMES	.025
BAT	.019
RUN	.019
THROW	.016
BALLS	.015
TENNIS	.011
HOME	.010
CATCH	.010
FIELD	.010

Figure 28.25: Three topics related to the word play. *From Figure 9 of [SG07]. Used with kind permission of Tom Griffiths.*

Document #29795

Bix beiderbecke, at age^{060} fifteen^{207}, sat^{174} on the slope^{071} of a bluff^{055} overlooking^{027} the mississippi^{137} river^{137}. He was listening^{077} to music^{077} coming^{009} from a passing^{043} riverboat. The music^{077} had already captured^{006} his heart^{157} as well as his ear^{119}. It was jazz^{077}. Bix beiderbecke had already had music^{077} lessons^{077}. He showed^{002} promise^{134} on the piano^{077}, and his parents^{035} hoped^{268} he might consider^{118} becoming a concert^{077} pianist^{077}. But bix was interested^{268} in another kind^{050} of music^{077}. He wanted^{268} to $\boxed{\text{play}^{077}}$ the cornet. And he wanted^{268} to $\boxed{\text{play}^{077}}$ jazz^{077}...

Document #1883

There is a simple^{050} reason^{106} why there are so few periods^{078} of really great theater^{082} in our whole western^{046} world. Too many things^{300} have to come right at the very same time. The dramatists must have the right actors^{082}, the actors^{082} must have the right playhouses, the playhouses must have the right audiences^{082}. We must remember^{288} that plays^{082} exist^{143} to be performed^{077}, not merely^{050} to be read^{254}. (even when you read^{254} a $\boxed{\text{play}^{082}}$ to yourself, try^{288} to perform^{062} it, to put^{174} it on a stage^{078}, as you go along.) as soon^{028} as a $\boxed{\text{play}^{082}}$ has to be performed^{082}, then some kind^{126} of theatrical^{082}...

Document #21359

Jim^{296} has a game^{166} book^{254}. Jim^{296} reads^{254} the book^{254}. Jim^{296} sees^{081} a game^{166} for one. Jim^{296} plays^{166} the game^{166}. Jim^{296} likes^{081} the game^{166} for one. The game^{166} book^{254} helps^{081} jim^{296}. Don^{180} comes^{040} into the house^{038}. Don^{180} and jim^{296} read^{254} the game^{166} book^{254}. The boys^{020} see a game^{166} for two. The two boys^{020} $\boxed{\text{play}^{166}}$ the game^{166}. The boys^{020} $\boxed{\text{play}^{166}}$ the game^{166} for two. The boys^{020} like the game^{166}. Meg^{282} comes^{040} into the house^{282}. Meg^{282} and don^{180} and jim^{296} read^{254} the book^{254}. They see a game^{166} for three. Meg^{282} and don^{180} and jim^{296} $\boxed{\text{play}^{166}}$ the game^{166}. They $\boxed{\text{play}^{166}}$...

Figure 28.26: Three documents from the TASA corpus containing different senses of the word play. *Grayed out words were ignored by the model, because they correspond to uninteresting stop words (such as "and", "the", etc.) or very low frequency words. From Figure 10 of [SG07]. Used with kind permission of Tom Griffiths.*

28.5.1.3 Posterior inference

Many algorithms have been proposed to perform approximate posterior inference in the LDA model. In the original LDA paper, [BNJ03a], they use variational mean field inference (see Section 10.3). In [HBB10], they use stochastic VI (see Supplementary Section 28.1.2). In [GS04], they use collapsed Gibbs sampling, which marginalizes out the discrete latents (see Supplementary Section 28.1.1). In [MB16; SS17b] they discuss how to learned amortized inference networks to perform VI for the collapsed model.

Recently, there has been considerable interest in spectral methods for fitting LDA-like models which are fast and which come with provable guarantees about the quality of the solution they obtain (unlike MCMC and variational methods, where the solution is just an approximation of unknown quality). These methods make certain (reasonable) assumptions beyond the basic model, such as the existence of some anchor words, which uniquely the topic for a document. See [Aro+13] for details.

28.5.1.4 Determining the number of topics

Choosing N_z, the number of topics, is a standard model selection problem. Here are some approaches that have been taken:

- Use annealed importance sampling (Section 11.5.4) to approximate the evidence [Wal+09].
- Cross validation, using the log likelihood on a test set.
- Use the variational lower bound as a proxy for $\log p(\mathcal{D}|N_z)$.
- Use non-parametric Bayesian methods [Teh+06].

28.5.2 Correlated topic model

One weakness of LDA is that it cannot capture correlation between topics. For example, if a document has the "business" topic, it is reasonable to expect the "finance" topic to co-occcur. The source of the problem is the use of a Dirichlet prior for $\boldsymbol{z}_n$. The problem with the Dirichlet it that it is characterized by just a mean vector $\boldsymbol{\alpha}$, but its covariance is fixed ($\Sigma_{ij} = -\alpha_i\alpha_j$), rather than being a free parameter.

One way around this is to replace the Dirichlet prior with the **logistic normal** distribution, which is defined as follows:

$$p(\boldsymbol{z}) = \int \text{Cat}(\boldsymbol{z}|\text{softmax}(\boldsymbol{\epsilon}))\mathcal{N}(\boldsymbol{\epsilon}|\boldsymbol{\mu}, \boldsymbol{\Sigma})d\boldsymbol{\epsilon} \tag{28.133}$$

This is known as the **correlated topic model** [BL07b];

The difference from categorical PCA discussed in Section 28.3.5.2 is that CTM uses a logistic normal to model the mean parameters, so $\boldsymbol{z}_n$ is sparse and non-negative, whereas CatPCA uses a normal to model the natural parameters, so $\boldsymbol{z}_n$ is dense and can be negative. More precisely, the CTM defines $x_{nl} \sim \text{Cat}(\mathbf{W}\text{softmax}(\boldsymbol{\epsilon}_n))$, but CatPCA defines $\boldsymbol{x}_{nd} \sim \text{Cat}(\text{softmax}(\mathbf{W}_d\boldsymbol{z}_n))$.

Fitting the CTM model is tricky, since the prior for $\boldsymbol{\epsilon}_n$ is no longer conjugate to the multinomial likelihood for m_{nl}. However, we can derive a variational mean field approximation, as described in [BL07b].

Having fit the model, one can then convert $\hat{\boldsymbol{\Sigma}}$ to a sparse precision matrix $\hat{\boldsymbol{\Sigma}}^{-1}$ by pruning low-strength edges, to get a sparse Gaussian graphical model. This allows you to visualize the correlation between topics. Figure 28.27 shows the result of applying this procedure to articles from *Science* magazine, from 1990–1999.

28.5.3 Dynamic topic model

In LDA, the topics (distributions over words) are assumed to be static. In some cases, it makes sense to allow these distributions to evolve smoothly over time. For example, an article might use the topic "neuroscience", but if it was written in the 1900s, it is more likely to use words like "nerve", whereas if it was written in the 2000s, it is more likely to use words like "calcium receptor" (this reflects the general trend of neuroscience towards molecular biology).

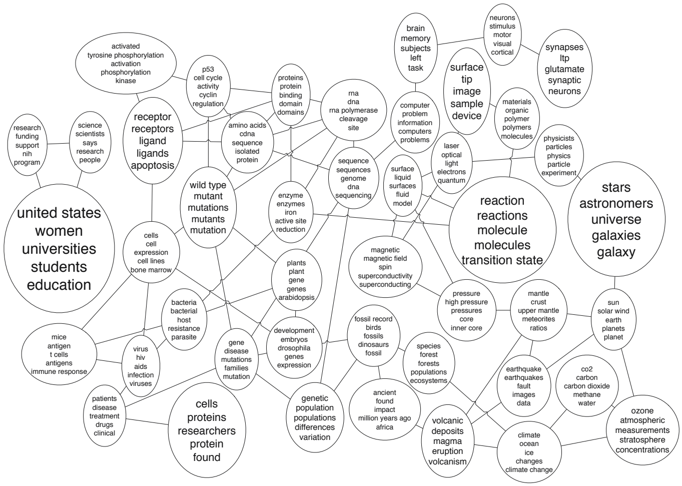


Figure 28.27: Output of the correlated topic model (with $K = 50$ topics) when applied to articles from Science. *Nodes represent topics, with the 5 most probable phrases from each topic shown inside. Font size reflects overall prevalence of the topic. See* `http://www.cs.cmu.edu/~lemur/science/` *for an interactive version of this model with 100 topics. Used with kind permission of Figure 2 of [BL07b]. Used with kind permission of David Blei.*

One way to model this is to assume the topic distributions evolve according to a Gaussian random walk, as in a state space model (see Section 29.1). We can map these Gaussian vectors to probabilities via the softmax function, resulting in the following model:

$$\boldsymbol{w}_k^t | \boldsymbol{w}_k^{t-1} \sim \mathcal{N}(\boldsymbol{w}_{t-1,k}, \sigma^2 \mathbf{1}_{N_w}) \tag{28.134}$$

$$\boldsymbol{z}_n^t \sim \text{Dir}(\alpha \mathbf{1}_{N_z}) \tag{28.135}$$

$$m_{nl}^t | \boldsymbol{z}_n^t \sim \text{Cat}(\boldsymbol{z}_n^t) \tag{28.136}$$

$$x_{nl}^t | m_{nl}^t = k, \mathbf{W}^t \sim \text{Cat}(\text{softmax}(\boldsymbol{w}_k^t)) \tag{28.137}$$

This is known as a **dynamic topic model** [BL06]. See Figure 28.28 for the DPGM.

One can perform approximate inference in this model using a structured mean field method (Section 10.4.1), that exploits the Kalman smoothing algorithm (Section 8.2.2) to perform exact inference on the linear-Gaussian chain between the $\boldsymbol{w}_k^t$ nodes (see [BL06] for details). Figure 28.29 illustrates a typical output of the system when applied to 100 years of articles from *Science.*

It is also possible to use amortized inference, and to learn embeddings for each word, which works much better with rare words. This is called the **dynamic embedded topic model** [DRB19].

28.5.4 LDA-HMM

The Latent dirichlet allocation (LDA) model of Section 28.5.1 assumes words are exchangeable, and thus ignores word order. A simple way to model sequential dependence between words is to use an HMM. The trouble with HMMs is that they can only model short-range dependencies, so they cannot

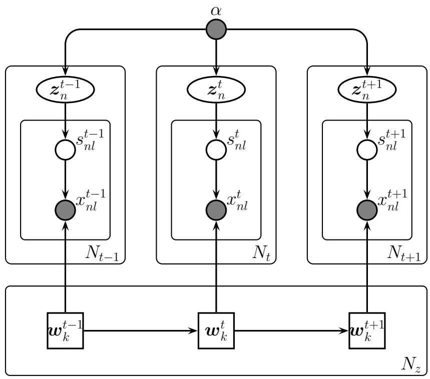


Figure 28.28: The dynamic topic model as a DPGM.

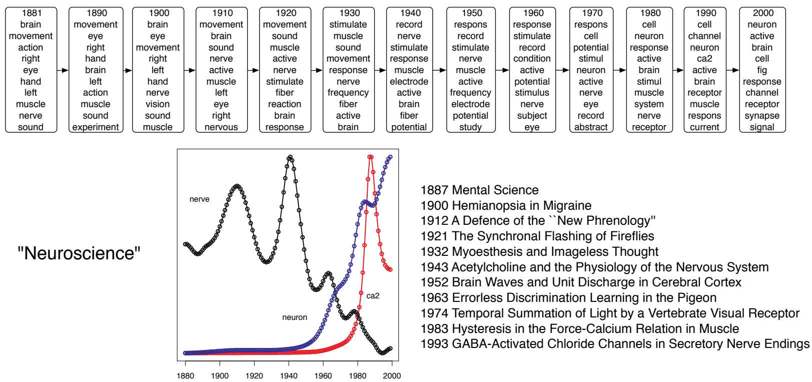


Figure 28.29: Part of the output of the dynamic topic model when applied to articles from Science*. At the top, we show the top 10 words for the neuroscience topic over time. On the bottom left, we show the probability of three words within this topic over time. On the bottom right, we list paper titles from different years that contained this topic. From Figure 4 of [BL06]. Used with kind permission of David Blei.*

capture the overall gist of a document. Hence they can generate syntactically correct sentences, but not semantically plausible ones.

It is possible to combine LDA with HMM to create a model called **LDA-HMM** [Gri+04]. This model uses the HMM states to model function or syntactic words, such as "and" or "however", and uses the LDA to model content or semantic words, which are harder to predict. There is a distinguished HMM state which specifies when the LDA model should be used to generate the word; the rest of the time, the HMM generates the word.

More formally, for each document n, the model defines an HMM with states $h_{nl} \in \{0, \dots, H\}$. In addition, each document has an LDA model associated with it. If $h_{nl} = 0$, we generate word x_{nl} from the semantic LDA model, with topic specified by m_{nl}; otherwise we generate word x_{nl} from the

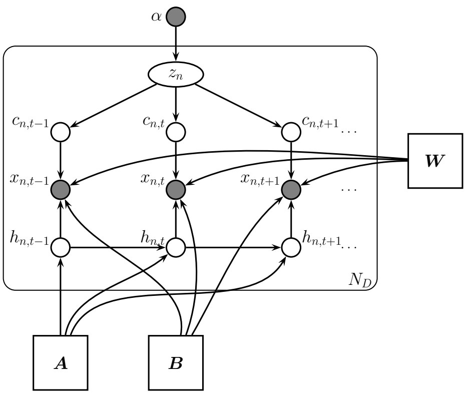


Figure 28.30: LDA-HMM model as a DPGM.

syntactic HMM model. The DPGM is shown in Figure 28.30. The CPDs are as follows:

$$p(\boldsymbol{z}_n) = \text{Dir}(\boldsymbol{z}_n|\alpha\mathbf{1}_{N_z}) \tag{28.138}$$

$$p(m_{nl} = k|\boldsymbol{z}_n) = z_{nk} \tag{28.139}$$

$$p(h_{n,l} = j|h_{n,l-1} = i) = A_{ij} \tag{28.140}$$

$$p(x_{nl} = d|m_{nl} = k, h_{nl} = j) = \begin{cases} W_{kd} & \text{if } j = 0 \\ B_{jd} & \text{if } j > 0 \end{cases} \tag{28.141}$$

where $\mathbf{W}$ is the usual topic-word matrix, $\mathbf{B}$ is the state-word HMM emission matrix, and $\mathbf{A}$ is the state-state HMM transition matrix.

Inference in this model can be done with collapsed Gibbs sampling, analytically integrating out all the continuous quantities. See [Gri+04] for the details.

The results of applying this model (with $N_z = 200$ LDA topics and $H = 20$ HMM states) to the combined Brown and TASA corpora[4] are shown in Table 28.2. We see that the HMM generally is responsible for syntactic words, and the LDA for semantics words. If we did not have the HMM, the LDA topics would get "polluted" by function words (see top of figure), which is why such words are normally removed during preprocessing.

The model can also help disambiguate when the same word is being used syntactically or semantically. Figure 28.31 shows some examples when the model was applied to the NIPS corpus.[5] We see that the roles of words are distinguished, e.g., "we require the algorithm to *return* a matrix" (verb) vs "the maximal expected *return*" (noun). In principle, a part of speech tagger could disambiguate these two uses, but note that (1) the LDA-HMM method is fully unsupervised (no POS tags were used), and (2) sometimes a word can have the same POS tag, but different senses, e.g., "the left graph" (a synactic role) vs "the graph G" (a semantic role).

4. The Brown corpus consists of 500 documents and 1,137,466 word tokens, with part-of-speech tags for each token. The TASA corpus is an untagged collection of educational materials consisting of 37,651 documents and 12,190,931 word tokens. Words appearing in fewer than 5 documents were replaced with an asterisk, but punctuation was included. The combined vocabulary was of size 37,202 unique words.
5. NIPS stands for "Neural Information Processing Systems". It is one of the top machine learning conferences. The NIPS corpus volumes 1–12 contains 1713 documents.

the	the	the	the	the	a	the	the	the
blood	,	,	of	a	the	,	,	,
,	and	and	,	of	of	of	a	a
of	of	of	to	,	,	a	of	in
body	a	in	in	in	in	and	and	game
heart	in	land	and	to	water	in	drink	ball
and	trees	to	classes	picture	is	story	alcohol	and
in	tree	farmers	government	film	and	is	to	team
to	with	for	a	image	matter	to	bottle	to
is	on	farm	state	lens	are	as	in	play
blood	forest	farmers	government	light	water	story	drugs	ball
heart	trees	land	state	eye	matter	stories	drug	game
pressure	forests	crops	federal	lens	molecules	poem	alcohol	team
body	land	farm	public	image	liquid	characters	people	*
lungs	soil	food	local	mirror	particles	poetry	drinking	baseball
oxygen	areas	people	act	eyes	gas	character	person	players
vessels	park	farming	states	glass	solid	author	effects	football
arteries	wildlife	wheat	national	object	substance	poems	marijuana	player
*	area	farms	laws	objects	temperature	life	body	field
breathing	rain	corn	department	lenses	changes	poet	use	basketball
the	in	he	*	be	said	can	time	,
a	for	it	new	have	made	would	way	;
his	to	you	other	see	used	will	years	(
this	on	they	first	make	came	could	day	:
their	with	i	same	do	went	may	part	)
these	at	she	great	know	found	had	number	
your	by	we	good	get	called	must	kind	
her	from	there	small	go		do	place	
my	as	this	little	take		have		
some	into	who	old	find		did		

Table 28.2: Upper row: topics extracted by the LDA model when trained on the combined Brown and TASA corpora. Middle row: topics extracted by LDA part of LDA-HMM model. Bottom row: topics extracted by HMM part of LDA-HMM model. Each column represents a single topic/class, and words appear in order of probability in that topic/class. Since some classes give almost all probability to only a few words, a list is terminated when the words account for 90% of the probability mass. From Figure 2 of [Gri+04]. Used with kind permission of Tom Griffiths.

More recently, [Die+17] proposed topic-RNN, which is similar to LDA-HMM, but replaces the HMM model with an RNN, which is a much more powerful model.

28.6 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 analyzing 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 28.32 for an example.

1. In contrast to this approach, we study here how the overall network activity can control single cell parameters such as input resistance, as well as time and space constants, parameters that are crucial for excitability and spariotemporal (sic) integration.

   The integrated architecture in this paper combines feed forward control and error feedback adaptive control using neural networks.

2. In other words, for our proof of convergence, we require the softassign algorithm to return a doubly stochastic matrix as *sinkhorn theorem guarantees that it will instead of a matrix which is merely close to being doubly stochastic based on some reasonable metric.

   The aim is to construct a portfolio with a maximal expected return for a given risk level and time horizon while simultaneously obeying *institutional or *legally required constraints.

3. The left graph is the standard experiment the right from a training with # samples.

   The graph G is called the *guest graph, and H is called the host graph.

Figure 28.31: Function and content words in the NIPS corpus, as distinguished by the LDA-HMM model. Graylevel indicates posterior probability of assignment to LDA component, with black being highest. The boxed word appears as a function word in one sentence, and as a content word in another sentence. Asterisked words had low frequency, and were treated as a single word type by the model. From Figure 4 of [Gri+04]. Used with kind permission of Tom Griffiths.

28.6.1 Noiseless ICA model

We can formalize the problem as follows. Let $\boldsymbol{x}_n \in \mathbb{R}^D$ be the vector of observed responses, at "time" n, where D is the number of sensors/microphones. Let $\boldsymbol{z}_n \in \mathbb{R}^D$ be the hidden vector of source signals at time n, of the same dimensionality as the observed signal. We assume that

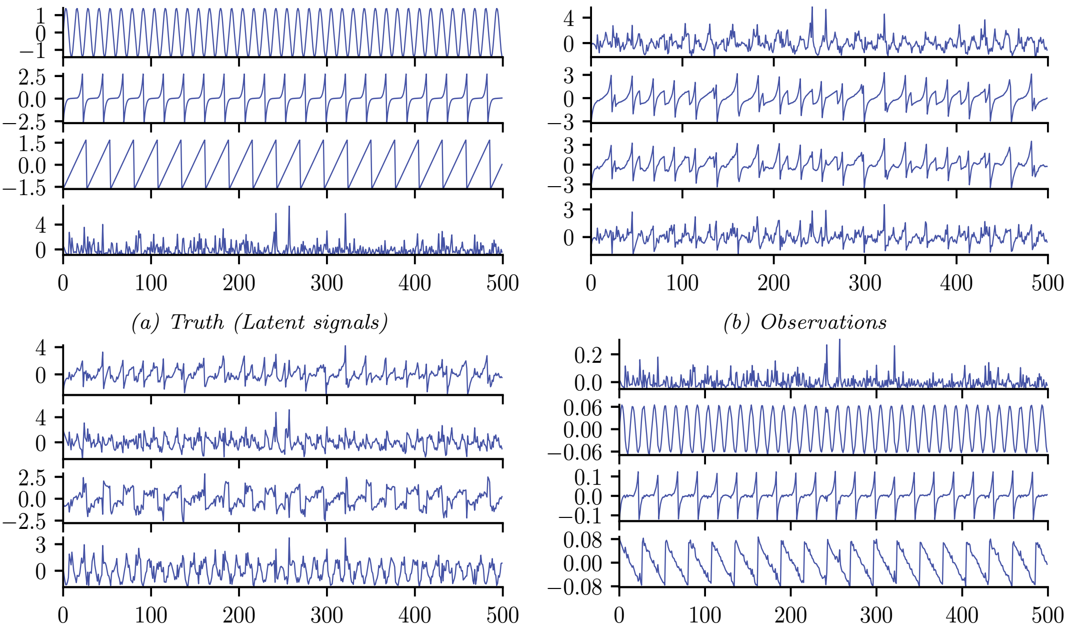
$$\boldsymbol{x}_n = \mathbf{A}\boldsymbol{z}_n \tag{28.142}$$

where $\mathbf{A}$ is an invertible $D \times D$ matrix known as the **mixing matrix** or the **generative weights**. The prior has the form $p(\boldsymbol{z}_n) = \prod_{j=1}^{D} p_j(z_j)$. Typically we assume this is a sparse prior, so only a subset of the signals are active at any one time (see Section 28.6.2 for further discussion of priors for this model). This model is called **independent components analysis** or **ICA**, since we assume that each observation $\boldsymbol{x}_n$ is a linear combination of independent components represented by sources $\boldsymbol{z}_n$, i.e,

$$x_{nj} = \sum_i A_{ij} z_{nj} \tag{28.143}$$

Our goal is to infer the source signals, $p(\boldsymbol{z}_n|\boldsymbol{x}_n, \mathbf{A})$. Since the model is noiseless, we have

$$p(\boldsymbol{z}_n|\boldsymbol{x}_n, \mathbf{A}) = \delta(\boldsymbol{z}_n - \mathbf{B}\boldsymbol{x}_n) \tag{28.144}$$



(a) Truth (Latent signals) *(b) Observations* *(c) PCA estimate* *(d) ICA estimate*

Figure 28.32: Illustration of ICA applied to 500 iid samples of a 4d source signal. This matches the true sources, up to permutation of the dimension indices. Generated by ica_demo.ipynb.

where $\mathbf{B} = \mathbf{A}^{-1}$ are the **recognition weights**. (We discuss how to estimate these weights in Section 28.6.3.)

28.6.2 The need for non-Gaussian priors

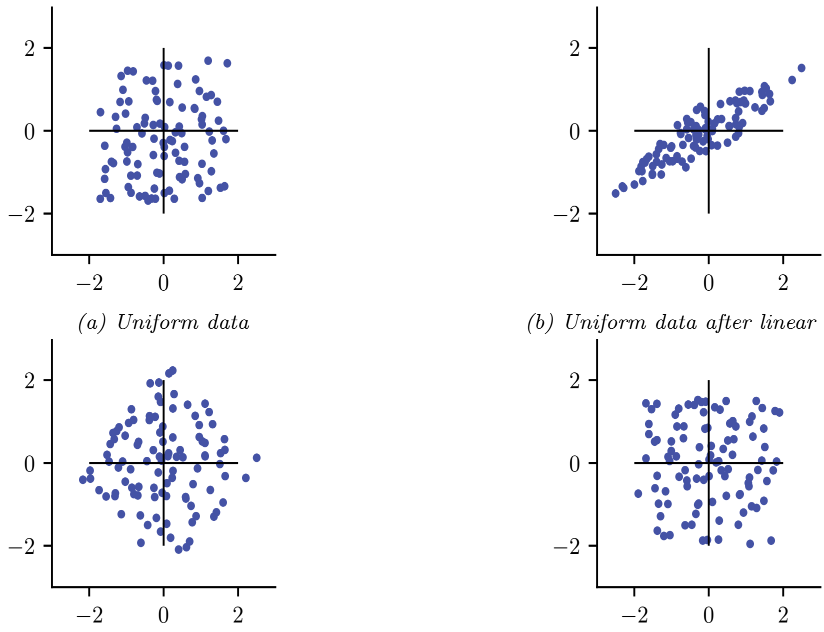
Since $\boldsymbol{x} = \mathbf{A}\boldsymbol{z}$, we have $\mathbb{E}[\boldsymbol{x}] = \mathbf{A}\mathbb{E}[\boldsymbol{z}]$ and $\text{Cov}[\boldsymbol{x}] = \text{Cov}[\mathbf{A}\boldsymbol{z}] = \mathbf{A}\text{Cov}[\boldsymbol{z}]\mathbf{A}^{\mathsf{T}}$. Without loss of generality, we can assume $\mathbb{E}[\boldsymbol{z}] = \mathbf{0}$, since we can always center the data. Similarly, we can assume $\text{Cov}[\boldsymbol{z}] = \mathbf{I}$, since $\mathbf{A}\mathbf{A}^{\mathsf{T}}$ can capture any correlation in $\boldsymbol{x}$. Thus $\boldsymbol{z}$ is a set of D unit variance, uncorrelated variables, as in factor analysis (Section 28.3.1).

However, this is not sufficient to uniquely identify $\mathbf{A}$ and hence $\boldsymbol{z}$, as we explained in Section 28.3.1.6. So we need to go beyond an uncorrelated prior and enforce an independent, and non-Gaussian, prior.

To illustrate this, suppose we have two independent sources with uniform distributions, as shown in Figure 28.33(a). Now suppose we have the following mixing matrix

$$\mathbf{A} = 0.3\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \tag{28.145}$$

Then we observe the data shown in Figure 28.33(b) (assuming no noise). The full-rank PCA model (where $K = D$) is equivalent to ICA, except it uses a factored Gaussian prior for $\boldsymbol{z}$. The result of using PCA is shown in Figure 28.33(c). This corresponds to a **whitening** or **sphering** of the data,



(a) Uniform data *(b) Uniform data after linear mixing*

(c) PCA estimate *(d) ICA estimate*

Figure 28.33: Illustration of ICA and PCA applied to 100 iid samples of a 2d source signal with a uniform distribution. Generated by ica_demo_uniform.ipynb.

in which $\text{Cov}[\boldsymbol{z}] = \mathbf{I}$. To uniquely recover the sources, we need to perform an additional rotation. The trouble is, there is no information in the symmetric Gaussian posterior to tell us which angle to rotate by. In a sense, PCA solves "half" of the problem, since it identifies the linear subspace; all that ICA has to do is then to identify the appropriate rotation. To do this, ICA uses an independent, but non-Gaussian, prior. The result is shown in Figure 28.33(d). This shows that ICA can recover the source variables, up to a permutation of the indices and possible sign change.

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

$$\log p(z) = -\sqrt{2}|z| - \log(\sqrt{2}) \tag{28.146}$$

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. If we set the mean to 0 and the variance to 1, we have $\mu = 0$ and $s = \frac{\sqrt{3}}{\pi}$, so the log pdf becomes the following (using the relationship $\text{sech}(x) = 1/\cosh(x)$):

$$\log p(z) = \log \text{sech}^2(z/2s) - \log(4s) = -2\log\cosh\left(\frac{\pi}{2\sqrt{3}}z\right) - \log\frac{4\sqrt{3}}{\pi} \tag{28.147}$$

28.6.3 Maximum likelihood estimation

In this section, we discuss how to estimate the mixing matrix $\mathbf{A}$ using maximum likelohood. By the change of variables formula we have

$$p_x(\boldsymbol{x}) = p_z(\boldsymbol{z})|\det(\mathbf{A}^{-1})| = p_z(\mathbf{B}\boldsymbol{x})|\det(\mathbf{B}))| \tag{28.148}$$

where $\mathbf{B} = \mathbf{A}^{-1}$. We can simplify the problem by first whitening the data by computing $\tilde{\boldsymbol{x}} = \mathbf{S}^{-\frac{1}{2}}\mathbf{U}^{\mathsf{T}}(\boldsymbol{x} - \overline{\boldsymbol{x}})$, where $\boldsymbol{\Sigma} = \mathbf{U}\mathbf{S}\mathbf{U}^{\mathsf{T}}$ is the SVD of the covariance matrix. We can now replace the general matrix $\mathbf{B}$ with an orthogonal matrix $\mathbf{V}$. Hence the likelihood becomes

$$p_x(\tilde{\boldsymbol{x}}) = p_z(\mathbf{V}\tilde{\boldsymbol{x}})|\det(\mathbf{V})| \tag{28.149}$$

Since we are constraining $\mathbf{V}$ to be orthogonal, the $|\det(\mathbf{V})|$ term is a constant, so we can drop it. In addition, we drop the tilde symbol, for brevity. Thus the average negative log likelihood can be written as

$$\text{NLL}(\mathbf{V}) = -\frac{1}{N}\log p(\mathbf{X}|\mathbf{V}) = -\frac{1}{N}\sum_{j=1}^{L}\sum_{n=1}^{N}\log p_j(\boldsymbol{v}_j^{\mathsf{T}}\boldsymbol{x}_n) \tag{28.150}$$

where $\boldsymbol{v}_j$ is the j'th row of $\mathbf{V}$, and the prior is factored, so $p(\boldsymbol{z}) = \prod_j p_j(z_j)$. We can also replace the sum over n with an expectation wrt the empirical distribution to get the following objective

$$\text{NLL}(\mathbf{V}) = \sum_j \mathbb{E}\left[G_j(z_j)\right] \tag{28.151}$$

where $z_j = \boldsymbol{v}_j^{\mathsf{T}}\boldsymbol{x}$ and $G_j(z_j) \triangleq -\log p_j(z_j)$. We want to minimize this (nonconvex) objective subject to the constraint that $\mathbf{V}$ is an orthogonal matrix.

It is straightforward to derive a (projected) gradient descent algorithm to fit this model. (For some JAX code, see https://github.com/tuananhle7/ica). One can also derive a faster algorithm that follows the natural gradient; see e.g., [Mac03, ch 34] for details. However, the most popular method is to use an approximate Newton method, known as **fast ICA** [HO00]. This was used to produce Figure 28.32.

28.6.4 Alternatives to MLE

In this section, we discuss various alternatives estimators for ICA that have been proposed over the years. We will show that they are equivalent to MLE. However, they bring interesting perspectives to the problem.

28.6.4.1 Maximizing non-Gaussianity

An early approach to ICA was to find a matrix $\mathbf{V}$ such that the distribution $\boldsymbol{z} = \mathbf{V}\boldsymbol{x}$ is as far from Gaussian as possible. (There is a related approach in statistics called **projection pursuit** [FT74].) One measure of non-Gaussianity is kurtosis, but this can be sensitive to outliers. Another measure is the **negentropy**, defined as

$$\text{negentropy}(z) \triangleq \mathbb{H}\left(\mathcal{N}(\mu, \sigma^2)\right) - \mathbb{H}(z) \tag{28.152}$$

where $\mu = \mathbb{E}[z]$ and $\sigma^2 = \mathbb{V}[z]$. Since the Gaussian is the maximum entropy distribution (for a fixed variance), this measure is always non-negative and becomes large for distributions that are highly non-Gaussian.

We can define our objective as maximizing

$$J(\mathbf{V}) = \sum_j \text{negentropy}(z_j) = \sum_j \mathbb{H}\left(\mathcal{N}(\mu_j, \sigma_j^2)\right) - \mathbb{H}(z_j) \tag{28.153}$$

where $\boldsymbol{z} = \mathbf{V}\boldsymbol{x}$. Since we assume $\mathbb{E}[\boldsymbol{z}] = \mathbf{0}$ and $\text{Cov}[\boldsymbol{z}] = \mathbf{I}$, the first term is a constant. Hence

$$J(\mathbf{V}) = \sum_j -\mathbb{H}(z_j) + \text{const} = \sum_j \mathbb{E}[\log p(z_j)] + \text{const} \tag{28.154}$$

which we see is equal (up to a sign change, and irrelevant constants) to the log-likelihood in Equation (28.151).

28.6.4.2 Minimizing total correlation

In Section 5.3.5.1, we show that the total correlation of $\boldsymbol{z}$ is given by

$$\text{TC}(\boldsymbol{z}) = \sum_j \mathbb{H}(z_j) - \mathbb{H}(\boldsymbol{z}) = D_{\mathbb{KL}}\left(p(\boldsymbol{z}) \parallel \prod_j p_k(z_j)\right) \tag{28.155}$$

This is zero iff the components of $\boldsymbol{z}$ are all mutually independent. In Section 21.3.1.1, we show that minimizing this results in a representation that is **disentangled**.

Now since $\boldsymbol{z} = \mathbf{V}\boldsymbol{x}$, we have

$$\text{TC}(\boldsymbol{z}) = \sum_j \mathbb{H}(z_j) - \mathbb{H}(\mathbf{V}\boldsymbol{x}) \tag{28.156}$$

Since we constrain $\mathbf{V}$ to be orthogonal, we can drop the last term, since $\mathbb{H}(\mathbf{V}\boldsymbol{x}) = \mathbb{H}(\boldsymbol{x}) = \text{const}$, since multiplying by $\mathbf{V}$ does not change the shape of the distribution. Hence we have $\text{TC}(\boldsymbol{z}) = \sum_k \mathbb{H}(z_k)$. Minimizing this is equivalent to maximizing the negentropy, which is equivalent to maximum likelihood.

28.6.4.3 Maximizing mutual information (InfoMax)

Let $z_j = \phi(\boldsymbol{v}_j^\mathsf{T}\boldsymbol{x}) + \epsilon$ be the noisy output of an encoder, where ϕ is some nonlinear scalar function, and $\epsilon \sim \mathcal{N}(0, 1)$. It seems reasonable to try to maximize the information flow through this system, a principle known as **infomax** [Lin88b; BS95a]. That is, we want to maximize the mutual information between $\boldsymbol{z}$ (the internal neural representation) and $\boldsymbol{x}$ (the observed input signal). We have $\mathbb{I}(\boldsymbol{x}; \boldsymbol{z}) = \mathbb{H}(\boldsymbol{z}) - \mathbb{H}(\boldsymbol{z}|\boldsymbol{x})$, 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

$$\mathbb{H}(\boldsymbol{z}) = \sum_j \mathbb{E}\left[\log \phi'(\boldsymbol{v}_j^\mathsf{T}\boldsymbol{x})\right] + \log|\det(\mathbf{V})| \tag{28.157}$$

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 (28.147) (ignoring irrelevant constants). Thus we see that infomax is equivalent to maximum likelihood.

28.6.5 Sparse coding

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 $\boldsymbol{z}$. The resulting model is as follows:

$$p(\boldsymbol{z}, \boldsymbol{x}) = p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z}) = \left[\prod_k \text{Laplace}(z_k|0, 1/\lambda)\right] \mathcal{N}(\boldsymbol{x}|\mathbf{W}\boldsymbol{z}, \sigma^2\mathbf{I}) \tag{28.158}$$

Thus each observation $\boldsymbol{x}$ is approximated by a sparse combination of columns of $\mathbf{W}$, known as **basis functions**; the sparse vector of weights is given by $\boldsymbol{z}$. (This can be thought of as a form of sparse factor analysis, except the sparsity is in the latent code $\boldsymbol{z}$, not the weight matrix $\mathbf{W}$.)

Not all basis functions will be active for any given observation, due to the sparsity penalty. Hence we can allow for more latent factors K than observations D. This is called **overcomplete representation**.

If we have a batch of N examples, stored in the rows of $\mathbf{X}$, the negative log joint becomes

$$-\log p(\mathbf{X}, \mathbf{Z}|\mathbf{W}) = \frac{1}{2\sigma^2}\sum_{n=1}^{N} ||\boldsymbol{x}_n - \mathbf{W}\boldsymbol{z}_n||_2^2 + \lambda||\boldsymbol{z}_n||_1 + \text{const} \tag{28.159}$$

$$= \frac{1}{2\sigma^2}||\mathbf{X} - \mathbf{W}\mathbf{Z}||_F^2 + \lambda||\mathbf{Z}||_{1,1} + \text{const} \tag{28.160}$$

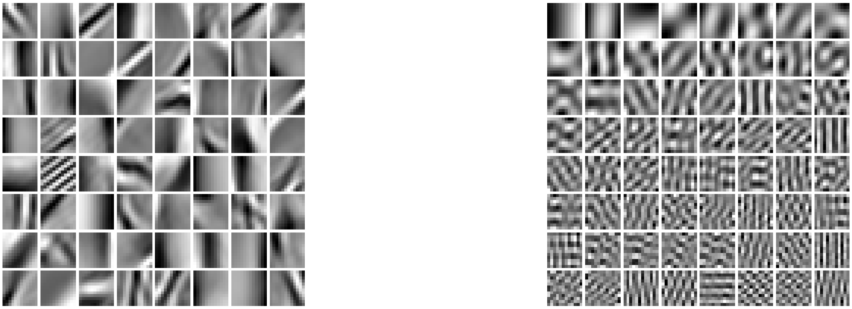
The MAP inference problem consists of estimating $\mathbf{Z}$ for a fixed $\mathbf{W}$; this is known as **sparse coding**, and can be solved using standard algorithms for sparse linear regression (see Section 15.2.6).[6],

The learning problem consists of estimating $\mathbf{W}$, marginalizing out $\mathbf{Z}$. This is called **dictionary learning**. Since this is computationally difficult, it is common to jointly optimize $\mathbf{W}$ and $\mathbf{Z}$ (thus "maxing out" wrt $\mathbf{Z}$ instead of marginalizing it out). We can do this by applying alternating optimization to Equation (28.160): estimating $\mathbf{Z}$ given $\mathbf{W}$ is a sparse linear regression problem, and estimating $\mathbf{W}$ given $\mathbf{Z}$ is a simple least squares problem. (For faster algorithms, see [Mai+10].)

Figure 28.34(a) illustrates the results of dictionary learning when applied to a dataset of natural image patches. (Each patch is first centered and normalized to unit norm.) We see that the method 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 28.34(b).[7]

6. Solving an ℓ_1 optimization problem for each data example can be slow. However, it is possible to train a neural network to approximate the outcome of this process; this is known as **predictive sparse decomposition** [KRL08; GL10].

7. 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 $\text{Cov}\left[I(x,y), I(x',y')\right] = f\left[(x-x')^2 + (y-y')^2\right]$ 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**.



(a) (b)

Figure 28.34: 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.

28.6.6 Nonlinear ICA

There are various ways to extend ICA to the nonlinear case. The resulting methods are similar to variational autoencoders (Chapter 21). For details, see e.g., [KKH20].

29 State-space models

29.1 Introduction

A **state-space model** (**SSM**) is a **partially observed Markov model**, in which the hidden state, $\boldsymbol{z}_t$, evolves over time according to a Markov process (Section 2.6), and each hidden state generates some observations $\boldsymbol{y}_t$ at each time step. (We focus on discrete time systems.) The main goal is to infer the hidden states given the observations. However, we may also be interested in using the model to predict future observations (e.g., for time-series forecasting).

An SSM can be represented as a stochastic discrete time nonlinear dynamical system of the form

$$\boldsymbol{z}_t = \boldsymbol{f}(\boldsymbol{z}_{t-1}, \boldsymbol{u}_t, \boldsymbol{q}_t) \tag{29.1}$$

$$\boldsymbol{y}_t = \boldsymbol{h}(\boldsymbol{z}_t, \boldsymbol{u}_t, \boldsymbol{y}_{1:t-1}, \boldsymbol{r}_t) \tag{29.2}$$

where $\boldsymbol{z}_t \in \mathbb{R}^{N_z}$ are the hidden states, $\boldsymbol{u}_t \in \mathbb{R}^{N_u}$ are optional observed inputs, $\boldsymbol{y}_t \in \mathbb{R}^{N_y}$ are observed outputs, $\boldsymbol{f}$ is the **transition function**, $\boldsymbol{q}_t$ is the **process noise**, $\boldsymbol{h}$ is the **observation function**, and $\boldsymbol{r}_t$ is the **observation noise**.

Rather than writing this as a deterministic function of random noise, we can represent it as a probabilistic model as follows:

$$p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{u}_t) = p(\boldsymbol{z}_t|\boldsymbol{f}(\boldsymbol{z}_{t-1}, \boldsymbol{u}_t)) \tag{29.3}$$

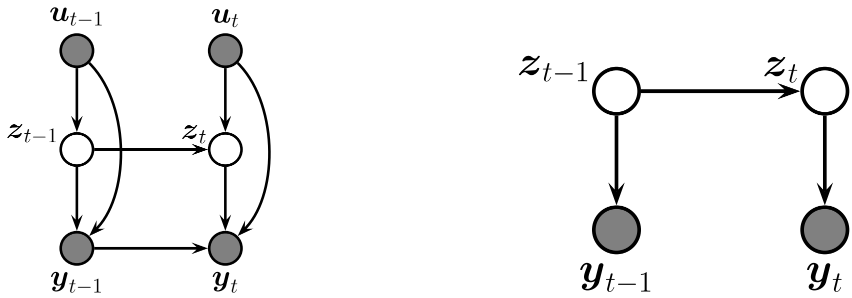
$$p(\boldsymbol{y}_t|\boldsymbol{z}_t, \boldsymbol{u}_t, \boldsymbol{y}_{1:t-1}) = p(\boldsymbol{y}_t|\boldsymbol{h}(\boldsymbol{z}_t, \boldsymbol{u}_t, \boldsymbol{y}_{1:t-1})) \tag{29.4}$$

where $p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{u}_t)$ is the **transition model**, and $p(\boldsymbol{y}_t|\boldsymbol{z}_t, \boldsymbol{u}_t, \boldsymbol{y}_{1:t-1})$ is the **observation model**. Unrolling over time, we get the following joint distribution:

$$p(\boldsymbol{y}_{1:T}, \boldsymbol{z}_{1:T}|\boldsymbol{u}_{1:T}) = \left[p(\boldsymbol{z}_1|\boldsymbol{u}_1) \prod_{t=2}^{T} p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{u}_t)\right] \left[\prod_{t=1}^{T} p(\boldsymbol{y}_t|\boldsymbol{z}_t, \boldsymbol{u}_t, \boldsymbol{y}_{1:t-1})\right] \tag{29.5}$$

If we assume the current observation $\boldsymbol{y}_t$ only depends on the current hidden state, $\boldsymbol{z}_t$, and the previous observation, $\boldsymbol{y}_{t-1}$, we get the graphical model in Figure 29.1(a). (This is called an auto-regressive state-space model.) However, by using a sufficient expressive hidden state $\boldsymbol{z}_t$, we can implicitly represent all the past observations, $\boldsymbol{y}_{1:t-1}$. Thus it is more common to assume that the observations are conditionally independent of each other (rather than having Markovian dependencies) given the hidden state. In this case the joint simplifies to

$$p(\boldsymbol{y}_{1:T}, \boldsymbol{z}_{1:T}|\boldsymbol{u}_{1:T}) = \left[p(\boldsymbol{z}_1|\boldsymbol{u}_1) \prod_{t=2}^{T} p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{u}_t)\right] \left[\prod_{t=1}^{T} p(\boldsymbol{y}_t|\boldsymbol{z}_t, \boldsymbol{u}_t)\right] \tag{29.6}$$



(a) (b)

Figure 29.1: State-space model represented as a graphical model. (a) Generic form, with inputs $\boldsymbol{u}_t$, hidden state $\boldsymbol{z}_t$, and observations $\boldsymbol{y}_t$. We assume the observation likelihood is first-order auto-regressive. (b) Simplified form, with no inputs, and Markovian observations.

Sometimes there are no external inputs, so the model further simplifies to the following unconditional generative model:

$$p(\boldsymbol{y}_{1:T}, \boldsymbol{z}_{1:T}) = \left[p(\boldsymbol{z}_1) \prod_{t=2}^{T} p(\boldsymbol{z}_t | \boldsymbol{z}_{t-1}) \right] \left[\prod_{t=1}^{T} p(\boldsymbol{y}_t | \boldsymbol{z}_t) \right] \tag{29.7}$$

See Figure 29.1(b) for the simplified graphical model.

29.2 Hidden Markov models (HMMs)

In this section, we discuss the **hidden Markov model** or **HMM**, which is an SSM in which the hidden states are discrete, so $z_t \in \{1, \ldots, K\}$. The observations may be discrete, $y_t \in \{1, \ldots, N_y\}$, or continuous, $\boldsymbol{y}_t \in \mathbb{R}^{N_y}$, or some combination, as we illustrate below. More details on HMMs can be found in Supplementary Chapter 29, as well as other references, such as [Rab89; Fra08; CMR05]. For an interactive introduction, see `https://nipunbatra.github.io/hmm/`.

29.2.1 Conditional independence properties

The HMM graphical model is shown in Figure 29.1(b). This encodes the assumption that the hidden states are Markovian, and the observations are iid conditioned on the hidden states. All that remains is to specify the form of the conditional probability distributions of each node.

29.2.2 State transition model

The initial state distribution is denoted by

$$p(z_1 = j) = \pi_j \tag{29.8}$$

where $\boldsymbol{\pi}$ is a discrete distribution over the K states.

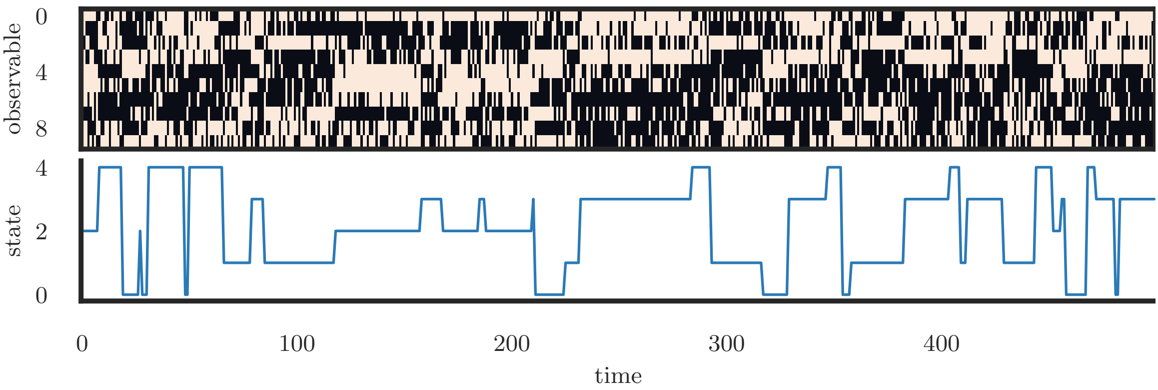


Figure 29.2: Some samples from an HMM with 10 Bernoulli observables. Generated by bernoulli_hmm_example.ipynb.

The **transition model** is denoted by

$$p(z_t = j | z_{t-1} = i) = A_{ij} \tag{29.9}$$

Here the i'th row of $\mathbf{A}$ 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 2.15.

29.2.3 Discrete likelihoods

The **observation model** $p(\boldsymbol{y}_t | z_t = j)$ can take multiple forms, depending on the type of data. For discrete observations we can use

$$p(y_t = k | z_t = j) = y_{jk} \tag{29.10}$$

For example, see the casino HMM example in Section 9.2.1.1.

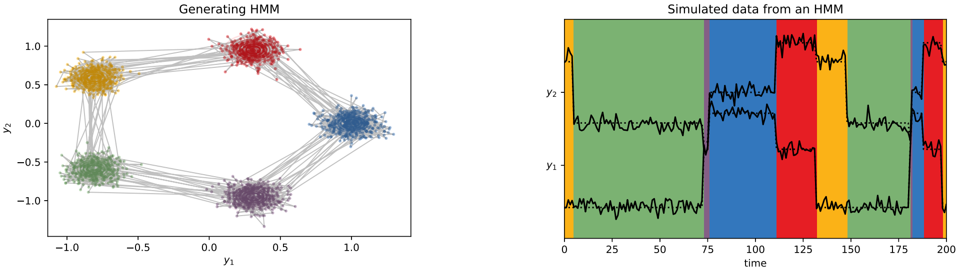
If we have D discrete observations per time step, we can use a factorial model of the form

$$p(\boldsymbol{y}_t | z_t = j) = \prod_{d=1}^{D} \mathrm{Cat}(y_{td} | \boldsymbol{y}_{d,j,:}) \tag{29.11}$$

In the special case of binary observations, this becomes

$$p(\boldsymbol{y}_t | z_t = j) = \prod_{d=1}^{D} \mathrm{Ber}(y_{td} | y_{d,j}) \tag{29.12}$$

In Figure 29.2, we give an example of an HMM with 5 hidden states and 10 Bernoulli observables.



(a) (b)

Figure 29.3: (a) Some 2d data sampled from a 5 state HMM. Each state emits from a 2d Gaussian. (b) The hidden state sequence is shown by the colors. We superimpose the observed 2d time series (note that we have shifted the vertical scale so the values don't overlap). Generated by gaussian_hmm_2d.ipynb.

29.2.4 Gaussian likelihoods

If $\boldsymbol{y}_t$ is continuous, it is common to use a Gaussian observation model:

$$p(\boldsymbol{y}_t|z_t = j) = \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \tag{29.13}$$

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 as 2d ellipses, as shown in Figure 29.3(a). We call these "**lily pads**", because of their shape. We can imagine a frog hopping from one lily 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 29.3(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 28.2.1), in that it generates clusters of observations, except now there is temporal correlation between the datapoints.

We can also use more flexible observation models. For example, if we use a M-component GMM, then we have

$$p(\boldsymbol{y}_t|z_t = j) = \sum_{k=1}^{M} w_{jk}\mathcal{N}(\boldsymbol{y}_t|\boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}) \tag{29.14}$$

This is called a **GMM-HMM**.

29.2.5 Autoregressive likelihoods

The standard HMM assumes the observations are conditionally independent given the hidden state. In practice this is often not the case. However, it is straightforward to have direct arcs from $\boldsymbol{y}_{t-1}$ to $\boldsymbol{y}_t$ as well as from z_t to $\boldsymbol{y}_t$, as in Figure 29.1(a). This is known as an **auto-regressive HMM**.

For continuous data, we can use an observation model of the form

$$p(\boldsymbol{y}_t|\boldsymbol{y}_{t-1}, z_t = j, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}_t|\mathbf{E}_j\boldsymbol{y}_{t-1} + \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \tag{29.15}$$

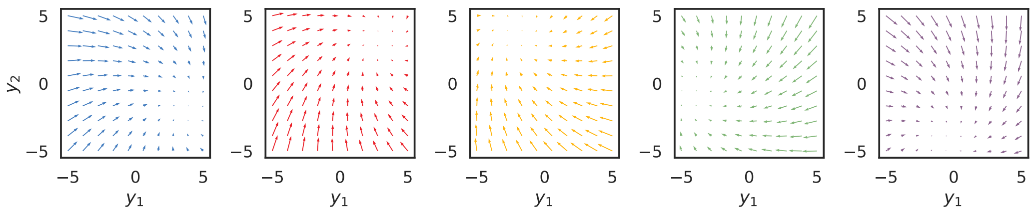


Figure 29.4: Illustration of the observation dynamics for each of the 5 hidden states. The attractor point corresponds to the steady state solution for the corresponding autoregressive process. Generated by hmm_ar.ipynb.

This is a linear regression model, where the parameters are chosen according to the current hidden state. (We could also use a nonlinear model, such as a neural network.) Such models are widely used in econometrics, where they are called **regime switching Markov model** [Ham90]. Similar models can be defined for discrete observations (see e.g., [SJ99]).

We can also consider higher-order extensions, where we condition on the last L observations:

$$p(\boldsymbol{y}_t|\boldsymbol{y}_{t-L:t-1}, z_t = j, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}_t|\sum_{\ell=1}^{L} \mathbf{W}_{j,\ell}\boldsymbol{y}_{t-\ell} + \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \tag{29.16}$$

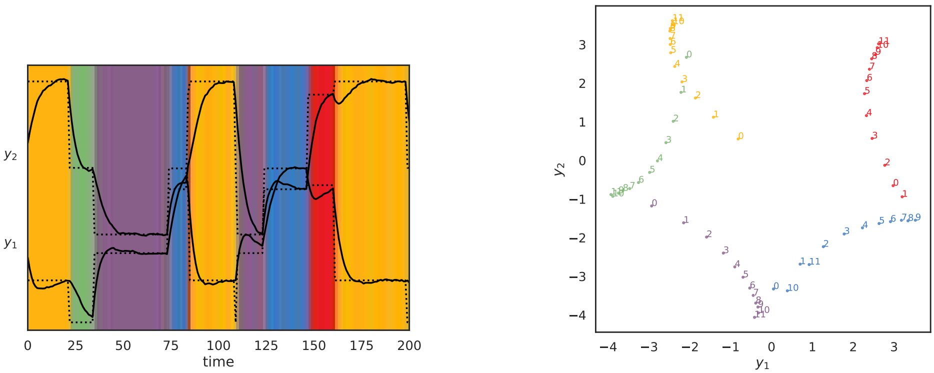
The AR-HMM essentially combines two Markov chains, one on the hidden variables, to capture long range dependencies, and one on the observed variables, to capture short range dependencies [Ber99]. Since all the visible nodes are observed, adding connections between them just changes the likelihood, but does not complicate the task of posterior inference (see Section 9.2.3).

Let us now consider a 2d example of this, due to Scott Linderman. We use a left-to-right transition matrix with 5 states. In addition, the final state returns to first state, so we just cycle through the states. Let $\boldsymbol{y}_t \in \mathbb{R}^2$, and suppose we set $\mathbf{E}_j$ to a rotation matrix with a small angle of 7 degrees, and we set each $\boldsymbol{\mu}_j$ to 72-degree separated points on a circle about the origin, so each state rotates 1/5 of the way around the circle. If the model stays in the same state j for a long time, the observed dynamics will converge to the steady state $\boldsymbol{y}_{*,j}$, which satisfies $\boldsymbol{y}_{*,j} = \mathbf{E}_j\boldsymbol{y}_{*,j} + \boldsymbol{\mu}_j$; we can solve for the steady state vector using $\boldsymbol{y}_{*,j} = (\mathbf{I} - \mathbf{E}_j)^{-1}\boldsymbol{\mu}_j$. We can visualize the induced 2d flow for each of the 5 states as shown in Figure 29.4.

In Figure 29.5(a), we show a trajectory sampled from this model. We see that the two components of the observation vector undergo different dynamics, depending on the underlying hidden state. In Figure 29.5(b), we show the same data in a 2d scatter plot. The first observation is the yellow dot (from state 2) at $(-0.8, 0.5)$. The dynamics converge to the stationary value of $\boldsymbol{y}_{*,2} = (-2.0, 3.8)$. Then the system jumps to the green state (state 3), so it adds an offset of $\boldsymbol{\mu}_3$ to the last observation, and then converges to the stationary value of $\boldsymbol{y}_{*,3} = (-4.3, -0.8)$. And so on.

29.2.6 Neural network likelihoods

For higher dimensional data, such as images, it can be useful to use a normalizing flow (Chapter 23), one per latent state (see e.g., [HNBK18; Gho+21]), as the class-conditional generative model. However, it is also possible to use discriminative neural network classifiers, which are much easier to train. In



(a) (b)

Figure 29.5: 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 4.7) 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 hmm_ ar.ipynb.

particular, note that the likelihood per state can be rewritten as follows:

$$p(\boldsymbol{y}_t|z_t = j) = \frac{p(z_t = j|\boldsymbol{y}_t)p(\boldsymbol{y}_t)}{p(z_t = j)} \propto \frac{p(z_t = j|\boldsymbol{y}_t)}{p(z_t = j)} \tag{29.17}$$

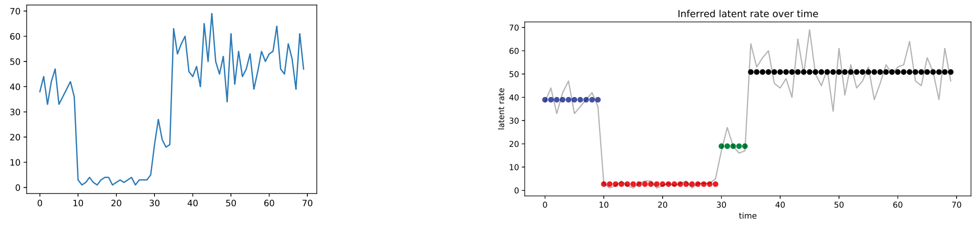
where we have dropped the $p(\boldsymbol{y}_t)$ term since it is independent of the state z_t. Here $p(z_t = j|\boldsymbol{y}_t)$ is the output of a classifier, and $p(z_t = j)$ is the probability of being in state j, which can be computed from the stationary distribution of the Markov chain (or empirically, if the state sequence is known). We can thus use discriminative classifiers to define the likelihood function when using gradient-based training. This is called the **scaled likelihood trick** [BM93; Ren+94]. [Guo+14] used this to create a **hybrid CNN-HMM** model for estimatng sequences of digits based on street signs.

29.3 HMMs: applications

In this section, we discuss some applications of HMMs.

29.3.1 Time series segmentation

In this section, we give a variant of the casino example from Section 9.2.1.1, where our goal is to segment a time series into different regimes, each of which corresponds to a different statistical distribution. In Figure 29.6a 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



(a) *(b)*

Figure 29.6: (a) A sample time series dataset of counts. (b) A segmentation of this data using a 4 state HMM. Generated by poisson_hmm_changepoint.ipynb.

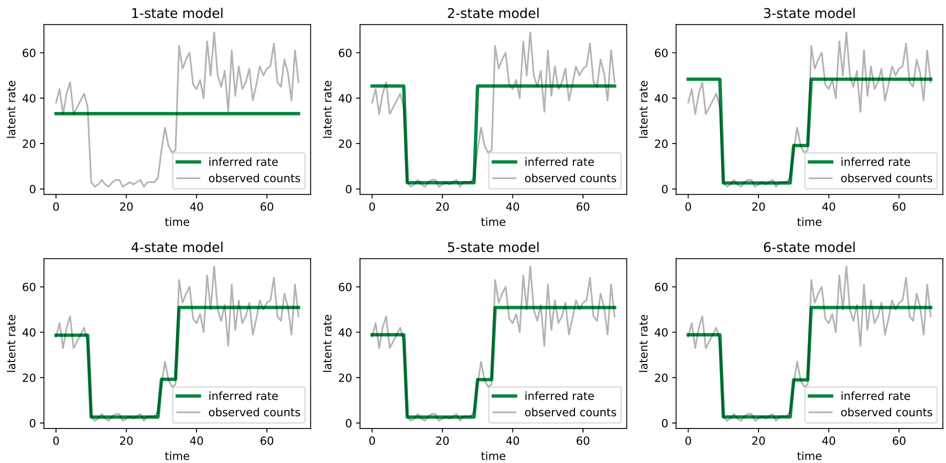


Figure 29.7: Segmentation of the time series using HMMs with 1–6 states. Generated by poisson_hmm_changepoint.ipynb.

rate λ_k:

$$p(y_t|z_t = k) = \text{Poi}(y_t|\lambda_k) \tag{29.18}$$

We use a uniform prior over the initial states. For the transition matrix, we assume 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:

$$z_1 \sim \text{Categorical}\left(\left\{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right\}\right) \tag{29.19}$$

$$z_t|z_{t-1} \sim \text{Categorical}\left(\left\{\begin{array}{cc} p & \text{if } z_t = z_{t-1} \\ \frac{1-p}{4-1} & \text{otherwise} \end{array}\right\}\right) \tag{29.20}$$

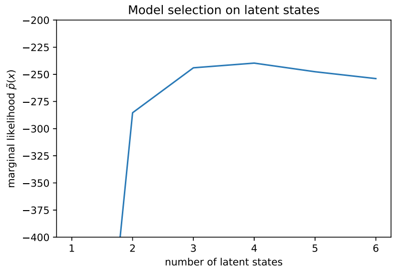


Figure 29.8: Marginal likelihood vs number of states K in the Poisson HMM. Generated by poisson_hmm_changepoint.ipynb.

We compute a MAP estimate for the parameters $\boldsymbol{\lambda}_{1:K}$ 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 29.6b. See that the method has successfully partitioned the data into 4 regimes, which is in fact how it was generated. (The generating rates are $\boldsymbol{\lambda} = (40, 3, 20, 50)$, 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 29.7, 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 $\boldsymbol{\lambda}$, we just maximize over the parameters (empirical Bayes approximation):

$$p(\boldsymbol{y}_{1:T}|K) \approx \max_{\boldsymbol{\lambda}} \sum_{\boldsymbol{z}} p(\boldsymbol{y}_{1:T}, \boldsymbol{z}_{1:T}|\boldsymbol{\lambda}, K) \tag{29.21}$$

We show this plot in Figure 29.8. 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.

29.3.2 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 $\boldsymbol{y}_{1:T}$ 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.)

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 matrix** (PSSM) by $\theta_t(v) = p(y_t = v)$. These parameters can be estimated by counting.

Now we turn the PSSM into an HMM with 3 hidden states, representing the events that the location t matches the consensus sequence, $z_t = M$, or inserts its own unique symbol, $z_t = I$, or deletes (skips) the corresponding consensus symbol, $z_t = D$. We define the observation models for these 3 events as follows. For matches, we use the PSSM $p(y_t = v|z_t = M) = \theta_t(v)$. For insertions

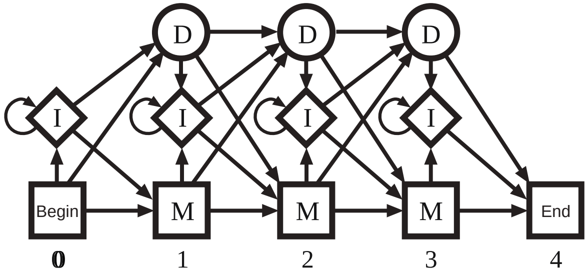


Figure 29.9: State transition diagram for a profile HMM. From Figure 5.7 of [Dur+98]. Used with kind permission of Richard Durbin.

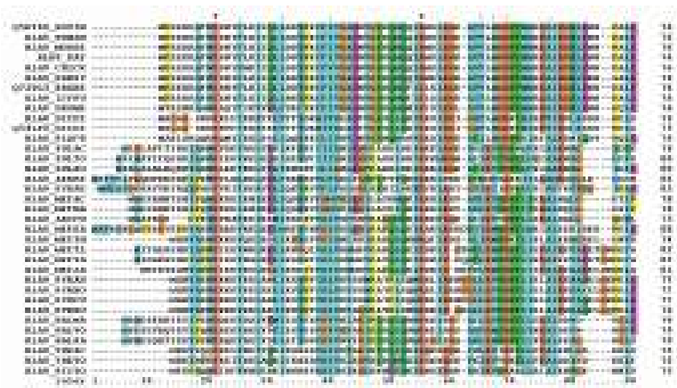


Figure 29.10: Example of multiple sequence alignment. We show the first 90 positions of the acidic ribosomal protein P0 from several organisms. Colors represent functional properties of the corresponding amino acid. Dashes represent insertions or deletions. From `https://en.wikipedia.org/wiki/Multiple_sequence_alignment`. Used with kind permission of Wikipedia author Miguel Andrade.

we use the uniform distribution $p(y_t = v|z_t = I) = 1/V$, where V is the size of the vocabulary. For deletions, we use $p(y_t = -|z_t = D)$, where "-" is a special deletion symbol used to pad the generated sequence to the correct length. The corresponding state transition matrix is shown in Figure 29.9: we see that matches and deletions advance one location along the consensus sequence, but insertions stay in the same location (represented by the self-transition from I to I). This model is known as a **profile HMM**.

Given a profile HMM with consensus parameters $\boldsymbol{\theta}$, we can compute $p(\boldsymbol{y}_{1:T}|\boldsymbol{\theta})$ in $O(T)$ time using the forwards algorithm, as described in Section 9.2.2. This can be used to decide if the sequence belongs to this family or not, by thresholding the log-odds score, $L(\boldsymbol{y}) = \log p(\boldsymbol{y}|\boldsymbol{\theta})/p(\boldsymbol{y}|\mathcal{M}_0)$, where $\mathcal{M}_0$ is a baseline model, such as the uniform distribution. If the string matches, we can compute an alignment to the consensus using the Viterbi algorithm, as described in Section 9.2.6. See Figure 29.10 for an illustration of such a **multiple sequence alignment**. If we don't have an initial set of aligned sequences from which to compute the consensus sequence $\boldsymbol{\theta}$, we can use the Baum-Welch algorithm

(Section 29.4.1) to compute the MLE for the parameters $\boldsymbol{\theta}$ from a set of unaligned sequences. For details, see e.g., [Dur+98, Ch.6].

29.3.3 Spelling correction

In this section, we illustrate how to use an HMM for **spelling correction**. The goal is to infer the sequence of words $\boldsymbol{z}_{1:T}$ that the user meant to type, given observations of what they actually did type, $\boldsymbol{y}_{1:T}$.

29.3.3.1 Baseline model

We start by using a simple unigram language model, so $p(\boldsymbol{z}_{1:T}) = \prod_{1:T} p(z_t)$, 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 "message" z_t gets corrupted by one of four kinds of error: substitution error, where we swap one letter for another (e.g., "government" mistyped as "govermment"); transposition errors, where we swap the order of two adjacent letters (e.g., "government" mistyped as "govermnent"); deletion errors, where we omit one letter (e.g., "government" mistyped as "goverment"); and insertion errors, where we add an extra latter (e.g., "government" mistyped as "governmennt"). If y differs from z by d such 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 distance d away from y. We can then define the following likelihood function:

$$p(y|z) = \begin{cases} p_1 & y = z \\ p_2 & y \in \mathcal{D}(z, 1) \\ p_3 & y \in \mathcal{D}(z, 2) \\ p_4 & \text{otherwise} \end{cases} \tag{29.22}$$

where $p_1 > p_2 > p_3 > p_4$.

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[1], can work can 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) 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 and then quickly changes it to q_2 followed by a click on a link, it suggests that q_2 is a manual correction for q_1, so we can set $(z = q_2, y = q_1)$. This heuristic has been used in the Etsy search engine.[2] It is also possible to manually collect such data (see e.g., [Hag+17]), or to algorithmically create $(\boldsymbol{z}, \boldsymbol{y})$ pairs, where $\boldsymbol{y}$ is an automatically generated misspelling of $\boldsymbol{z}$ (see e.g., [ECM18]).

1. See his excellent tutorial at `http://norvig.com/spell-correct.html`.
2. See this blogpost by Mohit Nayyar for details: `https://codeascraft.com/2017/05/01/modeling-spelling-correction-for-search-at-etsy/`.

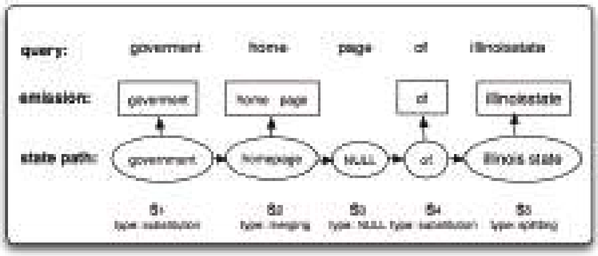


Figure 29.11: Illustration of an HMM applied to spelling correction. The top row, labeled "query", represents the search query $\boldsymbol{y}_{1:T}$ typed by the user, namely "goverment home page of illinoisstate". The bottom row, labeled "state path", represents the most probable assignment to the hidden states, $\boldsymbol{z}_{1:T}$, 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].

29.3.3.2 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 $p(\boldsymbol{z}_{1:T}) = \prod_t p(z_t)$ by a standard first-order language model on words, $p(\boldsymbol{z}_{1:T}) = \prod_t p(z_t|z_{t-1})$. The parameters of this model can be estimated by counting bigrams in a large corpus of "clean" text (see Section 2.6.3.1). 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 9.2.6.5. 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 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 $\boldsymbol{\alpha}_t$.

29.3.3.3 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., "illinoisstate" → "illinois state".
- Within-word errors, such as substitution, transposition, insertion and deletion of letters, as we discussed in Section 29.3.3.2.

We can model this with an HMM, where we augment the state space with a **silent state**, that does not emit any symbols. Figure 29.11 illustrates how this model can "denoise" the observed query "goverment home page of illinoisstate" 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].

29.4 HMMs: parameter learning

In this section, we discuss how to compute a point estimate or the full posterior over the model parameters of an HMM given a set of partially observed sequences.

29.4.1 The Baum-Welch (EM) algorithm

In this section, we discuss how to compute an approximate MLE for the parameters of an HMM using the EM algorithm which is an iterative bound optimization algorithm (see Section 6.5.3 for details). When applied to HMMs, the resulting method is known as the **Baum-Welch** algorithm [Bau+70].

29.4.1.1 Log likelihood

The joint probability of a single sequence is given by

$$p(\boldsymbol{y}_{1:T}, \boldsymbol{z}_{1:T}|\boldsymbol{\theta}) = [p(z_1|\boldsymbol{\pi})] \left[\prod_{t=2}^{T} p(z_t|z_{t-1}, \mathbf{A})\right] \left[\prod_{t=1}^{T} p(\boldsymbol{y}_t|z_t, \mathbf{B})\right] \tag{29.23}$$

$$= \left[\prod_{k=1}^{K} \pi_k^{\mathbb{I}(z_1=k)}\right] \left[\prod_{t=2}^{T} \prod_{j=1}^{K} \prod_{k=1}^{K} A_{jk}^{\mathbb{I}(z_{t-1}=j, z_t=k)}\right] \left[\prod_{t=1}^{T} \prod_{k=1}^{K} p(\boldsymbol{y}_t|\mathbf{B}_k)^{\mathbb{I}(z_t=k)}\right] \tag{29.24}$$

where $\boldsymbol{\theta} = (\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$. Of course, we cannot compute this objective, since $\boldsymbol{z}_{1:T}$ is hidden. So instead we will optimize the expected complete data log likelihood, where expectations are taken using the parameters from the previous iteration of the algorithm:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \mathbb{E}_{p(\boldsymbol{z}_{1:T}|\boldsymbol{y}_{1:T}, \boldsymbol{\theta}^{\text{old}})} \left[\log p(\boldsymbol{y}_{1:T}, \boldsymbol{z}_{1:T}|\boldsymbol{\theta})\right] \tag{29.25}$$

This can be easily summed over N sequences. See Figure 29.12 for the graphical model.

The above objective is a lower bound on the observed data log likelihood, $\log p(\boldsymbol{y}_{1:T}|\boldsymbol{\theta})$, so the entire procedure is a bound optimization method that is guaranteed to converge to a local optimum. (In fact, if suitably initialized, the method can be shown to converge to (close to) one of the global optima [YBW15].)

29.4.1.2 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.

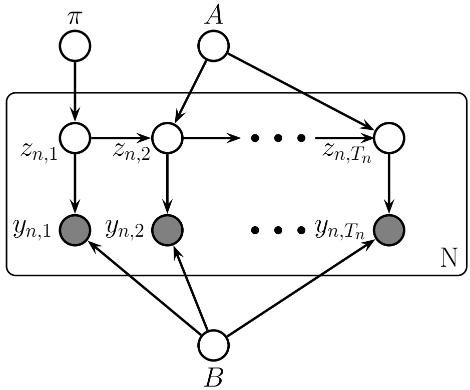


Figure 29.12: HMM with plate notation. A are the parameters for the state transition matrix $p(z_t|z_{t-1})$ and B are the parameters for the discrete observation model $p(x_t|z_t)$. T_n is the length of the n'th sequence.

To compute the expected sufficient statistics, we first run the forwards-backwards algorithm on each sequence (see Section 9.2.3). This returns the following node and edge marginals:

$$\gamma_{n,t}(j) \triangleq p(z_t = j|\boldsymbol{y}_{n,1:T_n}, \boldsymbol{\theta}^{old}) \tag{29.26}$$

$$\xi_{n,t}(j,k) \triangleq p(z_{t-1} = j, z_t = k|\boldsymbol{y}_{n,1:T_n}, \boldsymbol{\theta}^{\text{old}}) \tag{29.27}$$

where T_n is the length of sequence n. 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):

$$\mathbb{E}\left[N_k^1\right] = \sum_{n=1}^{N} \gamma_{n,1}(k),\ \mathbb{E}\left[N_k\right] = \sum_{n=1}^{N}\sum_{t=2}^{T_n} \gamma_{n,t}(k),\ \mathbb{E}\left[N_{jk}\right] = \sum_{n=1}^{N}\sum_{t=2}^{T_n} \xi_{n,t}(j,k) \tag{29.28}$$

Given the above quantities, we can compute the expected complete data log likelihood as follows:

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) &= \sum_{k=1}^{K} \mathbb{E}\left[N_k^1\right] \log \pi_k + \sum_{j=1}^{K}\sum_{k=1}^{K} \mathbb{E}\left[N_{jk}\right] \log A_{jk} \\ &+ \sum_{n=1}^{N}\sum_{t=1}^{T_n}\sum_{k=1}^{K} p(z_t = k|\boldsymbol{y}_{n,1:T_n}, \boldsymbol{\theta}^{\text{old}}) \log p(\boldsymbol{y}_{n,t}|\boldsymbol{\theta}_k) \end{aligned} \tag{29.29}$$

29.4.1.3 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:

$$\hat{A}_{jk} = \frac{\mathbb{E}\left[N_{jk}\right]}{\sum_{k'} \mathbb{E}\left[N_{jk'}\right]},\ \hat{\pi}_k = \frac{\mathbb{E}\left[N_k^1\right]}{N} \tag{29.30}$$

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

$$\mathbb{E}\left[M_{kv}\right] = \sum_{n=1}^{N} \sum_{t=1}^{T_n} \gamma_{n,t}(k)\mathbb{I}\left(y_{n,t} = v\right) = \sum_{n=1}^{N} \sum_{t:\boldsymbol{y}_{n,t}=v} \gamma_{n,t}(k) \tag{29.31}$$

The M step has the form

$$\hat{B}_{kv} = \frac{\mathbb{E}\left[M_{kv}\right]}{\mathbb{E}\left[N_k\right]} \tag{29.32}$$

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

$$\overline{\boldsymbol{y}}_k = \sum_{n=1}^{N} \sum_{t=1}^{T_n} \gamma_{n,t}(k)\boldsymbol{y}_{n,t}, \quad \overline{\boldsymbol{y}\boldsymbol{y}^{\mathsf{T}}}_k = \sum_{n=1}^{N} \sum_{t=1}^{T_n} \gamma_{n,t}(k)\boldsymbol{y}_{n,t}\boldsymbol{y}_{n,t}^{\mathsf{T}} \tag{29.33}$$

The M step becomes

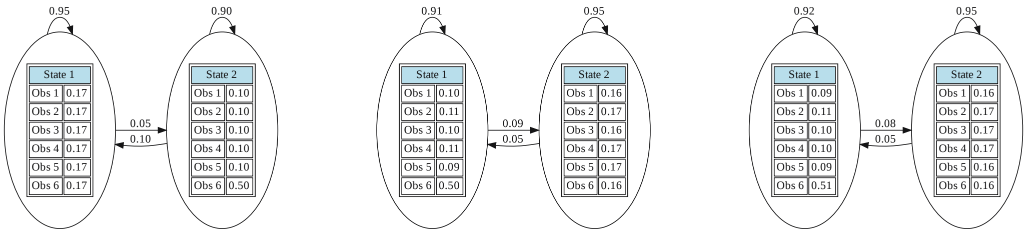
$$\hat{\boldsymbol{\mu}}_k = \frac{\overline{\boldsymbol{y}}_k}{\mathbb{E}\left[N_k\right]} \tag{29.34}$$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{\overline{\boldsymbol{y}\boldsymbol{y}^{\mathsf{T}}}_k - \mathbb{E}\left[N_k\right]\hat{\boldsymbol{\mu}}_k\hat{\boldsymbol{\mu}}_k^{\mathsf{T}}}{\mathbb{E}\left[N_k\right]} \tag{29.35}$$

In practice, we often need to add a log prior to these estimates to ensure the resulting $\hat{\boldsymbol{\Sigma}}_k$ estimate is well-conditioned. See [Mur22, Sec 4.5.2] for details.

Algorithm 29.1: Baum-Welch algorithm for (discrete observation) HMMs

Initialize parameters $\boldsymbol{\theta}$
for *each iteration* **do**
  // E step
  Initialize expected counts: $\mathbb{E}\left[N_k\right] = 0$, $\mathbb{E}\left[N_{jk}\right] = 0$, $\mathbb{E}\left[M_{kv}\right] = 0$
  for *each datacase* n **do**
    Use forwards-backwards algorithm on $\boldsymbol{y}_n$ to compute $\gamma_{n,t}$ and $\xi_{n,t}$ (Equations 29.26–29.27)
    $\mathbb{E}\left[N_k\right] := \mathbb{E}\left[N_k\right] + \sum_{t=2}^{T_n} \gamma_{n,t}(k)$
    $\mathbb{E}\left[N_{jk}\right] := \mathbb{E}\left[N_{jk}\right] + \sum_{t=2}^{T_n} \xi_{n,t}(j,k)$
    $\mathbb{E}\left[M_{kv}\right] := \mathbb{E}\left[M_{kv}\right] + \sum_{t:x_{n,t}=v} \gamma_{n,t}(k)$
  // M step
  Compute new parameters $\boldsymbol{\theta} = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ using Equations 29.30



(a) *(b)* *(c)*

Figure 29.13: Illustration of the casino HMM. (a) True parameters used to generate the data. (b) Estimated parameters using EM. (c) Estimated parameters using SGD. Note that in the learned models (b-c), states 1 and 2 are switched compared to the generating model (a), due to unidentifiability. Generated by casino_hmm_training.ipynb.

29.4.1.4 Initialization

As usual with EM, we must take care to ensure that we initialize the parameters carefully, to minimize the chance of getting stuck in poor local optima. There are several ways to do this, such as:

- Use some fully labeled data to initialize the parameters.
- Initially ignore the Markov dependencies, and estimate the observation parameters using the standard mixture model estimation methods, such as K-means or EM.
- Randomly initialize the parameters, use multiple restarts, and pick the best solution.

Techniques such as deterministic annealing [UN98; RR01a] can help mitigate the effect of local minima. Also, just as K-means is often used to initialize EM for GMMs, so it is common to initialize EM for HMMs using Viterbi training. The Viterbi algorithm is explained in Section 9.2.6, but basically it is an algorithm to compute the single most probable path. As an approximation to the E step, we can replace the sum over paths with the statistics computed using this single path. Sometimes this can give better results [AG11].

29.4.1.5 Example: casino HMM

In this section, we fit the casino HMM from Section 9.2.1.1. The true generative model is shown in Figure 29.13a. We used this to generate 4 sequences of length 5000, totalling 20,000 observations. We initialized the model with random parameters. We ran EM for 200 iterations and got the results in Figure 29.13b. We see that the learned parameters are close to the true parameters, modulo label switching of the states (see Section 28.2.6).

29.4.2 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

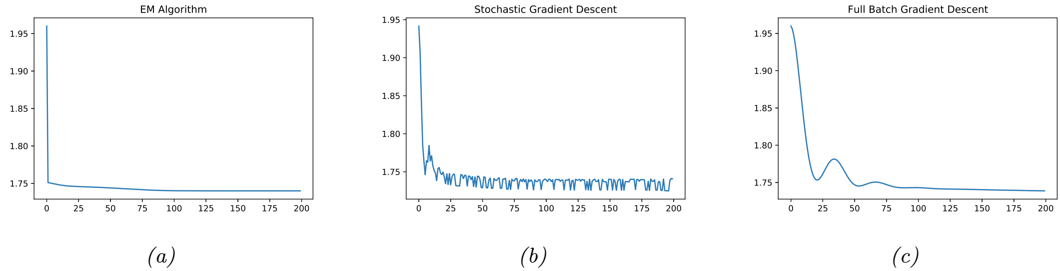


Figure 29.14: Average negative log likelihood per learning step the casino HMM. (a) EM. (b) SGD with minibatch size 1. *(b) Full batch gradient descent. Generated by casino_hmm_training.ipynb.*

bound optimization to the online case (see e.g., [Mai15]), this can take a lot of memory.

A simple alternative is to optimize $\log p(\boldsymbol{y}_{1:T}|\boldsymbol{\theta})$ using SGD. We can compute this objective using the forwards algorithm, as shown in Equation (8.7):

$$\log p(\boldsymbol{y}_{1:T}|\boldsymbol{\theta}) = \sum_{t=1}^{T} \log p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) = \sum_{t=1}^{T} \log Z_t \tag{29.36}$$

where the normalization constant for each time step is given by

$$Z_t \triangleq p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}) = \sum_{j=1}^{K} p(z_t = j|\boldsymbol{y}_{1:t-1}) p(\boldsymbol{y}_t|z_t = j) \tag{29.37}$$

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 $\boldsymbol{\Sigma}_k$ 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].

29.4.2.1 Example: casino HMM

In this section, we use SGD to fit the casino HMM using the same data as in Section 29.4.1.5. We show the learning the learning curves in Figure 29.14. We see that SGD converges slightly more slowly than EM, and is not monotonic in how it decreases the NLL loss, even in the full batch case. However, the final parameters are similar, as shown in Figure 29.13.

29.4.3 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

$$[\boldsymbol{y}_t]_k \triangleq p(y_t = k|\boldsymbol{y}_{1:t-1}) \tag{29.38}$$

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 (9.8), 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:

$$[\mathbf{P}_2]_{ij} = p(y_t = i, y_{t-1} = j) \tag{29.39}$$

We also need to estimate the third order statistics

$$[\mathbf{P}_3]_{ijk} = p(y_t = i, y_{t-1} = j, y_{t-2} = k) \tag{29.40}$$

Using these quantities, it 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 29.8.2 where we discuss spectral methods for fitting linear dynamical systems.

29.4.4 Bayesian HMMs

MLE methods can easily overfit, and can suffer from numerical problems, especially when sample sizes are small. In this section, we briefly discuss some approaches to inferring the posterior over the parameters, $p(\boldsymbol{\theta}|\mathcal{D})$. By adopting a Bayesian approach, we can also allow the number of states to be unbounded by using a hierarchical Dirichlet process (Supplementary Section 31.1) to get a **HDP-HMM** [Fox+08].

There are various algorithms we can use to perform posterior inference, such as variational Bayes EM [Bea03] or blocked Gibbs sampling (see Section 29.4.4.1), that alternates between sampling latent sequences $\boldsymbol{z}^s_{1:T,1:N}$ using the forwards filtering backwards sampling algorithm (Section 9.2.7) and sampling the parameters from their full conditionals, $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:T}, \boldsymbol{z}^s_{1:T,1:N})$. Unfortuntely, the high correlation between $\boldsymbol{z}$ and $\boldsymbol{\theta}$ makes this coordinate-wise approach rather slow.

A faster approach is to marginalize out the discrete latents (using the forwards algorithm), and then to use MCMC [Fot+14] or SVI [Obe+19] to sample from the following log posterior:

$$\log p(\boldsymbol{\theta}, \mathcal{D}) = \log p(\boldsymbol{\theta}) + \sum_{n=1}^{N} \log p(\boldsymbol{y}_{1:T,n}|\boldsymbol{\theta}) \tag{29.41}$$

This is a form of "collapsed" inference.

29.4.4.1 Blocked Gibbs sampling for HMMs

This section is written by Xinglong Li.

In this section, we discuss Bayesian inference for HMMs using blocked Gibbs sampling [Sco02]. For the observation model, we consider the first-order auto-regressive HMM model in Section 29.2.5, so $p(\boldsymbol{y}_t|\boldsymbol{y}_{t-1}, z_t = j, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}_t|\mathbf{E}_j\boldsymbol{y}_{t-1} + \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$. For a model with K hidden states, the unknown parameters are $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \mathbf{A}, \mathbf{E}_1, \ldots, \mathbf{E}_K, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K\}$, where we assume (for notational simplicity) that $\boldsymbol{\mu}_j$ of each autoregressive model is known, and that we condition the observations on $\boldsymbol{y}_1$.

We alternate between sampling from $p(\boldsymbol{z}_{1:T}|\boldsymbol{y}_{1:T}, \boldsymbol{\theta})$ using the forwards filtering backwards sampling algorithm (Section 9.2.7), and sampling from $p(\boldsymbol{\theta}|\boldsymbol{z}_{1:T}, \boldsymbol{y}_{1:T})$. Sampling from $p(\boldsymbol{\theta}|\boldsymbol{z}_{1:T}, \boldsymbol{y}_{1:T})$ is easy if we use conjugate priors. Here we use a Dirichlet prior for $\boldsymbol{\pi}$ and each row $\mathbf{A}_{j\cdot}$ of the transition matrix, and choose the matrix normal inverse Wishart distribution as the prior for $\{\mathbf{E}_j, \boldsymbol{\Sigma}_j\}$ of each autoregressive model, similar to Bayesian multivariate linear regression Section 15.2.9. In particular, the prior distributions of $\boldsymbol{\theta}$ are:

$$\boldsymbol{\pi} \sim \mathrm{Dir}(\breve{\boldsymbol{\alpha}}_\pi) \qquad \mathbf{A}_{j\cdot} \sim \mathrm{Dir}(\breve{\boldsymbol{\alpha}}_A) \tag{29.42}$$

$$\boldsymbol{\Sigma}_j \sim \mathrm{IW}(\breve{\nu}_j, \breve{\boldsymbol{\Psi}}_j) \qquad \mathbf{E}_j|\boldsymbol{\Sigma}_j \sim \mathcal{MN}(\breve{\mathbf{M}}_j, \boldsymbol{\Sigma}_j, \breve{\mathbf{V}}_j) \tag{29.43}$$

where $\breve{\alpha}_{\pi,k}=\breve{\alpha}_\pi /K$ and $\breve{\alpha}_{A,k}=\breve{\alpha}_A /K$. The log prior probability is

$$\begin{aligned}
\log p(\boldsymbol{\theta}) =& c + \sum_{k=1}^K \frac{\breve{\alpha}_\pi}{K} \log \pi_k + \sum_{j=1}^K \sum_{k=1}^K \frac{\breve{\alpha}_A}{K} \log A_{jk} - \sum_{j=1}^K \left(\frac{\breve{\nu}_j + N_y + 1}{2} \log|\boldsymbol{\Sigma}_j| + \frac{1}{2}\mathrm{tr}\left(\breve{\boldsymbol{\Psi}}_j \boldsymbol{\Sigma}_j^{-1}\right)\right) \\
& - \sum_{j=1}^K \left(\frac{1}{2}\log|\boldsymbol{\Sigma}_j| + \frac{1}{2}\mathrm{tr}((\mathbf{E}_j - \breve{\mathbf{M}}_j)^\mathsf{T} \boldsymbol{\Sigma}_j^{-1} (\mathbf{E}_j - \breve{\mathbf{M}}_j)\, \breve{\mathbf{V}}_j) \right)
\end{aligned} \tag{29.44}$$

Given $\boldsymbol{y}_{1:T}$ and $\boldsymbol{z}_{1:T}$ we denote $N_j = \sum_{t=2}^T \mathbb{I}(z_t = j)$ and $N_{jk} = \sum_{t=1}^{T-1} \mathbb{I}(z_t = j, z_{t+1} = k)$. The joint likelihood is

$$\begin{aligned}
\log p(\boldsymbol{y}_{1:T}, \boldsymbol{z}_{1:T}|\boldsymbol{\theta}) =& c + \sum_{k=1}^K \mathbb{I}(z_1 = k) \log \pi_k + \sum_{j=1}^K \sum_{k=1}^K N_{jk} \log A_{jk} \\
& - \sum_{j=1}^K \sum_{z_t=j} \left(\frac{1}{2}\log|\boldsymbol{\Sigma}_j| + \frac{1}{2}(\boldsymbol{y}_t - \mathbf{E}_j\boldsymbol{y}_{t-1} - \boldsymbol{\mu}_j)^\mathsf{T} \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{y}_t - \mathbf{E}_j\boldsymbol{y}_{t-1} - \boldsymbol{\mu}_j) \right)
\end{aligned} \tag{29.45}$$

$$\begin{aligned}
=& c + \sum_{k=1}^K \mathbb{I}(z_1 = k) \log \pi_k + \sum_{j=1}^K \sum_{k=1}^K N_{jk} \log A_{jk} \\
& - \sum_{j=1}^K \left(\frac{N_j}{2}\log|\boldsymbol{\Sigma}_j| + \frac{1}{2}(\hat{\mathbf{Y}}_j - \mathbf{E}_j\tilde{\mathbf{Y}}_j)^\mathsf{T} \boldsymbol{\Sigma}_j^{-1} (\hat{\mathbf{Y}}_j - \mathbf{E}_j\tilde{\mathbf{Y}}_j) \right)
\end{aligned} \tag{29.46}$$

where $\hat{\mathbf{Y}}_j = [\boldsymbol{y}_t - \boldsymbol{\mu}_j]_{z_t=j}$ and $\tilde{\mathbf{Y}}_j = [\boldsymbol{y}_{t-1}]_{z_t=j}$, and it can be seen that $\hat{\mathbf{Y}}_j \sim \mathcal{MN}\left(\hat{\mathbf{Y}}_j|\mathbf{E}_j\tilde{\mathbf{Y}}_j, \boldsymbol{\Sigma}_j, \mathbf{I}_{N_j}\right)$.

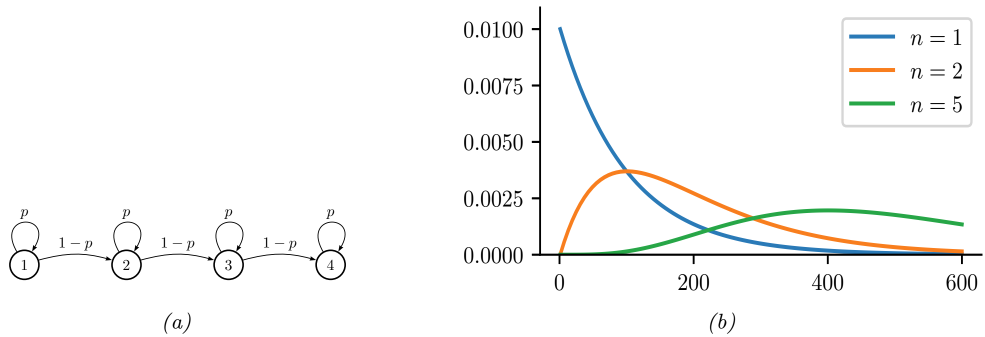


Figure 29.15: (a) A Markov chain with $n = 4$ repeated states and self loops. (b) The resulting distribution over sequence lengths, for $p = 0.99$ and various n. Generated by hmm_self_loop_dist.ipynb.

It is obvious from $\log p(\boldsymbol{\theta}) + \log p(\boldsymbol{y}_{1:T}, \boldsymbol{z}_{1:T}|\boldsymbol{\theta})$ that the posteriors of $\boldsymbol{\pi}$ and $\mathbf{A}_{j\cdot}$ are both still Dirichlet distributions. It can also be shown that the posterior distributions of $\{\mathbf{E}_j, \boldsymbol{\Sigma}_j\}$ are still matrix normal inverse Wishart distributions, whose hyperparameters can be directly obtained by replacing $\mathbf{Y}, \mathbf{A}, \mathbf{X}$ in Equation (15.105) with $\hat{\mathbf{Y}}_j$, $\mathbf{E}_j$ and $\tilde{\mathbf{Y}}_j$ respectively. To summarize, the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{z}_{1:T}, \boldsymbol{y}_{1:T})$ is:

$$\boldsymbol{\pi}|\boldsymbol{z}_{1:T} \sim \mathrm{Dir}(\widehat{\boldsymbol{\alpha}}_\pi), \qquad \widehat{\alpha}_{\pi,k} = \breve{\alpha}_\pi\ /K + \mathbb{I}\left(z_1 = k\right) \tag{29.47}$$

$$\mathbf{A}_{j\cdot}|\boldsymbol{z}_{1:T} \sim \mathrm{Dir}(\widehat{\boldsymbol{\alpha}}_A), \qquad \widehat{\alpha}_{A_j,k} = \breve{\alpha}_A\ /K + N_{jk} \tag{29.48}$$

$$\boldsymbol{\Sigma}_j|\boldsymbol{z}_{1:T}, \boldsymbol{y}_{1:T} \sim \mathrm{IW}(\widehat{\nu}_j, \widehat{\boldsymbol{\Psi}}_j) \qquad \mathbf{E}_j|\boldsymbol{\Sigma}_j, \boldsymbol{z}_{1:T}, \boldsymbol{y}_{1:T} \sim \mathcal{MN}(\widehat{\mathbf{M}}_j, \boldsymbol{\Sigma}_j, \widehat{\mathbf{V}}_j) \tag{29.49}$$

29.5 HMMs: generalizations

In this section, we discuss various extensions of the vanilla HMM introduced in Section 29.2.

29.5.1 Hidden semi-Markov model (HSMM)

In a standard HMM (Section 29.2), the probability we remain in state i for exactly d steps is

$$p(d_i = d) = (1 - A_{ii})A_{ii}^d \propto \exp(d \log A_{ii}) \tag{29.50}$$

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 29.15(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

$$p(d) = \binom{d-1}{n-1} p^{d-n}(1-p)^n \tag{29.51}$$

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 29.15(b).

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 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 29.16(a). The $d_t \in \{1, \ldots, D\}$ 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:

$$p(d_t = d'|d_{t-1} = d, z_t = j) = \begin{cases} D_j(d') & \text{if } d = 1 \\ 1 & \text{if } d' = d - 1 \text{ and } d > 1 \\ 0 & \text{otherwise} \end{cases} \tag{29.52}$$

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 (truncated) 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. (We assume $A_{jj} = 0$.) More precisely, we define the state CPD as follows:

$$p(z_t = k|z_{t-1} = j, d_{t-1} = d) = \begin{cases} 1 & \text{if } d > 0 \text{ and } j = k \\ A_{jk} & \text{if } d = 1 \\ 0 & \text{otherwise} \end{cases} \tag{29.53}$$

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 $p(\boldsymbol{y}_{t:t+l-1}|z_t = k, d_t = l)$, 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:

$$p(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{d}) = \left[p(z_1)p(d_1|z_1) \prod_{t=2}^{T} p(z_t|z_{t-1}, d_{t-1})p(d_t|z_t, d_{t-1}) \right] p(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{d}) \tag{29.54}$$

In a standard HSMM, we assume

$$p(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{d}) = \prod_{t=1}^{T} p(y_t|z_t) \tag{29.55}$$

so the duration variables only determine the hidden state dynamics. To define $p(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{d})$ for a segmental HMM, let us use s_i and e_i to denote the start and end times of segment i. This sequence

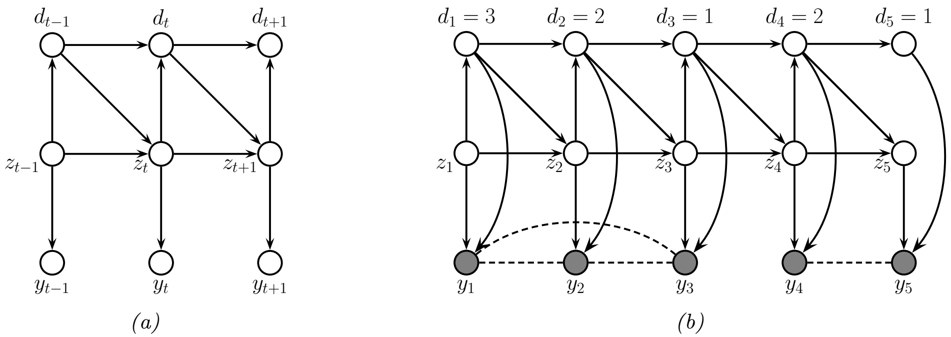


Figure 29.16: Encoding a hidden semi-Markov model as a DPGM. (a) d_t is a deterministic down counter (duration variable). Each observation is generated independently. (b) Similar to (a), except now we generate the observations within each segment as a block. In this figure, we represent the non-Markovian dependencies between the observations within each segment by using undirected edges. We represent the conditional independence between the observations across different segments by disconnecting $\boldsymbol{y}_{1:3}$ from $\boldsymbol{y}_{4:5}$; this can be enforced by "breaking the link" whenever $d_t = 1$ (representing the end of a segment).

can be computed deterministically from $\boldsymbol{d}$ 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:

$$p(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{d}) = \prod_{i=1}^{|\boldsymbol{s}|} p(\boldsymbol{y}_{s_i:e_i}|z_{s_i}, d_{s_i}) \tag{29.56}$$

See Figure 29.16(b) for the DPGM.

If we use an RNN for each segment, we have

$$p(\boldsymbol{y}_{s_i:e_i}|z_{s_i}, d_{s_i}) = \prod_{t=s_i}^{e_i} p(y_t|\boldsymbol{y}_{s_i:t-1}, z_{s_i}) = \prod_{t=s_i}^{e_i} p(y_t|h_t, z_{s_i}) \tag{29.57}$$

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 $p(z_t, d_t|\boldsymbol{y}_{1:T})$ 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 $q(\boldsymbol{z}, \boldsymbol{d}|\boldsymbol{y}) = \prod_t q(z_t|\boldsymbol{y})q(d_t|\boldsymbol{y})$, to compute the posterior in $O(TK + TD)$ time.

29.5.2 Hierarchical HMMs

A **hierarchical HMM** (HHMM) [FST98] is an extension of the HMM that is designed to model domains with hierarchical structure. Figure 29.17 gives an example of an HHMM used in automatic speech recognition, where words are composed of phones which are composed of subphones. We can always "flatten" an HHMM to a regular HMM, but a factored representation is often easier to interpret, and allows for more efficient inference and model fitting.

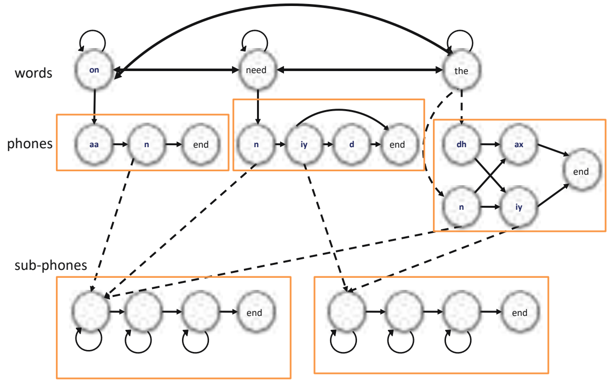


Figure 29.17: An example of an HHMM for an ASR system which can recognize 3 words. The top level represents bigram word probabilities. The middle level represents the phonetic spelling of each word. The bottom level represents the subphones of each phone. (It is traditional to represent a phone as a 3 state HMM, representing the beginning, middle and end; these are known as subphones.) Adapted from Figure 7.5 of [JM00].

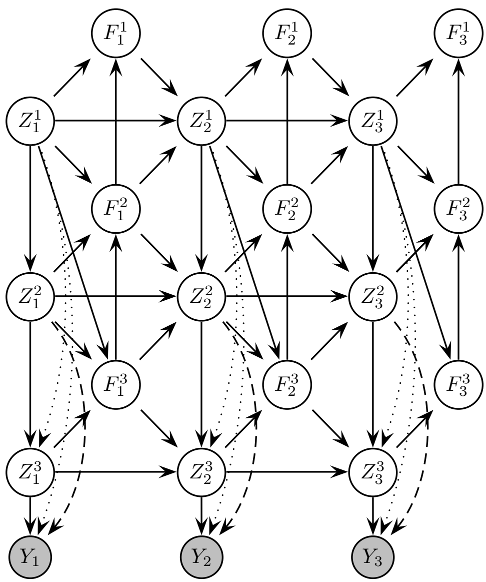


Figure 29.18: An HHMM represented as a DPGM. Z_t^ℓ is the state at time t, level ℓ; $F_t^\ell = 1$ if the HMM at level ℓ has finished (entered its exit state), otherwise $F_t^\ell = 0$. Shaded nodes are observed; the remaining nodes are hidden. We may optionally clamp $F_T^\ell = 1$, where T is the length of the observation sequence, to ensure all models have finished by the end of the sequence. From Figure 2 of [MP01].

HHMMs have been used in many application domains, e.g., speech recognition [Bil01], gene finding [Hu+00], plan recognition [BVW02], monitoring transportation patterns [Lia+07], indoor robot localization [TMK04], etc. HHMMs are less expressive than stochastic context free grammars (SCFGs) since they only allow hierarchies of bounded depth, but they support more efficient inference. In particular, inference in SCFGs (using the inside outside algorithm, [JM08]) takes $O(T^3)$ whereas inference in an HHMM takes $O(T)$ time [MP01; WM12].

We can represent an HHMM as a directed graphical model as shown in Figure 29.18. Q_t^ℓ represents the state at time t and level ℓ. A state transition at level ℓ is only "allowed" if the chain at the level below has "finished", as determined by the $F_t^{\ell-1}$ node. (The chain below finishes when it chooses to enter its end state.) This mechanism ensures that higher level chains evolve more slowly than lower level chains, i.e., lower levels are nested within higher levels.

A variable duration HMM can be thought of as a special case of an HHMM, where the top level is a deterministic counter, and the bottom level is a regular HMM, which can only change states once the counter has "timed out". See [MP01] for further details.

29.5.3 Factorial HMMs

An HMM represents the hidden state using a single discrete random variable $z_t \in \{1, \ldots, K\}$. To represent 10 bits of information would require $K = 2^{10} = 1024$ states. By contrast, consider a **distributed representation** of the hidden state, where each $z_{t,m} \in \{0, 1\}$ represents the m'th bit of the t'th hidden state. Now we can represent 10 bits using just 10 binary variables. This model is called a **factorial HMM** [GJ97].

More precisely, the model is defined as follows:

$$p(\boldsymbol{z}, \boldsymbol{y}) = \prod_t \left[\prod_m p(z_{tm}|z_{t-1,m}) \right] p(\boldsymbol{y}_t|\boldsymbol{z}_t) \tag{29.58}$$

where $p(z_{tm} = k|z_{t-1,m} = j) = A_{mjk}$ is an entry in the transition matrix for chain m, $p(z_{1m} = k|z_{0m}) = p(z_{1m} = k) = \pi_{mk}$, is the initial state distribution for chain m, and

$$p(\boldsymbol{y}_t|\boldsymbol{z}_t) = \mathcal{N}\left(\boldsymbol{y}_t | \sum_{m=1}^M \mathbf{W}_m \boldsymbol{z}_{tm}, \boldsymbol{\Sigma} \right) \tag{29.59}$$

is the observation model, where $\boldsymbol{z}_{tm}$ is a 1-of-K encoding of z_{tm} and $\mathbf{W}_m$ is a $D \times K$ matrix (assuming $\boldsymbol{y}_t \in \mathbb{R}^D$). Figure 29.19a illustrates the model for the case where $M = 3$.

An interesting application of FHMMs is to the problem of **energy disaggregation** [KJ12]. In this problem, we observe the total energy usage of a house at each moment in time, i.e., the observation model has the form

$$p(y_t|\boldsymbol{z}_t) = \mathcal{N}(y_t | \sum_{m=1}^M w_m z_{tm}, \sigma^2) \tag{29.60}$$

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

$$p(z_{t,m} = 1|z_{t-1,m}) = \begin{cases} A_{01} & \text{if } z_{t-1,m} = 0 \\ A_{11} & \text{if } z_{t-1,m} = 1 \end{cases} \tag{29.61}$$

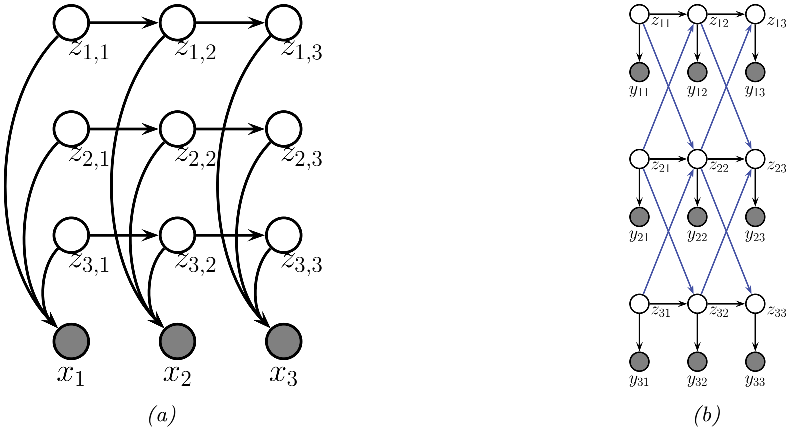


Figure 29.19: (a) A factorial HMM with 3 chains. (b) A coupled HMM with 3 chains.

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.

Unfortunately, conditioned on $\boldsymbol{y}_t$, all the hidden variables are correlated (due to explaining away the common observed child $\boldsymbol{y}_t$). This make exact state estimation intractable. However, we can derive efficient approximate inference algorithms, as we discuss in Supplementary Section 10.3.2.

29.5.4 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

$$p(\boldsymbol{z}_t, \boldsymbol{y}_t | \boldsymbol{z}_{t-1}) = \prod_m p(\boldsymbol{y}_{tm} | z_{tm}) p(z_{tm} | \boldsymbol{z}_{t-1,m-1:m+1}) \tag{29.62}$$

with boundary conditions defined in the obvious way. See Section 29.5.4 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 a 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.

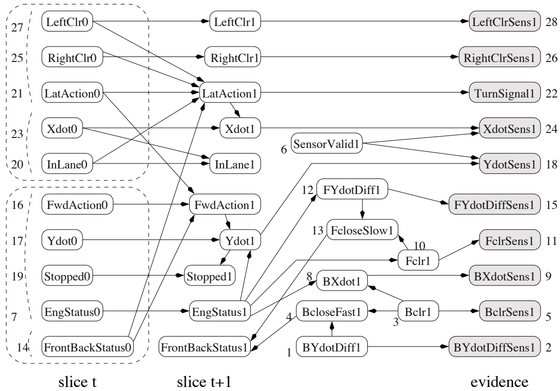


Figure 29.20: The BATnet DBN. The transient nodes are only shown for the second slice, to minimize clutter. The dotted lines are used to group related variables. Used with kind permission of Daphne Koller.

29.5.5 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 29.20, 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].

29.5.6 Changepoint detection

In this section, we discuss **changepoint detection**, which is the task of detecting when there are "abrupt" changes in the distribution of the observed values in a time series. We focus on the online case. (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.)

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 29.5.1. This is like an HMM in which we explicitly model the duration 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 1. 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

$$p(r_t|r_{t-1}) = \begin{cases} H(r_{t-1}+1) & \text{if } r_t = 0 \\ 1 - H(r_{t-1}+1) & \text{if } r_t = r_{t-1}+1 \\ 0 & \text{otherwise} \end{cases} \tag{29.63}$$

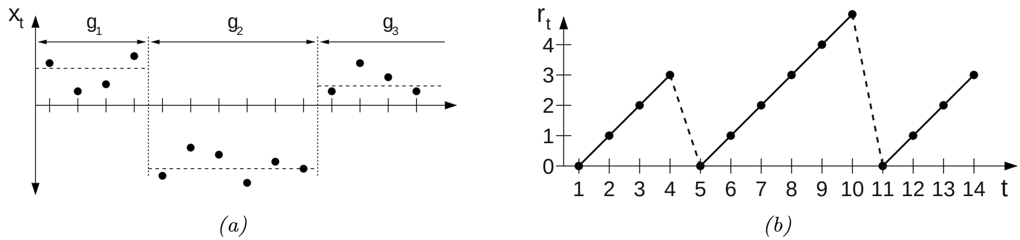


Figure 29.21: Illustration of Bayesian online changepoint detection (BOCPD). (a) Hypothetical segmentation of a univariate time series divided by changepoints on the mean into three segments of lengths $g_1 = 4$, $g_2 = 6$, and an undetermined length for g_3 (since it the third segment has not yet ended). From Figure 1 of [AM07]. Used with kind permission of Ryan Adams.

where $H(\tau)$ is a **hazard function**:

$$H(\tau) = \frac{p_g(\tau)}{\sum_{t=\tau}^{\infty} p_g(t)} \tag{29.64}$$

where $p_g(t)$ is the probability of a segment of length t. See Figure 29.21 for an illustration. If we set p_g to be a geometric distribution with parameter λ, 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):

$$p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, r_t = r, z_t = k) = p(\boldsymbol{y}_t|\boldsymbol{y}_{t-r:t-1}, z_t = k) = \int p(\boldsymbol{y}_t|\boldsymbol{\eta})p(\boldsymbol{\eta}|\boldsymbol{y}_{t-r:t-1}, z_t = k)d\boldsymbol{\eta} \tag{29.65}$$

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

$$p(\boldsymbol{\eta}|\boldsymbol{y}_{t-r:t-1}, z_t = k) \propto p(\boldsymbol{\eta}|z_t = k) \prod_{i=t-r}^{t-1} p(\boldsymbol{y}_i|\boldsymbol{\eta}) \tag{29.66}$$

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

$$\pi_t^{r,k} = p(\boldsymbol{y}_t|\boldsymbol{y}_{t-r:t-1}, z_t = k) = p(\boldsymbol{y}_t|\boldsymbol{\psi}_t^{r,k}) \tag{29.67}$$

where $\boldsymbol{\psi}_t^{r,k}$ 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]:

$$p(\boldsymbol{y}|\boldsymbol{r}) = p(\boldsymbol{y}_{s_1:e_1}) \dots p(\boldsymbol{y}_{s_N:e_N}) \tag{29.68}$$

where s_i and e_i are the start and end of segment i, which can be computed from the run lengths $\boldsymbol{r}$. (We initialize with $r_0 = 0$.) Thus there is no information sharing between segments. This can be

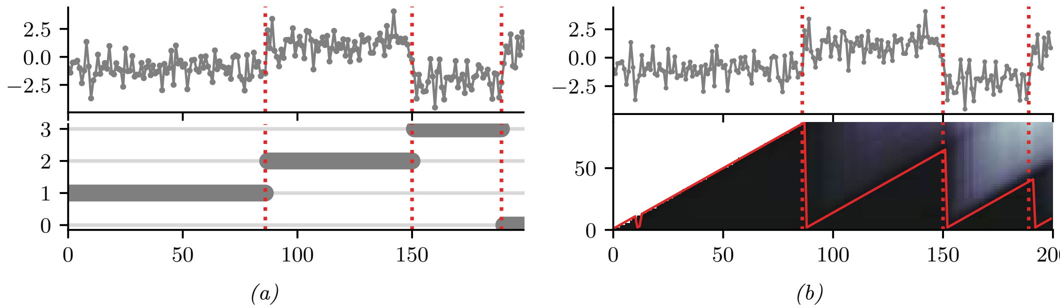


Figure 29.22: Illustration of BOCPD. (a) Synthetic data from a GMM with 4 states. Top row is the data, bottom row is the generating state. (b) Output of algorithm. Top row: Estimated changepoint locations. Bottom row: posterior predicted probability of a changepoint at each step. Generated by changepoint_detection.ipynb.

useful for time series 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:

$$p(r_t|\boldsymbol{y}_{1:t}) \propto p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, r_t)p(r_t|\boldsymbol{y}_{1:t-1}) \tag{29.69}$$

where we initialize with $p(r_0 = 0) = 1$. The marginal likelihood $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, r_t)$ is given by Equation (29.65) (with $z_t = 1$ dropped, since there is just one state). The prior predictive is given by

$$p(r_t|\boldsymbol{y}_{1:t-1}) = \sum_{r_{t-1}} p(r_t|r_{t-1})p(r_{t-1}|\boldsymbol{y}_{1:t-1}) \tag{29.70}$$

The one step ahead predictive distribution is given by

$$p(\boldsymbol{y}_{t+1}|\boldsymbol{y}_{1:t}) = \sum_{r_t} p(\boldsymbol{y}_{t+1}|\boldsymbol{y}_{1:t}, r_t)p(r_t|\boldsymbol{y}_{1:t}) \tag{29.71}$$

29.5.6.1 Example

We give an example of the method in Figure 29.22 applied to a synthetic 1d dataset generated from a 4 state GMM. The likelihood is a univariate Gaussian, $p(y_t|\mu) = \mathcal{N}(y_t|\mu, \sigma^2)$, where $\sigma^2 = 1$ is fixed, and μ is inferred using a Gaussian prior. The hazard function is set to a geometric distribution with rate N/T, where $N = 4$ is the true number of change points and $T = 200$ is the length of the sequence.

29.5.6.2 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.

In addition, the above method relies on a conjugate exponential model in order to compute the marginal likelihood, and update the posterior parameters for each r, in $O(1)$ time. For more complex models, we need to use approximations. In [TBS13], they use variational Bayes (Section 10.3.3), and in [Mav16], they use particle filtering (Section 13.2), which is more general, but much slower.

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 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

$$p(y_t|r_t, d_t, \eta) = \mathcal{N}(y_t|\boldsymbol{\phi}(r_t/d_t)^\mathsf{T}\boldsymbol{\eta}, \sigma^2) \tag{29.72}$$

where $0 \leq r_t/d_t \leq 1$, and $\boldsymbol{\phi}()$ 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.

29.6 Linear dynamical systems (LDSs)

In this section, we discuss **linear-Gaussian state-space model** (**LG-SSM**), also called **linear dynamical system** (**LDS**). This is a special case of an SSM in which the transition function and observation function are both linear, and the process noise and observation noise are both Gaussian.

29.6.1 Conditional independence properties

The LDS graphical model is shown in Figure 29.1(a). This encodes the assumption that the hidden states are Markovian, and the observations are iid conditioned on the hidden states. All that remains is to specify the form of the conditional probability distributions of each node.

29.6.2 Parameterization

An LDS model is defined as follows:

$$p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{z}_t|\mathbf{F}_t\boldsymbol{z}_{t-1} + \mathbf{B}_t\boldsymbol{u}_t + \boldsymbol{b}_t, \mathbf{Q}_t) \tag{29.73}$$
$$p(\boldsymbol{y}_t|\boldsymbol{z}_t, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{y}_t|\mathbf{H}_t\boldsymbol{z}_t + \mathbf{D}_t\boldsymbol{u}_t + \boldsymbol{d}_t, \mathbf{R}_t) \tag{29.74}$$

We often assume the bias (offset) terms are zero, in which case the model simplifies to

$$p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{z}_t|\mathbf{F}_t\boldsymbol{z}_{t-1} + \mathbf{B}_t\boldsymbol{u}_t, \mathbf{Q}_t) \tag{29.75}$$

$$p(\boldsymbol{y}_t|\boldsymbol{z}_t, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{y}_t|\mathbf{H}_t\boldsymbol{z}_t + \mathbf{D}_t\boldsymbol{u}_t, \mathbf{R}_t) \tag{29.76}$$

Furthermore, if there are no inputs, the model further simplifies to

$$p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) = \mathcal{N}(\boldsymbol{z}_t|\mathbf{F}_t\boldsymbol{z}_{t-1}, \mathbf{Q}_t) \tag{29.77}$$

$$p(\boldsymbol{y}_t|\boldsymbol{z}_t) = \mathcal{N}(\boldsymbol{y}_t|\mathbf{H}_t\boldsymbol{z}_t, \mathbf{R}_t) \tag{29.78}$$

We can also write this as a structural equation model (Section 4.7.2):

$$\boldsymbol{z}_t = \mathbf{F}_t\boldsymbol{z}_{t-1} + \boldsymbol{q}_t \tag{29.79}$$

$$\boldsymbol{y}_t = \mathbf{H}_t\boldsymbol{z}_t + \boldsymbol{r}_t \tag{29.80}$$

where $\boldsymbol{q}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$ is the **process noise**, and $\boldsymbol{r}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$ is the **observation noise**.

Typically we assume the parameters $\boldsymbol{\theta}_t = (\mathbf{F}_t, \mathbf{H}_t, \mathbf{B}_t, \mathbf{D}_t, \mathbf{Q}_t, \mathbf{R}_t)$ are independent of time, so the model is stationary. (We discuss how to learn the parameters in Section 29.8.) Given the parameters, we discuss how to perform online posterior inference of the latent states using the Kalman filter in Section 8.2.2, and offline inference using the Kalman smoother in Section 8.2.3.

29.7 LDS: applications

In this section, we discuss some applications of LDS models.

29.7.1 Object tracking and state estimation

Consider an object moving in $\mathbb{R}^2$. Let the state at time t be the position and velocity of the object, $\boldsymbol{z}_t = \begin{pmatrix} u_t & v_t & \dot{u}_t & \dot{v}_t \end{pmatrix}$. (We use u and v for the two coordinates, to avoid confusion with the state and observation variables.) We assume this evolves in continuous time according to a linear **stochastic differential equation** or SDE, in which the dynamics are given by Newton's law of motion, and where the random acceleration corresponds to a **white noise process** (aka **Brownian motion**). However, since the observations occur at discrete time steps t_k, we will only evaluate the system at discrete time points; this is called a **continuous-discrete SSM** [SS19, p199]. (We shall henceforth write t instead of t_k, since in this book we only consider discrete time.) The corresponding discrete time SSM is given by the following [SS19, p82]:

$$\underbrace{\begin{pmatrix} u_t \\ v_t \\ \dot{u}_t \\ \dot{v}_t \end{pmatrix}}_{\boldsymbol{z}_t} = \underbrace{\begin{pmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{F}} \underbrace{\begin{pmatrix} u_{t-1} \\ v_{t-1} \\ \dot{u}_{t-1} \\ \dot{v}_{t-1} \end{pmatrix}}_{\boldsymbol{z}_{t-1}} + \boldsymbol{q}_t \tag{29.81}$$

where Δ is the step size between consecutive discrete measurement times, $\boldsymbol{q}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ is the process noise, and the noise covariance matrix $\mathbf{Q}$ is given by

$$\mathbf{Q} = \begin{pmatrix} \frac{q_1\Delta^3}{3} & 0 & \frac{q_1\Delta^2}{2} & 0 \\ 0 & \frac{q_2\Delta^3}{3} & 0 & \frac{q_2\Delta^2}{2} \\ \frac{q_1\Delta^2}{2} & 0 & q_1\Delta & 0 \\ 0 & \frac{q_2\Delta^2}{2} & 0 & q_2\Delta \end{pmatrix}$$

where q_i are the diffusion coefficients of the white noise process for dimension i (see [SS19, p44] for details).

Now suppose that at each discrete time point we observe the location, corrupted by Gaussian noise. Thus the observation model becomes

$$\underbrace{\begin{pmatrix} y_{1,t} \\ y_{2,t} \end{pmatrix}}_{\boldsymbol{y}_t} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\mathbf{H}} \underbrace{\begin{pmatrix} u_t \\ \dot{u}_t \\ v_t \\ \dot{v}_t \end{pmatrix}}_{\boldsymbol{z}_t} + \boldsymbol{r}_t \tag{29.82}$$

where $\boldsymbol{r}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ is the **observation noise**. We see that the observation matrix $\mathbf{H}$ simply "extracts" the relevant parts of the state vector.

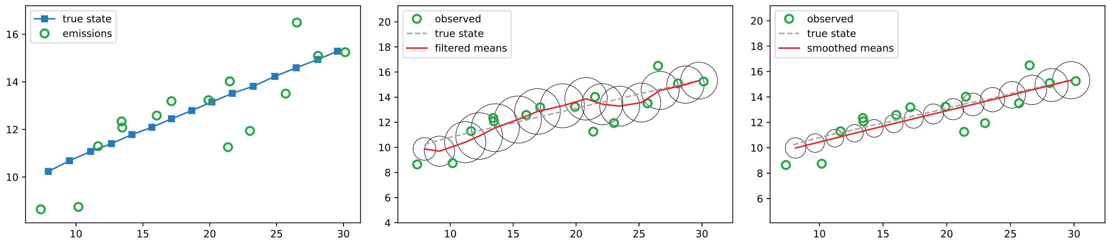
Suppose we sample a trajectory and corresponding set of noisy observations from this model, $(\boldsymbol{z}_{1:T}, \boldsymbol{y}_{1:T}) \sim p(\boldsymbol{z}, \boldsymbol{y}|\boldsymbol{\theta})$. (We use diagonal observation noise, $\mathbf{R} = \text{diag}(\sigma_1^2, \sigma_2^2)$.) The results are shown in Figure 29.23(a). We can use the Kalman filter (Section 8.2.2) to compute $p(\boldsymbol{z}_t|\boldsymbol{y}_{1:t}, \boldsymbol{\theta})$ for each t,. (We initialize the filter with a vague prior, namely $p(\boldsymbol{z}_0) = \mathcal{N}(\boldsymbol{z}_0|\mathbf{0}, 10^5\mathbf{I})$.) The results are shown in Figure 29.23(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 8.2.3) to compute $p(\boldsymbol{z}_t|\boldsymbol{y}_{1:T}, \boldsymbol{\theta})$, where we condition on all the data, past and future. The results are shown in Figure 29.23(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.)

29.7.2 Online Bayesian linear regression (recursive least squares)

In Section 15.2.2, we discuss how to compute $p(\boldsymbol{w}|\sigma^2, \mathcal{D})$ for a linear regression model in batch mode, using a Gaussian prior of the form $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$. In this section, we discuss how to compute this posterior online, by repeatedly performing the following update:

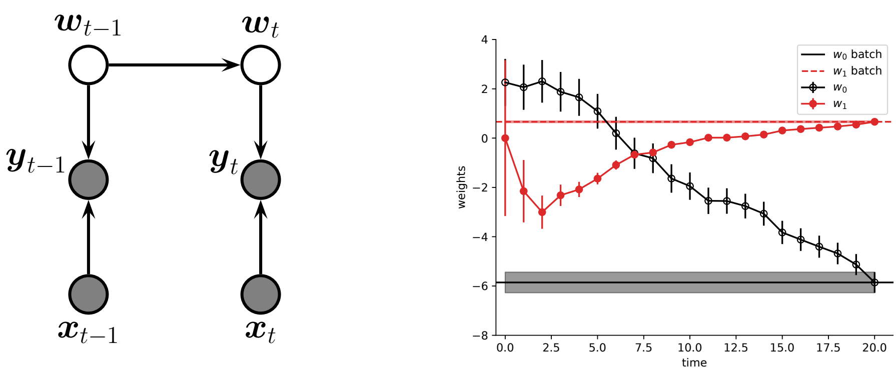
$$p(\boldsymbol{w}|\mathcal{D}_{1:t}) \propto p(\mathcal{D}_t|\boldsymbol{w})p(\boldsymbol{w}|\mathcal{D}_{1:t-1}) \tag{29.83}$$
$$\propto p(\mathcal{D}_t|\boldsymbol{w})p(\mathcal{D}_{t-1}|\boldsymbol{w}) \ldots p(\mathcal{D}_1|\boldsymbol{w})p(\boldsymbol{w}) \tag{29.84}$$

where $\mathcal{D}_t = (\boldsymbol{u}_t, y_t)$ 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 σ^2.) We see that the previous posterior, $p(\boldsymbol{w}|\mathcal{D}_{1:t-1})$, becomes the current prior, which gets updated by $\mathcal{D}_t$ to become the new posterior, $p(\boldsymbol{w}|\mathcal{D}_{1:t})$. This is an example of sequential Bayesian updating or online Bayesian inference. In the case of linear regression, this process is known as the **recursive least squares** or **RLS** algorithm.



(a) (b) (c)

Figure 29.23: Illustration of Kalman filtering and smoothing for a linear dynamical system. (Repeated from Figure 8.2.) (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. Circles are 95% confidence ellipses, whose center is the posterior mean, and whose shape is 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_script.ipynb.



(a) (b)

Figure 29.24: (a) A dynamic generalization of linear regression. (b) Illustration of the recursive least squares algorithm applied to the model $p(y|x, \boldsymbol{w}) = \mathcal{N}(y|w_0 + w_1 x, \sigma^2)$. We plot the marginal posterior of w_0 and w_1 vs number of datapoints. (Error bars represent $\mathbb{E}[w_j|\boldsymbol{y}_{1:t}, \boldsymbol{x}_{1:t}] \pm \sqrt{\mathbb{V}[w_j|\boldsymbol{y}_{1:t}, \boldsymbol{x}_{1:t}]}$.) 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 kf_linreg.ipynb.

We can implement this method by using a linear Gaussian state-space model(Section 29.6). The basic idea is to let the hidden state represent the regression parameters, and to let the (time-varying) observation model $\mathbf{H}_t$ represent the current feature vector $\boldsymbol{x}_t$.[3] That is, the observation model has the form

$$p(y_t|\boldsymbol{w}_t) = \mathcal{N}(y_t|\mathbf{H}_t \boldsymbol{z}_t, \mathbf{R}_t) = \mathcal{N}(y_t|\boldsymbol{x}_t^\mathsf{T} \boldsymbol{w}_t, \sigma^2) \tag{29.85}$$

If we assume the regression parameters do not change, the dynamics model becomes

$$p(\boldsymbol{w}_t|\boldsymbol{w}_{t-1}) = \mathcal{N}(\boldsymbol{w}_t|\boldsymbol{w}_{t-1}, 0) = \delta(\boldsymbol{w}_t - \boldsymbol{w}_{t-1}) \tag{29.86}$$

(If we do let the parameters change over time, we get a so-called **dynamic linear model** [Har90; WH97; PPC09].) See Figure 29.24a for the model, and Supplementary Section 8.1.2 for a simplification of the Kalman filter equations when applied to this special case.

We show a 1d example in Figure 29.24b. We see that online inference converges to the exact batch (offline) posterior in a single pass over the data.

If we approximate the Kalman gain matrix by $\mathbf{K}_t \approx \eta_t \mathbf{1}$, we recover the **least mean squares** or **LMS** algorithm, where η_t is the learning rate. In LMS, we need to adapt the learning rate to ensure convergence to the MLE. Furthermore, the algorithm may require multiple passes through the data to converge to this global optimum. By contrast, the RLS algorithm automatically performs step-size adaptation, and converges to the optimal posterior in a single pass over the data.

In Section 8.6.3, we extend this approach to perform online parameter estimation for logistic regression, and in Section 17.5.2, we extend this approach to perform online parameter estimation for MLPs.

29.7.3 Adaptive filtering

Consider an autoregressive model of order D:

$$y_t = w_1 y_{t-1} + \cdots + w_D y_{t-D} + \epsilon_t \tag{29.87}$$

where $\epsilon_t \sim \mathcal{N}(0, 1)$. The problem of **adaptive filtering** is to estimate the parameters $\boldsymbol{w}_{1:D}$ given the observations $\boldsymbol{y}_{1:t}$.

We can cast this as inference in an LG-SSM by defining the observation matrix to be $\mathbf{H}_t = (y_{t-1} \dots y_{t-D})$ and defining the state as $\boldsymbol{z}_t = \boldsymbol{w}$. We can also allow the parameters to evolve over time, similar to Section 29.7.2.

29.7.4 Time series forecasting

In Section 29.12 we discuss how to use LDS models to perform time series forecasting.

3. It is tempting to think we can just set the input $\boldsymbol{u}_t$ to the covariates $\boldsymbol{x}_t$. Unfortunately this will not work, since the effect of the inputs is to add an offset term to the output in a way which is independent of the hidden state (parameters). That is, since we have $\boldsymbol{y}_t = \mathbf{H}_t \boldsymbol{z}_t + \mathbf{D}_t \boldsymbol{u}_t + \boldsymbol{d} + \boldsymbol{r}_t$, if we set $\boldsymbol{u}_t = \boldsymbol{x}_t$ then the features get multiplied by the constant LDS parameter $\mathbf{D}_t$ instead of the hidden state $\boldsymbol{z}_t$ containing the regression weights.

29.8 LDS: parameter learning

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 29.4. For example, we can use EM, SGD, spectral methods, or Bayesian methods, as we discuss below. (For more details, see [Sar13, Ch 12].)

29.8.1 EM for LDS

This section is coauthored with Xinglong Li.

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 29.4.1), except we use Kalman smoothing (Section 8.2.3) instead of forwards-backwards in the E step, and use different calculations in the M step. The details can be found in [SS82; GH96a]. Here we extend these results to consider the case where the HMM may have an optional input sequence $\boldsymbol{u}_{1:T}$.

Our goal is to maximize the expected complete data log likelihood

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}_{k-1}) = \mathbb{E}\left[\log p(\boldsymbol{z}_{1:T}, \boldsymbol{y}_{1:T}|\boldsymbol{\theta})|\boldsymbol{y}_{1:T}, \boldsymbol{u}_{1:T}, \boldsymbol{\theta}_{k-1}\right] \tag{29.88}$$

where the expectations are taken wrt the parameters at the previous iteration $k-1$ of the algorithm. (We initialize with random parameters, or by first fitting a simpler model, such as factor analysis.) For brevity of notations, we assume that the bias terms are included in $\mathbf{D}$ and $\mathbf{B}$ (i.e., the last entry of $\boldsymbol{u}_t$ is 1).

The log joint is given by the following:

$$\log p(\boldsymbol{z}_{1:T}, \boldsymbol{y}_{1:T}|\boldsymbol{\theta}) = -\sum_{t=1}^{T}\left(\frac{1}{2}(\boldsymbol{y}_t - \mathbf{H}\boldsymbol{z}_t - \mathbf{D}\boldsymbol{u}_t)^{\mathsf{T}}\mathbf{R}^{-1}(\boldsymbol{y}_t - \mathbf{H}\boldsymbol{z}_t - \mathbf{D}\boldsymbol{u}_t)\right) - \frac{T}{2}\log|\mathbf{R}| \tag{29.89}$$

$$-\sum_{t=2}^{T}\left(\frac{1}{2}(\boldsymbol{z}_t - \mathbf{F}\boldsymbol{z}_{t-1} - \mathbf{B}\boldsymbol{u}_t)^{\mathsf{T}}\mathbf{Q}^{-1}(\boldsymbol{z}_t - \mathbf{F}\boldsymbol{z}_{t-1} - \mathbf{B}\boldsymbol{u}_t)\right) - \frac{T-1}{2}\log|\mathbf{Q}| \tag{29.90}$$

$$-\frac{1}{2}(\boldsymbol{z}_1 - \boldsymbol{m}_1)^{\mathsf{T}}\mathbf{V}_1^{-1}(\boldsymbol{z}_1 - \boldsymbol{m}_1) - \frac{1}{2}\log|\mathbf{V}_1| + \text{const} \tag{29.91}$$

where the prior on the initial state is $p(\boldsymbol{z}_1) = \mathcal{N}(\boldsymbol{z}_1|\boldsymbol{m}_1, \mathbf{V}_1)$.

In the E step, we can run the Kalman smoother to compute $\boldsymbol{\mu}_{t|T} = \mathbb{E}\left[\boldsymbol{z}_t|\boldsymbol{y}_{1:T}\right]$, and $\boldsymbol{\Sigma}_{t|T} = \text{Cov}\left[\boldsymbol{z}_t|\boldsymbol{y}_{1:T}\right]$, from which we can compute $\hat{\boldsymbol{z}}_t = \boldsymbol{\mu}_{t|T}$ and

$$\mathbf{P}_t = \mathbb{E}\left[\boldsymbol{z}_t\boldsymbol{z}_t^{\mathsf{T}}|\boldsymbol{y}_{1:T}\right] = \boldsymbol{\Sigma}_{t|T} + \boldsymbol{\mu}_{t|T}\boldsymbol{\mu}_{t|T}^{\mathsf{T}} \tag{29.92}$$

We also need to compute the cross term

$$\mathbf{P}_{t,t-1} = \mathbb{E}\left[\boldsymbol{z}_t\boldsymbol{z}_{t-1}^{\mathsf{T}}|\boldsymbol{y}_{1:T}\right] = \boldsymbol{\Sigma}_{t,t-1|T} + \boldsymbol{\mu}_{t|T}\boldsymbol{\mu}_{t-1|T}^{\mathsf{T}} \tag{29.93}$$

where

$$\boldsymbol{\Sigma}_{t-1,t-2|T} = \boldsymbol{\Sigma}_{t-1|t-1}\mathbf{J}_{t-2}^{\mathsf{T}} + \mathbf{J}_{t-1}(\boldsymbol{\Sigma}_{t,t-1|T} - \mathbf{F}\boldsymbol{\Sigma}_{t-1|t-1})\mathbf{J}_{t-2}^{\mathsf{T}} \tag{29.94}$$

where $\mathbf{J}_t$ is the backwards Kalman gain matrix defined in Equation (8.67). We initialize this using $\boldsymbol{\Sigma}_{T,T-1|T} = (\mathbf{I} - \mathbf{K}_T\mathbf{H})\mathbf{F}\boldsymbol{\Sigma}_{T-1|T-1}$, where $\mathbf{K}_T$ is the forwards Kalman gain matrix defined in Equation (8.28).

We can derive the M step as follows, using standard matrix calculus. We denote $\mathbf{A}_{\text{out}} = [\mathbf{H}, \mathbf{D}]$, $\hat{\boldsymbol{x}}_{\text{out},t} = [\hat{\boldsymbol{z}}_t^T, \boldsymbol{u}_t^T]^T$, $\mathbf{A}_{\text{dyn}} = [\mathbf{F}, \mathbf{B}]$, ,$\hat{\boldsymbol{x}}_{\text{dyn},t} = [\hat{\boldsymbol{z}}_{t-1}^T, \boldsymbol{u}_t^T]^T$, and

$$\mathbf{P}_{\text{out},t} = \begin{pmatrix} \mathbf{P}_t & \hat{\boldsymbol{z}}_t\boldsymbol{u}_t^{\mathsf{T}} \\ \boldsymbol{u}_t\hat{\boldsymbol{z}}_t^{\mathsf{T}} & \boldsymbol{u}_t\boldsymbol{u}_t^{\mathsf{T}} \end{pmatrix}, \qquad \mathbf{P}_{\text{dyn},t} = \begin{pmatrix} \mathbf{P}_{t-1} & \hat{\boldsymbol{z}}_{t-1}\boldsymbol{u}_t^{\mathsf{T}} \\ \boldsymbol{u}_t\hat{\boldsymbol{z}}_{t-1}^{\mathsf{T}} & \boldsymbol{u}_t\boldsymbol{u}_t^{\mathsf{T}} \end{pmatrix}. \tag{29.95}$$

- Output matrices:

$$\frac{\partial \mathcal{Q}}{\partial \mathbf{A}_{\text{out}}} = \sum_{t=1}^{T} \mathbf{R}^{-1}\boldsymbol{y}_t\hat{\boldsymbol{x}}_{\text{out},t}^{\mathsf{T}} - \sum_{t=1}^{T} \mathbf{R}^{-1}\mathbf{A}_{\text{out}}\mathbf{P}_{\text{out},t} = \mathbf{0} \tag{29.96}$$

$$\mathbf{A}_{\text{out}}^{\text{new}} = \left(\sum_{t=1}^{T} \boldsymbol{y}_t\hat{\boldsymbol{x}}_{\text{out},t}^{\mathsf{T}}\right)\left(\sum_{t=1}^{T} \mathbf{P}_{\text{out},t}\right)^{-1} \tag{29.97}$$

- Output noise covariance:

$$\frac{\partial \mathcal{Q}(\mathbf{A}_{\text{out}} = \mathbf{A}_{\text{out}}^{\text{new}})}{\partial \mathbf{R}^{-1}} = \frac{T}{2}\mathbf{R} - \frac{1}{2}\sum_{t=1}^{T}\left(\boldsymbol{y}_t\boldsymbol{y}_t^{\mathsf{T}} - 2\mathbf{A}_{\text{out}}^{\text{new}}\hat{\boldsymbol{x}}_{\text{out},t}\boldsymbol{y}_t^{\mathsf{T}} + \mathbf{A}_{\text{out}}^{\text{new}}\mathbf{P}_{\text{out},t}\mathbf{A}_{\text{out}}^{\text{new},\mathsf{T}}\right) = \mathbf{0} \tag{29.98}$$

$$\mathbf{R}^{\text{new}} = \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{y}_t\boldsymbol{y}_t^{\mathsf{T}} - \mathbf{A}_{\text{out}}^{\text{new}}\hat{\boldsymbol{x}}_{\text{out},t}\boldsymbol{y}_t^{\mathsf{T}}) \tag{29.99}$$

- System dynamics matrices:

$$\frac{\partial \mathcal{Q}}{\partial \mathbf{A}_{\text{dyn}}} = -\sum_{t=2}^{T} \mathbf{Q}^{-1}\left[\mathbf{P}_{t,t-1}, \hat{\boldsymbol{z}}_t\boldsymbol{u}_t^{\mathsf{T}}\right] + \sum_{t=2}^{T} \mathbf{Q}^{-1}\mathbf{A}_{\text{dyn}}\mathbf{P}_{\text{dyn},t} = \mathbf{0} \tag{29.100}$$

$$\mathbf{A}_{\text{dyn}}^{\text{new}} = \left(\sum_{t=2}^{T}[\mathbf{P}_{t,t-1}, \hat{\boldsymbol{z}}_t\boldsymbol{u}_t^{\mathsf{T}}]\right)\left(\sum_{t=2}^{T}\mathbf{P}_{\text{dyn},t}\right)^{-1} \tag{29.101}$$

- State noise covariance:

$$\frac{\partial \mathcal{Q}(\mathbf{A}_{\text{dyn}} = \mathbf{A}_{\text{dyn}}^{\text{new}})}{\partial \mathbf{Q}^{-1}} = \frac{T-1}{2}\mathbf{Q} - \frac{1}{2}\sum_{t=2}^{T}(\mathbf{P}_t - 2\mathbf{A}_{\text{dyn}}^{\text{new}}\left[\mathbf{P}_{t-1,t}, \boldsymbol{u}_t\hat{\boldsymbol{z}}_t^{\mathsf{T}}\right] + \mathbf{A}_{\text{dyn}}^{\text{new}}\mathbf{P}_{\text{dyn},t}\mathbf{A}_{\text{dyn}}^{\text{new},\mathsf{T}}) \tag{29.102}$$

$$= \frac{T-1}{2}\mathbf{Q} - \frac{1}{2}\sum_{t=2}^{T}\left(\mathbf{P}_t - \mathbf{A}_{\text{dyn}}^{\text{new}}\left[\mathbf{P}_{t-1,t}, \boldsymbol{u}_t\hat{\boldsymbol{z}}_t^{\mathsf{T}}\right]\right) = \mathbf{0} \tag{29.103}$$

$$\mathbf{Q}^{\text{new}} = \frac{1}{T-1}\sum_{t=2}^{T}\left(\mathbf{P}_t - \mathbf{A}_{\text{dyn}}^{\text{new}}\left[\mathbf{P}_{t-1,t}, \boldsymbol{u}_t\hat{\boldsymbol{z}}_t^{\mathsf{T}}\right]\right) \tag{29.104}$$

- Initial state mean:

$$\frac{\partial \mathcal{Q}}{\partial \boldsymbol{m}} = (\hat{\boldsymbol{z}}_1 - \boldsymbol{m})\mathbf{V}_1^{-1} = \mathbf{0} \tag{29.105}$$

$$\boldsymbol{m}^{\text{new}} = \hat{\boldsymbol{z}}_1 \tag{29.106}$$

- Initial state covariance:

$$\frac{\partial \mathcal{Q}}{\partial \mathbf{V}_1^{-1}} = \frac{\mathbf{V}_1}{2} - \frac{1}{2}(\mathbf{P}_1 - \hat{\boldsymbol{z}}_1\boldsymbol{m}_1^{\mathsf{T}} - \boldsymbol{m}_1\hat{\boldsymbol{z}}_1^{\mathsf{T}} + \boldsymbol{m}_1\boldsymbol{m}_1^{\mathsf{T}}) = \mathbf{0} \tag{29.107}$$

$$\mathbf{V}_1^{\text{new}} = \mathbf{P}_1 - \hat{\boldsymbol{z}}_1\hat{\boldsymbol{z}}_1^{\mathsf{T}} \tag{29.108}$$

Note that computing these expected sufficient statistics 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). Alternatively, if we have multiple processors, we can perform Kalman smoothing in $O(\log T)$ time using parallel scan operations (Section 8.2.3.4).

29.8.2 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].

To understand this approach, let us initially assume there is no observation noise and no system noise. In this case, we have $\boldsymbol{z}_t = \mathbf{F}\boldsymbol{z}_{t-1}$ and $\boldsymbol{y}_t = \mathbf{H}\boldsymbol{z}_t$, and hence $\boldsymbol{y}_t = \mathbf{H}\mathbf{F}^{t-1}\boldsymbol{z}_1$. Consequently all the observations must be generated from a $\dim(\boldsymbol{z}_t)$-dimensional linear manifold or subspace. We can identify this subspace using PCA. Once we have an estimate of the $\boldsymbol{z}_t$'s, we can fit the model as if it were fully observed. We can either use these estimates in their own right, or use them to initialize EM. Several papers (e.g., [Smi+00; BK15]) have shown that initializing EM this way gives much better results than initializing EM at random, or just using SSID without EM.

Although the theory only works for noise-free data, we can try to estimate the system noise covariance $\mathbf{Q}$ from the residuals in predicting $\boldsymbol{z}_t$ from $\boldsymbol{z}_{t-1}$, and to estimate the observation noise covariance $\mathbf{R}$ from the residuals in predicting $\boldsymbol{y}_t$ from $\boldsymbol{z}_t$. We can either use these estimates in their own right, or use them to initialize EM. Because this method relies on taking an SVD, it is called a **spectral estimation method**. Similar methods can also be used for HMMs (see Section 29.4.3).

29.8.3 Ensuring stability of the dynamical system

When estimating the dynamics matrix $\mathbf{F}$, it is very useful to impose a constraint on its eigenvalues. To see why this is important, consider the case of no system noise. In this case, the hidden state at time t is given by

$$\boldsymbol{z}_t = \mathbf{F}^t\boldsymbol{z}_1 = \mathbf{U}\boldsymbol{\Lambda}^t\mathbf{U}^{-1}\boldsymbol{z}_1 \tag{29.109}$$

where $\mathbf{U}$ is the matrix of eigenvectors for $\mathbf{F}$, and $\boldsymbol{\Lambda} = \text{diag}(\lambda_i)$ contains the eigenvalues. If any $\lambda_i > 1$, then for large t, $\boldsymbol{z}_t$ will blow up in magnitude. Consequently, to ensure stability, it is useful to require that all the eigenvalues are less than 1 [SBG07]. Of course, if all the eigenvalues are less than 1, then $\mathbb{E}[\boldsymbol{z}_t] = \mathbf{0}$ for large t, so the state will return to the origin. Fortunately, when we add noise, the state becomes non-zero, so the model does not degenerate.

29.8.4 Bayesian LDS

SSMs can be quite sensitive to their parameter values, which is a particular concern when they are used for forecasting applications (see Section 29.12.1), or when the latent states or parameters are interpreted for scientific purposes (see e.g., [AM+16]). In such cases, it is wise to represent our uncertainty about the parameters by using Bayesian inference.

There are various algorithms we can use to perform this task. For linear-Gaussian SSMs, it is possible to use variational Bayes EM [Bea03; BC07] (see Section 10.3.5), or blocked Gibbs sampling (see Section 29.8.4.1). Note, however, that $\boldsymbol{\theta}$ and $\boldsymbol{z}$ are highly correlated, so the mean field approximation can be inaccurate, and the blocked Gibbs method can mix slowly. It is also possible to use collapsed MCMC in which we marginalize out $\boldsymbol{z}_{1:T}$ and just work with $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:T})$, which we can sample using HMC.

29.8.4.1 Blocked Gibbs sampling for LDS

This section is written by Xinglong Li.

In this section, we discuss blocked Gibbs sampling for LDS [CK94b; CMR05; FS07]. We alternate between sampling from $p(\boldsymbol{z}_{1:T}|\boldsymbol{y}_{1:T}, \boldsymbol{\theta})$ using the forwards-filter backwards-sampling algorithm (Section 8.2.3.5), and sampling from $p(\boldsymbol{\theta}|\boldsymbol{z}_{1:T}, \boldsymbol{y}_{1:T})$, which is easy to do if we use conjugate priors.

In more detail, we will consider the following linear Gaussian state space model with homogeneous parameters:

$$p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{z}_t|\mathbf{F}\boldsymbol{z}_{t-1} + \mathbf{B}\boldsymbol{u}_t, \mathbf{Q}) \tag{29.110}$$

$$p(\boldsymbol{y}_t|\boldsymbol{z}_t, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{y}_t|\mathbf{H}\boldsymbol{z}_t + \mathbf{D}\boldsymbol{u}_t, \mathbf{R}) \tag{29.111}$$

The set of all the parameters is $\boldsymbol{\theta} = \{\mathbf{F}, \mathbf{H}, \mathbf{B}, \mathbf{D}, \mathbf{Q}, \mathbf{R}\}$. For the sake of simplicity, we assume that the regression coefficient matrix $\mathbf{B}$ and $\mathbf{D}$ include the intercept term (i.e., the last entry of $\boldsymbol{u}_t = 1$).

We use conjugate MNIW priors, as in Bayesian multivariate linear regression Section 15.2.9. Specifically,

$$p(\mathbf{Q}, [\mathbf{F}, \mathbf{B}]) = \text{MNIW}(\mathbf{M}_{z0}, \mathbf{V}_{z0}, \nu_{q0}, \boldsymbol{\Psi}_{q0}) \tag{29.112}$$

$$p(\mathbf{R}, [\mathbf{H}, \mathbf{D}]) = \text{MNIW}(\mathbf{M}_{y0}, \mathbf{V}_{y0}, \nu_{r0}, \boldsymbol{\Psi}_{r0}) \tag{29.113}$$

Given $\boldsymbol{z}_{1:T}$, $\boldsymbol{u}_{1:T}$ and $\boldsymbol{y}_{1:T}$, the posteriors are also MNIW. Specifically,

$$\mathbf{Q}|\boldsymbol{z}_{1:T}, \boldsymbol{u}_{1:T} \sim \text{IW}(\nu_{q1}, \boldsymbol{\Psi}_{q1}) \tag{29.114}$$

$$[\mathbf{F}, \mathbf{B}]|\mathbf{Q}, \boldsymbol{z}_{1:T}, \boldsymbol{u}_{1:T} \sim \mathcal{MN}(\mathbf{M}_{z1}, \mathbf{Q}, \mathbf{V}_{z1}) \tag{29.115}$$

where the set of hyperparameters $\{\mathbf{M}_{z1}, \mathbf{V}_{z1}, \nu_{q_1}, \boldsymbol{\Psi}_{q1}\}$ of the posterior MNIW can be obtained by replacing $\mathbf{Y}, \mathbf{A}, \mathbf{X}$ in Equation (15.105) with $\boldsymbol{z}_{2:T}, [\mathbf{F}, \mathbf{B}]$, and $[\boldsymbol{z}_{t-1}^{\mathsf{T}}, \boldsymbol{u}_t^{\mathsf{T}}]_{t=2:T}^{\mathsf{T}}$, respectively. Similarly,

$$\mathbf{R}|\boldsymbol{z}_{1:T}, \boldsymbol{u}_{1:T}, \boldsymbol{y}_{1:T} \sim \text{IW}(\nu_{r1}, \boldsymbol{\Psi}_{r1}) \tag{29.116}$$

$$[\mathbf{H}, \mathbf{D}]|\mathbf{R}, \boldsymbol{z}_{1:T}, \boldsymbol{u}_{1:T}, \boldsymbol{y}_{1:T} \sim \mathcal{MN}(\mathbf{M}_{y1}, \mathbf{R}, \mathbf{V}_{y1}), \tag{29.117}$$

and the hyperparameters $\{\mathbf{M}_{y1}, \mathbf{V}_{y1}, \nu_{r1}, \mathbf{\Psi}_{r1}\}$ of the posterior MNIW can be obtained by replacing $\mathbf{Y}, \mathbf{A}, \mathbf{X}$ in Equation (15.105) with $\boldsymbol{y}_{1:T}, [\mathbf{H}, \mathbf{D}]$, and $[\boldsymbol{y}_t^\mathsf{T}, \boldsymbol{u}_t^\mathsf{T}]_{1:T}^\mathsf{T}$.

29.8.5 Online parameter learning for SSMs

For many applications, we need to estimate the parameters of the SSM (such as the transition noise $\mathbf{Q}$ and observation noise $\mathbf{R}$) online, so that we can track non-stationary environments, etc. This problem is known as **adaptive filtering**. For some classical methods (based on moment matching), see [JB67; Meh72]. For an online, recursive MLE method based on the derivative of the particle filter, see [ADT12]. For a recent online variational Bayes approach, see [Cam+21; Hua+18c; VW21] and the references therein.

29.9 Switching linear dynamical systems (SLDSs)

Consider a state-space modelin which the latent state has both a discrete latent variable, $m_t \in \{1, \ldots, K\}$, and a continuous latent variable, $\boldsymbol{z}_t \in \mathbb{R}^{N_z}$. (A model with discrete and continuous latent variables is known as a **hybrid system** in control theory.) We assume the observed responses are continuous, $\boldsymbol{y}_t \in \mathbb{R}^{N_y}$. We may also have continuous observed inputs $\boldsymbol{u}_t \in \mathbb{R}^{N_u}$. The discrete variable can be used to represent different kinds of system dynamics or operating regimes (e.g., normal or abnormal), or different kinds of observation models (e.g., to handle outliers due to sensor noise or failures). If the system is linear-Gaussian, it is called a **switching linear dynamical system** (**SLDS**), a **regime switching Markov model** [Ham90; KN98], or a **jump Markov linear system** (**JMLS**) [DGK01].

29.9.1 Parameterization

An SLDS model is defined as follows:

$$p(m_t = k | m_{t-1} = j) = A_{jk} \tag{29.118}$$

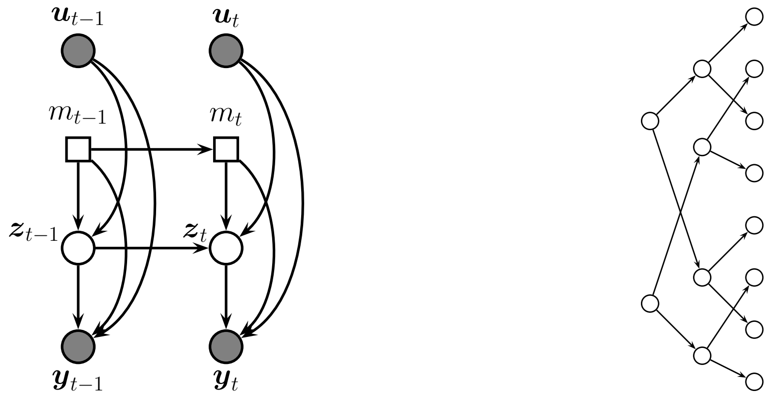
$$p(\boldsymbol{z}_t | \boldsymbol{z}_{t-1}, m_t = k, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{z}_t | \mathbf{F}_k \boldsymbol{z}_{t-1} + \mathbf{B}_k \boldsymbol{u}_t + \boldsymbol{b}_k, \mathbf{Q}_k) \tag{29.119}$$

$$p(\boldsymbol{y}_t | \boldsymbol{z}_t, m_t = k, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{y}_t | \mathbf{H}_k \boldsymbol{z}_t + \mathbf{D}_k \boldsymbol{u}_t + \boldsymbol{d}_k, \mathbf{R}_k) \tag{29.120}$$

See Figure 29.25a for the DPGM representation. It is straightforward to make a nonlinear version of this model.

29.9.2 Posterior inference

Unfortunately exact inference in such switching models is intractable, even in the linear Gaussian case. To see why, suppose for simplicity that the latent discrete switching variable m_t is binary, and that only the dynamics matrix $\mathbf{F}$ depend on m_t, not the observation matrix $\mathbf{H}$. Our initial belief state will be a mixture of 2 Gaussians, corresponding to $p(\boldsymbol{z}_1 | \boldsymbol{y}_1, m_1 = 1)$ and $p(\boldsymbol{z}_1 | \boldsymbol{y}_1, m_1 = 2)$. The one-step-ahead predictive density will be a mixture of 4 Gaussians $p(\boldsymbol{z}_2 | \boldsymbol{y}_1, m_1 = 1, m_2 = 1)$, $p(\boldsymbol{z}_2 | \boldsymbol{y}_1, m_1 = 1, m_2 = 2)$, $p(\boldsymbol{z}_2 | \boldsymbol{y}_1, m_1 = 2, m_2 = 1)$, and $p(\boldsymbol{z}_2 | \boldsymbol{y}_1, m_1 = 2, m_2 = 2)$, obtained by passing each of the prior modes through the 2 possible transition models. The belief state at step 2 will also be a mixture of 4 Gaussians, obtained by updating each of the above distributions with



(a) (b)

Figure 29.25: (a) A switching SSM. Squares represent discrete random variables, circles represent continuous random variables. (b) Illustration of how the number of modes in the belief state of a switching SSM grows exponentially over time. We assume there are two binary states.

$\boldsymbol{y}_2$. At step 3, the belief state will be a mixture of 8 Gaussians. And so on. So we see there is an exponential explosion in the number of modes. Each sequence of discrete values corresponds to a different hypothesis (sometimes called a **track**), which can be represented as a tree, as shown in Figure 29.25b.

Various methods for approximate online inference have been proposed for this model, such as the following:

- Prune off low probability trajectories in the discrete tree. This is widely used in multiple hypothesis tracking methods (see Section 29.9.3).
- Use particle filtering (Section 13.2) where we sample discrete trajectories, and apply the Kalman filter to the continuous variables. See Section 13.4.1 for details.
- Use ADF (Section 8.6), where we approximate the exponentially large mixture of Gaussians with a smaller mixture of Gaussians. See Section 8.6.2 for details.
- Use structured variational inference, where we approximate the posterior as a product of chain-structured distributions, one over the discrete variables and one over the continuous variables, with variational "coupling" terms in between (see e.g., [GH98; PJD21; Wan+22]).

29.9.3 Application: Multitarget tracking

The problem of **multi-target tracking** frequently arises in engineering applications (especially in aerospace and defence). This is a very large topic (see e.g., [BSF88; BSL93; Vo+15] for details), but in this section, we show how switching LDS models (or their nonlinear extensions) can be used to tackle the problem.

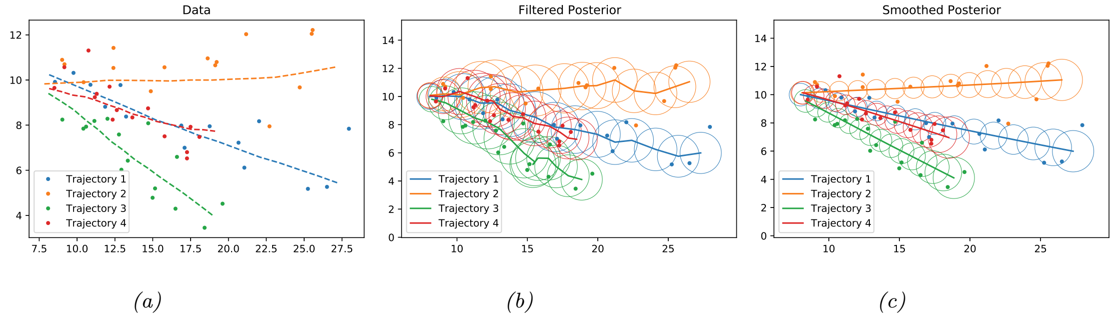


Figure 29.26: Illustration of Kalman filtering and smoothing for tracking multiple moving objects. Generated by kf_parallel.ipynb.

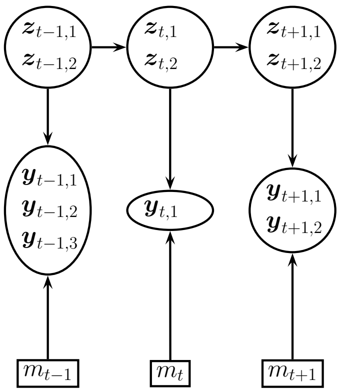


Figure 29.27: A model for tracking two objects in the presence of data-association ambiguity. We observe 3, 1 and 2 detections at time steps $t-1$, t and $t+1$. The m_t hidden variable encodes the association between the observations and the hidden causes.

29.9.3.1 Warm-up

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 29.26. (In this example, each object follows a linear dynamical model with different initial random velocities, as in Section 29.7.1.)

29.9.3.2 Data association

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 $\boldsymbol{x}_t^m$ and the N objects $\boldsymbol{z}_t^i$. This is called the problem of **data association**, and it arises in many application domains.

We can model this problem by augmenting the state space with discrete variables m_t that represent the association matrix between the observations, $\boldsymbol{y}_{t,1:M}$, and the sources, $\boldsymbol{z}_{t,1:N}$. See Figure 29.27 for an illustration, where we have $N = 2$ objects, but a variable number M_t of observations per time step.

As we mentioned in Section 29.9.2, inference in such hybrid (discrete-continouus) models is intractable, due to the exponential number of posterior modes. In the sections below, we briefly mention a few approximate inference methods.

29.9.3.3 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.

29.9.3.4 Other approximate inference schemes

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 9.4). 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) [WL14b; Mey+18].

The cost of each iteration of the iterative procedure is $O(NM)$. Furthermore, [WL14b] 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.

29.9.3.5 Handling an unknown number of targets

In general, we do not know the true number of targets N, so we have to deal with variable-sized state space. This is an example of an **open world** model (see Section 4.6.5), which differs from the standard **closed world assumption** where we know how many objects of interest there are.

For example, suppose at each time step we get two "blips" on our radar screen, representing the presence of an object at a given location. These measurements are not tagged with the source of the object that generated them, so the data looks like Figure 29.28(a). In Figure 29.28(b-c) we show two

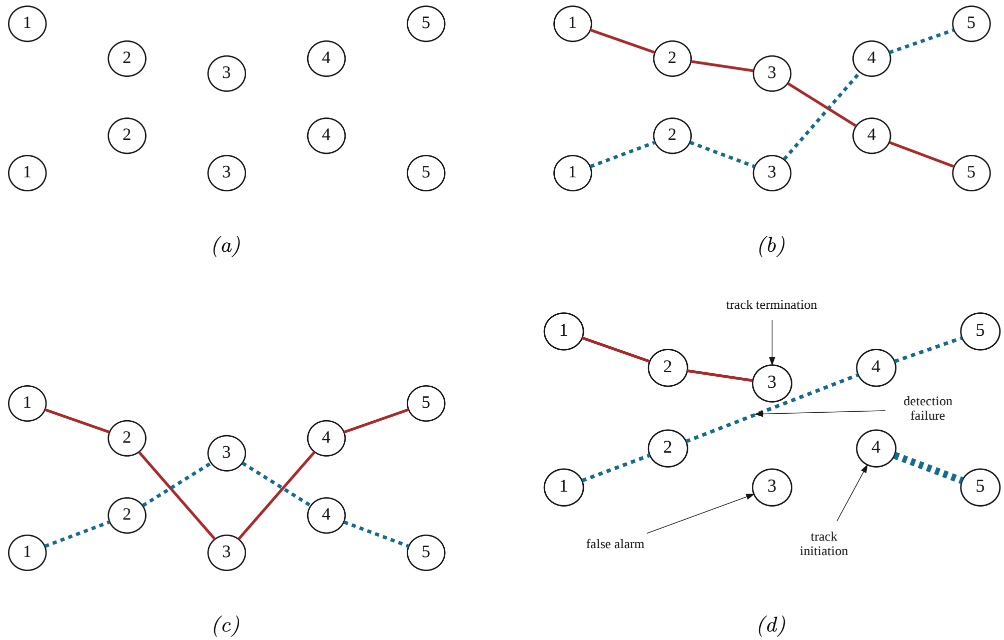


Figure 29.28: Illustration of multi-target tacking in 2d over 5 time steps. (a) We observe 2 measurements per time step. (b-c) Possible hypotheses about the underlying object tracks. (d) A more complex hypothesis in which the red track stops at step 3, the dashed red track starts at step 4, the dotted blue track has a detection failure at step 3, and one of the measurements at step 3 is a false alarm. Adapted from Figure 15.8 of [RN19].

different hypotheses about the underlying object trajectories that could have generated this data. However, how can we know there are two objects? Maybe there are more, but some are just not detected. Maybe there are fewer, and some observations are false alarms due to background clutter. One such more complex hypothesis is shown in Figure 29.28(d). Figuring out what is going on in problems such as this is known as **multiple hypothesis tracking**.

A common approximate solution to this is to create new objects whenever an observation cannot be "explained" (i.e., generated with high likelihood) by any existing objects, and to prune out old objects that have not been detected in a while (in order to keep the computational cost bounded). Sets whose size and content are both random are called **random finite sets**. An elegant mathematical framework for dealing with such objects is described in [Mah07; Mah13; Vo+15].

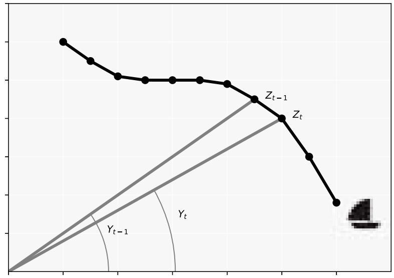


Figure 29.29: Illustration of a bearings-only tracking problem. Adapted from Figure 2.1 of [CP20b].

29.10 Nonlinear SSMs

In this section, we consider SSMs with nonlinear transition and/or observation functions, and additive Gaussian noise. That is, we assume the model has the following form

$$\boldsymbol{z}_t = \boldsymbol{f}(\boldsymbol{z}_{t-1}, \boldsymbol{u}_t) + \boldsymbol{q}_t \tag{29.121}$$
$$\boldsymbol{q}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t) \tag{29.122}$$
$$\boldsymbol{y}_t = \boldsymbol{h}(\boldsymbol{z}_t, \boldsymbol{u}_t) + \boldsymbol{r}_t \tag{29.123}$$
$$\boldsymbol{r}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t) \tag{29.124}$$

This is called a **nonlinear dynamical system** (**NLDS**), or **nonlinear Gaussian SSM** (**NLG-SSM**).

29.10.1 Example: object tracking and state estimation

In Section 8.3.2.3 we give an example of a 2d tracking problem where the motion model is nonlinear, but the observation model is linear.

Here we consider an example where the motion model is linear, but the observation model is nonlinear. In particular, suppose we use the same 2d linear dynamics as in Section 29.7.1, where the state space contains the position and velocity of the object, $\boldsymbol{z}_t = \begin{pmatrix} u_t & v_t & \dot{u}_t & \dot{v}_t \end{pmatrix}$. (We use u and v for the two coordinates, to avoid confusion with the state and observation variables.) Instead of directly observing the location, suppose we have a **bearings only tracking problem**, in which we just observe the angle to the target:

$$y_t = \tan^{-1}\left(\frac{v_t - s_y}{u_t - s_x}\right) + r_t \tag{29.125}$$

where (s_x, s_y) is the position of the measurement sensor. See Figure 29.29 for an illustration. This nonlinear observation model prevents the use of the Kalman filter, but we can still apply approximate inference methods, as we discuss below.

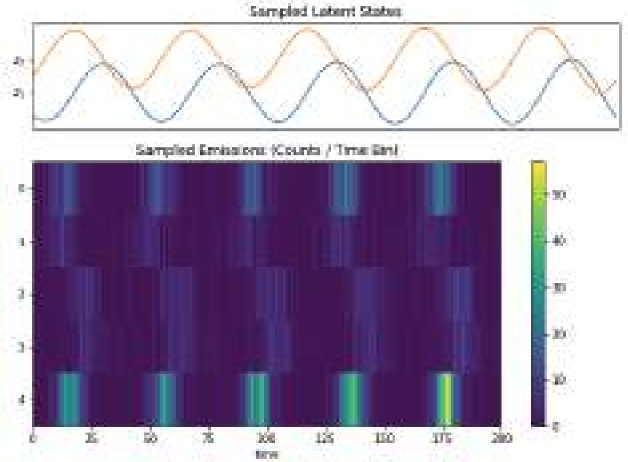


Figure 29.30: Samples from a 2d LDS with 5 Poisson likelihood terms. Generated by poisson_lds.ipynb.

29.10.2 Posterior inference

Inferring the states of an NLDS model is in general computationally difficult. Fortunately, there are a variety of approximate inference schemes that can be used, such as the extended Kalman filter (Section 8.3.2), the unscented Kalman filter (Section 8.4.2), etc.

29.11 Non-Gaussian SSMs

In this section, we consider SSMs in which the transition and observation noise is non-Gaussian. The transition and observation functions can be linear or nonlinear. We can represent this as a probabilistic model as follows:

$$p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{u}_t) = p(\boldsymbol{z}_t|\boldsymbol{f}(\boldsymbol{z}_{t-1}, \boldsymbol{u}_t)) \tag{29.126}$$
$$p(\boldsymbol{y}_t|\boldsymbol{z}_t, \boldsymbol{u}_t) = p(\boldsymbol{y}_t|\boldsymbol{h}(\boldsymbol{z}_t, \boldsymbol{u}_t)) \tag{29.127}$$

This is called a **non-Gaussian SSM** (**NSSM**).

29.11.1 Example: spike train modeling

In this section we discuss consider an SSM with linear-Gaussian latent dynamics and a Poisson likelihood. Such models are widely used in neuroscience for modeling **neural spike trains**. (see e.g., [Pan+10; Mac+11]). This is an example of an **exponential family state-space model** (see e.g., [Vid99; Hel17]).

We consider a simple example where the model has 2 continuous latent variables, and we set the

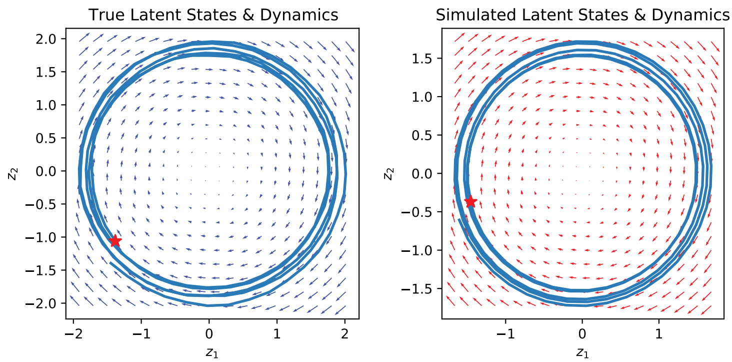


Figure 29.31: Latent state trajectory (blue lines) and dynamics matrix $\mathbf{A}$ *(arrows) for (left) true model and (right) estimated model. The star marks the start of the trajectory. Generated by poisson_lds.ipynb.*

dynamics matrix $\mathbf{A}$ to a random rotation matrix. The observation model has the form

$$p(\boldsymbol{y}_t|\boldsymbol{z}_t) = \prod_{d=1}^{D} \text{Poi}(y_{td}|\exp(\boldsymbol{w}_d^\mathsf{T}\boldsymbol{z}_t)) \tag{29.128}$$

where $\boldsymbol{w}_d$ is a random vector, and we use $D = 5$ observations per time step. Some samples from this model are shown in Figure 29.30.

We can fit this model by using EM, where in the E step we approximate $p(\boldsymbol{y}_t|\boldsymbol{z}_t)$ using a Laplace approximation, after which we can use the Kalman smoother to compute $p(\boldsymbol{z}_{1:T}|\boldsymbol{y}_{1:T})$. In the M step, we optimize the expected complete data log likelihood, similar to Section 29.8.1. We show the result in Figure 29.31, where we compare the parameters $\mathbf{A}$ and the posterior trajectory $\mathbb{E}[\boldsymbol{z}_t|\boldsymbol{y}_{1:T}]$ using the true model and the estimated model. We see good agreement.

29.11.2 Example: stochastic volatility models

In finance, it is common to model 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**, (see e.g., [KSC98]), has the following form:

$$y_t = \boldsymbol{u}_t^\mathsf{T}\boldsymbol{\beta} + \exp(z_t/2)r_t \tag{29.129}$$
$$z_t = \mu + \rho(z_{t-1} - \mu) + \sigma q_t \tag{29.130}$$
$$r_t \sim \mathcal{N}(0, 1) \tag{29.131}$$
$$q_t \sim \mathcal{N}(0, 1) \tag{29.132}$$

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

$$\begin{pmatrix} z_t - \mu \\ z_{t-1} - \mu \end{pmatrix} = \begin{pmatrix} \rho_1 & \rho_2 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} z_{t-1} - \mu \\ z_{t-2} - \mu \end{pmatrix} + \begin{pmatrix} q_t \\ 0 \end{pmatrix} \tag{29.133}$$

where $q_t \sim \mathcal{N}(0, \sigma_z^2)$. Thus we have

$$z_t = \mu + \rho_1(z_{t-1} - \mu) + \rho_2(z_{t-2} - \mu) + q_t \tag{29.134}$$

29.11.3 Posterior inference

Inferring the states of an NGSSM model is in general computationally difficult. Fortunately, there are a variety of approximate inference schemes that can be used, which we discuss in Chapter 8 and Chapter 13.

29.12 Structural time series models

In this section, we discuss **time series forecasting**, which is the problem of computing the predictive distribution over future observations given the data up until the present, i.e., computing $p(\boldsymbol{y}_{t+h}|\boldsymbol{y}_{1:t})$. (The model may optionally be conditioned on known future inputs, to get $p(\boldsymbol{y}_{t+h}|\boldsymbol{y}_{1:t}, \boldsymbol{u}_{1:t+h})$.) There are many approaches to this problem (see e.g., [HA21]), but in this section, we focus on **structural time series** (**STS**) models, which are defined in terms of linear-Gaussian SSMs.

Many classical time series methods, such as the **ARMA** (autoregressive moving average) method, can be represented as STS models (see e.g., [Har90; CK07; DK12; PFW21]). However, the STS approach has much more flexibility. For example, we can create nonlinear, non-Gaussian, and even hierarchical extensions, as we discuss below.

29.12.1 Introduction

The basic idea of an STS model is to represent the observed scalar time series as a sum of C individual **components**:

$$f(t) = f_1(t) + f_2(t) + \cdots + f_C(t) + \epsilon_t \tag{29.135}$$

where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$. For example, we might have a seasonal component that causes the observed values to oscillate up and down, and a growth component, that causes the observed values to get larger over time. Each latent process $f_c(t)$ is modeled by a linear Gaussian state-space model, which (in this context) is also called a **dynamic linear model** (**DLM**). Since these are linear, we can combine them altogether into a single LG-SSM. In particular, in the case of scalar observations, the model has the form

$$p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{z}_t|\mathbf{F}\boldsymbol{z}_{t-1}, \mathbf{Q}) \tag{29.136}$$

$$p(y_t|\boldsymbol{z}_t, \boldsymbol{\theta}) = \mathcal{N}(y_t|\mathbf{H}\boldsymbol{z}_t + \boldsymbol{\beta}^\mathsf{T}\boldsymbol{u}_t, \sigma_y^2) \tag{29.137}$$

where $\mathbf{F}$ and $\mathbf{Q}$ are block structured matrices, with one block per component. The vector $\mathbf{H}$ then adds up all the relevant pieces from each component to generate the overall mean. Note that the matrices $\mathbf{F}$ and $\mathbf{H}$ are fixed sparse matrices which can be derived from the form of the corresponding components of the model, as we discuss below. So the only model parameters are the variance terms, $\mathbf{Q}$ and σ_y^2, and the optional regression coefficients $\boldsymbol{\beta}$.[4] We can generalize this to vector-valued observations as follows:

$$p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{z}_t|\mathbf{F}\boldsymbol{z}_{t-1}, \mathbf{Q}) \tag{29.138}$$

$$p(\boldsymbol{y}_t|\boldsymbol{z}_t, \boldsymbol{\theta}) = \mathcal{N}(y_t|\mathbf{H}\boldsymbol{z}_t + \mathbf{D}\boldsymbol{u}_t, \mathbf{R}) \tag{29.139}$$

29.12.2 Structural building blocks

In this section, we discuss the building blocks of common STS models.

29.12.2.1 Local level model

The simplest latent dynamical process is known as the **local level model**. It assumes the observations $y_t \in \mathbb{R}$ are generated by a Gaussian with (latent) mean μ_t, which evolves over time according to a random walk:

$$y_t = \mu_t + \epsilon_{y,t} \quad \epsilon_{y,t} \sim \mathcal{N}(0, \sigma_y^2) \tag{29.140}$$

$$\mu_t = \mu_{t-1} + \epsilon_{\mu,t}, \quad \epsilon_{\mu,t} \sim \mathcal{N}(0, \sigma_\mu^2) \tag{29.141}$$

We also assume $\mu_1 \sim \mathcal{N}(0, \sigma_\mu^2)$. Hence the latent mean at any future step has distribution $\mu_t \sim \mathcal{N}(0, t\sigma_\mu^2)$, so the variance grows with time. We can also use an autoregressive (AR) process, $\mu_t = \rho\mu_{t-1} + \epsilon_{\mu,t}$, where $|\rho| < 1$. This has the stationary distribution $\mu_\infty \sim \mathcal{N}(0, \frac{\sigma_\mu^2}{1-\rho^2})$, so the uncertainty grows to a finite asymptote instead of unboundedly.

29.12.2.2 Local linear model

Many time series exhibit linear trends upwards or downwards, at least locally. We can model this by letting the level μ_t change by an amount δ_{t-1} (representing the slope of the line over an interval $\Delta t = 1$) at each step

$$\mu_t = \mu_{t-1} + \delta_{t-1} + \epsilon_{\mu,t} \tag{29.142}$$

The slope itself also follows a random walk,

$$\delta_t = \delta_{t-1} + \epsilon_{\delta,t} \tag{29.143}$$

and $\epsilon_{\delta,t} \sim \mathcal{N}(0, \sigma_\delta^2)$. This is called a **local linear trend** model.

We can combine these two processes by defining the following dynamics model:

$$\underbrace{\begin{pmatrix} \mu_t \\ \delta_t \end{pmatrix}}_{\boldsymbol{z}_t} = \underbrace{\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}}_{\mathbf{F}} \underbrace{\begin{pmatrix} \mu_{t-1} \\ \delta_{t-1} \end{pmatrix}}_{\boldsymbol{z}_{t-1}} + \underbrace{\begin{pmatrix} \epsilon_{\mu,t} \\ \epsilon_{\delta,t} \end{pmatrix}}_{\boldsymbol{\epsilon}_t} \tag{29.144}$$

4. In the statistics community, the notation often [DK12], who write the dynamics as $\boldsymbol{\alpha}_t = \mathbf{T}_t\boldsymbol{\alpha}_{t-1} + \boldsymbol{c}_t\mathbf{R}_t\boldsymbol{\eta}_t$ and the observations as $y_t = \mathbf{Z}_t\boldsymbol{\alpha}_t + \boldsymbol{\beta}^\mathsf{T}\boldsymbol{x}_t + H_t\epsilon_t$, where $\boldsymbol{\eta}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\epsilon_t \sim \mathcal{N}(0, 1)$.

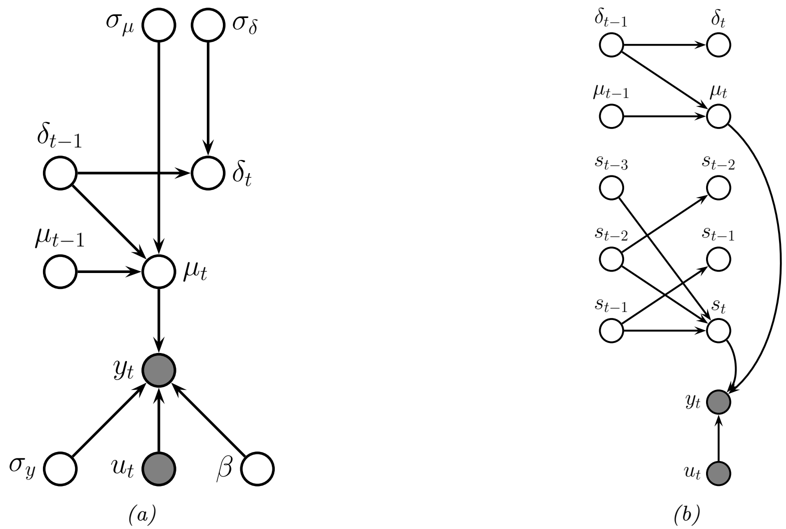


Figure 29.32: (a) A BSTS model with local linear trend and linear regression on inputs. The observed output is y_t. The latent state vector is defined by $\boldsymbol{z}_t = (\mu_t, \delta_t)$. The (static) parameters are $\boldsymbol{\theta} = (\sigma_y, \sigma_\mu, \sigma_\delta, \boldsymbol{\beta})$. The covariates are $\boldsymbol{u}_t$. (b) Adding a latent seasonal process (with $S = 4$ seasons). Parameter nodes are omitted for clarity.

For the emission model we have

$$y_t = \underbrace{\begin{pmatrix} 1 & 0 \end{pmatrix}}_{\mathbf{H}} \underbrace{\begin{pmatrix} \mu_t \\ \delta_t \end{pmatrix}}_{\boldsymbol{z}_t} + \epsilon_{y,t} \tag{29.145}$$

We can also use an autoregressive model for the slope, i.e.,

$$\delta_t = D + \rho(\delta_{t-1} - D) + \epsilon_{\delta,t} \tag{29.146}$$

where D is the long run slope to which δ will revert. This is called a "**semilocal linear trend**" model, and is useful for longer term forecasts.

29.12.2.3 Adding covariates

We can easily include covariates $\boldsymbol{u}_t$ into the model, to increase prediction accuracy. If we use a linear model, we have

$$y_t = \mu_t + \boldsymbol{\beta}^\mathsf{T} \boldsymbol{u}_t + \epsilon_{y,t} \tag{29.147}$$

See Figure 29.32a for an illustration of the local level model with covariates. Note that, when forecasting into the future, we will need some way to predict the input values of future $\boldsymbol{u}_{t+h}$; a simple approach is just to assume future inputs are the same as the present, $\boldsymbol{u}_{t+h} = \boldsymbol{u}_t$.

29.12.2.4 Modelling seasonality

Many time series also exhibit **seasonality**, i.e., they fluctuate periodically. This can be modeled by creating a latent process consisting of a series offset terms, s_t. To model cyclicity, we ensure that these sum to zero (on average) over a complete cycle of S steps:

$$s_t = -\sum_{k=1}^{S-1} s_{t-k} + \epsilon_{s,t}, \ \epsilon_{s,t} \sim \mathcal{N}(0, \sigma_s^2) \tag{29.148}$$

For example, for $S = 4$, we have $s_t = -(s_{t-1} + s_{t-2} + s_{t-3}) + \epsilon_{s,t}$. We can convert this to a first-order model by stacking the last $S - 1$ seasons into the state vector, as shown in Figure 29.32b.

29.12.2.5 Adding it all up

We can combine the various latent processes (local level, linear trend, and seasonal cycles) into a single linear-Gaussian SSM, because the sparse graph structure can be encoded by sparse matrices. More precisely, the transition model becomes

$$\underbrace{\begin{pmatrix} s_t \\ s_{t-1} \\ s_{t-2} \\ \mu_t \\ \delta_t \end{pmatrix}}_{\boldsymbol{z}_t} = \underbrace{\begin{pmatrix} -1 & -1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{F}} \underbrace{\begin{pmatrix} s_{t-1} \\ s_{t-2} \\ s_{t-3} \\ \mu_{t-1} \\ \delta_{t-1} \end{pmatrix}}_{\boldsymbol{z}_{t-1}} + \mathcal{N}\left(\mathbf{0}, \operatorname{diag}([\sigma_s^2, 0, 0, \sigma_\mu^2, \sigma_\delta^2]\right) \tag{29.149}$$

Having defined the model, we can use the Kalman filter to compute $p(\boldsymbol{z}_t | \boldsymbol{y}_{1:t})$, and then make predictions forwards in time by rolling forwards in latent space, and then predicting the outputs:

$$p(\boldsymbol{y}_{t+1} | \boldsymbol{y}_{1:t}) = \int p(\boldsymbol{y}_{t+1} | \boldsymbol{z}_{t+1}) p(\boldsymbol{z}_{t+1} | \boldsymbol{z}_t) p(\boldsymbol{z}_t | \boldsymbol{y}_{1:t}) d\boldsymbol{z}_t \tag{29.150}$$

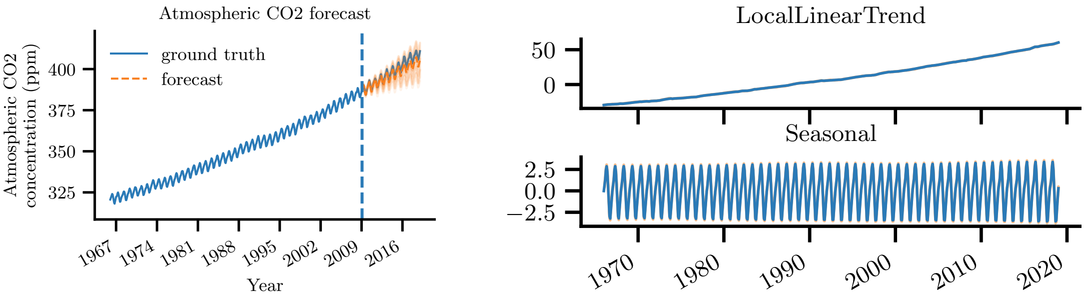
This can be computed in closed form, as explained in Section 8.2.2.

29.12.3 Model fitting

Once we have specified the form of the model, we need to learn the model parameters, $\boldsymbol{\theta} = (\mathbf{D}, \mathbf{R}, \mathbf{Q})$, since $\mathbf{F}$ and $\mathbf{H}$ fixed to the values specified by the structural blocks, and $\mathbf{B} = \mathbf{0}$. Common approaches are based on maximum likelihood estimation (see Section 29.8), and Bayesian inference (see Section 29.8.4). The latter approach is known as **Bayesian structural time series** or **BSTS** modeling [SV14; QRJN18], and often uses the following conjugate prior:

$$p(\boldsymbol{\theta}) = \text{MNIW}(\mathbf{R}, \mathbf{D}) \text{IW}(\mathbf{Q}) \tag{29.151}$$

Alternatively, if there are a large number of covariates, we may use a sparsity-promoting prior (e.g., spike and slab, Section 15.2.5) for the regression coefficients $\mathbf{D}$.



(a) (b)

Figure 29.33: (a) CO_2 levels from Mauna Loa. In orange plot we show predictions for the most recent 10 years. (b) Underlying components for the STS mode which was fit to Figure 29.33a. Generated by sts.ipynb.

29.12.4 Forecasting

Once the parameters have been estimated on an historical dataset, we can perform inference on a new time series to compute $p(\boldsymbol{z}_t|\boldsymbol{y}_{1:t}, \boldsymbol{u}_{1:t}, \boldsymbol{\theta})$ using the Kalman filter (Section 8.2.2). Given the current posterior, we can then "roll forwards" in time to forecast future observations h steps ahead by computing $p(\boldsymbol{y}_{t+h}|\boldsymbol{y}_{1:t}, \boldsymbol{u}_{1:t+h}, \boldsymbol{\theta})$, as in Section 8.2.2.3. If the parameters are uncertain, we can sample from the posterior, $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t}, \boldsymbol{u}_{1:t})$, and then perform Monte Carlo averaging of the forecasts.

29.12.5 Examples

In this section, we give various examples of STS models.

29.12.5.1 Example: forecasting CO_2 levels from Mauna Loa

In this section, we fit an STS model to the monthly atmospheric CO_2 readings from the Mauna Loa observatory in Hawaii.[5] The data is from January 1966 to February 2019. We combine a local linear trend model with a seasonal model, where we assume the periodicity is $S = 12$, since the data is monthly (see Figure 29.33a). We fit the model to all the data except for the last 10 years using variational Bayes. The resulting posterior mean and standard deviations for the parameters are $\sigma_y = 0.169 \pm 0.008$, $\sigma_\mu = 0.159 \pm 0.017$, $\sigma_\delta = 0.009 \pm 0.003$, $\sigma_s = 0.038 \pm 0.008$. We can sample 10 parameter vectors from the posterior and then plug them it to create a distribution over forecasts. The results are shown in orange in Figure 29.33a. Finally, in Figure 29.33b, we plot the posterior mean values of the two latent components (linear trend and current seasonal value) over time. We see how the model has successfully decomposed the observed signal into a sum of two simpler signals. (See also Section 18.8.1 where we model this data using a GP.)

5. For details, see `https://blog.tensorflow.org/2019/03/structural-time-series-modeling-in.html`.

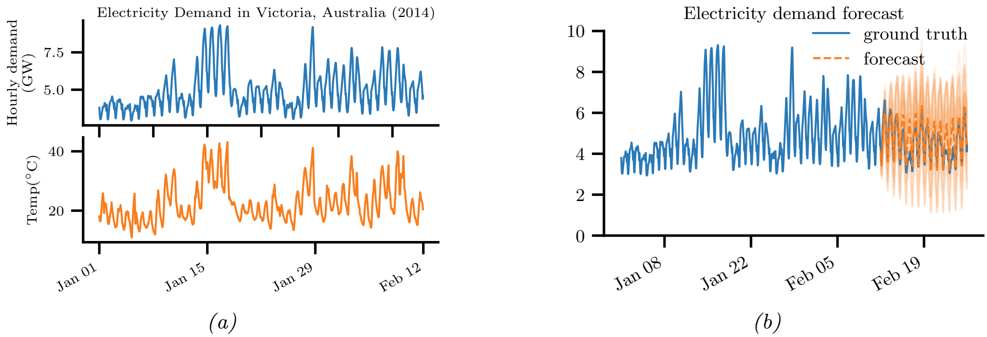


Figure 29.34: (a) Hourly temperature and electricity demand in Victoria, Australia in 2014. (b) Electricity forecasts using an STS. Generated by sts.ipynb.

29.12.5.2 Example: forecasting (real-valued) electricity usage

In this section, we consider a more complex example: forecasting electricity demand in Victoria, Australia, as a function of the previous value and the external temperature. (Remember that January is summer in Australia!) The hourly data from the first six weeks of 2014 is shown in Figure 29.34a.[6]

We fit an STS to this using 4 components: a seasonal hourly effect (period 24), a seasonal daily effect (period 7, with 24 steps per season), a linear regression on the temperature, and an autoregressive term on the observations themselves. We fit the model with variational inference. (This takes about a minute on a GPU.) We then draw 10 posterior samples and show the posterior predictive forecasts in Figure 29.34b. We see that the results are reasonable, but there is also considerable uncertainty.

We plot the individual components in Figure 29.35. Note that they have different vertical scales, reflecting their relative importance. We see that the regression on the external temperature is the most important effect. However, the hour of day effect is also quite significant, even after accounting for external temperature. The autoregressive effect is the most uncertain one, since it is responsible for modeling all of the residual variation in the data beyond what is accounted for by the observation noise.

We can also use the model for **anomaly detection**. To do this, we compute the one-step-ahead predictive distributions, $p(y_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{u}_{1:t})$, for each time step t, and then flag all time steps where the observation is improbable. The results are shown in Figure 29.36.

29.12.5.3 Example: forecasting (integer valued) sales

In Section 29.12.5.2, we used a linear Gaussian STS model to forecast electricity demand. However, for some problems, we have integer valued observations, e.g., for neural spike data (see Section 29.11.1), RNA-Seq data [LJY19], sales data, etc. Here we focus on the case of sales data, where $y_t \in \{0, 1, 2, \ldots\}$ is the number of units of some item that are sold on a given day. Predicting future values of y_t is important for many businesses. (This problem is known as **demand forecasting**.)

We assume the observed counts are due to some latent demand, $z_t \in \mathbb{R}$. Hence we can use a model similar to Section 29.11.1, with a Poisson likelihood, except the linear dynamics are given

6. The data is from `https://github.com/robjhyndman/fpp2-package`.

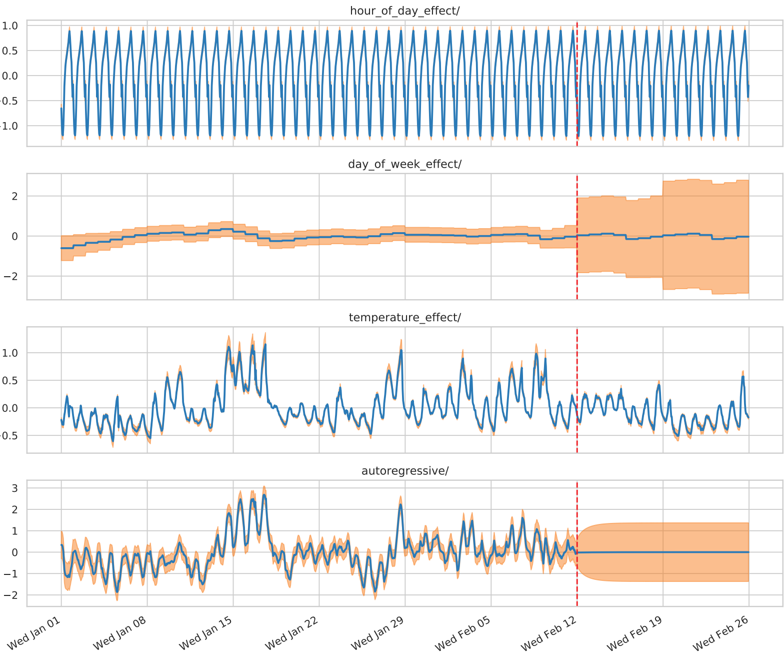


Figure 29.35: Components of the electricity forecasts. Generated by sts.ipynb.

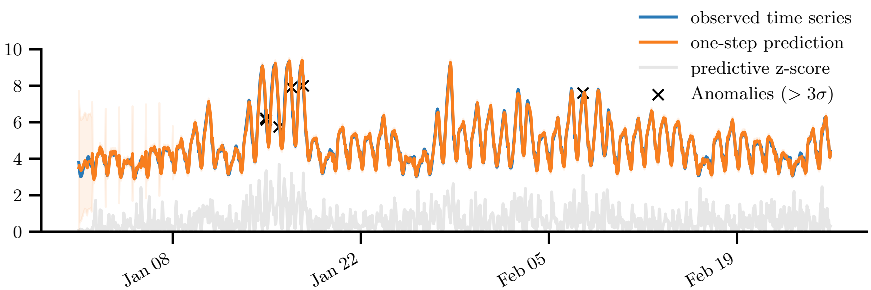


Figure 29.36: Anomaly detection in a time series. We plot the observed electricity data in blue and the predictions in orange. In gray, we plot the z-score at time t , given by $(y_t - \mu_t)/\sigma_t$*, where* $p(y_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{u}_{1:t}) = \mathcal{N}(\mu_t, \sigma_t^2)$*. Anomalous observations are defined as points where* $z_t > 3$ *and are marked with red crosses. Generated by sts.ipynb.*

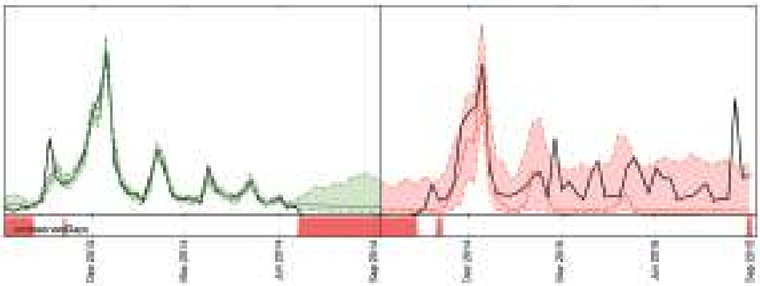


Figure 29.37: Visualization of a probabilistic demand forecast for a hypothetical product. Note the sudden spike near the Christmas holiday in December 2013. The black line denotes the actual demand. Green lines denote the model samples in the training range, while the red lines show the actual probabilistic forecast on data unseen by the model. The red bars at the bottom indicate out-of-stock events which can explain the observed zeros. From Figure 1 of [Bös+17]. Used with kind permission of Tim Januschowski.

by an STS model. In [SSF16; See+17], they consider a likelihood of the form $y_t \sim \text{Poi}(y_t|g(d_t^y))$, where $d_t = z_t + \boldsymbol{u}_t^\mathsf{T}\boldsymbol{w}$ is the instantaneous latent demand, $\boldsymbol{u}_t$ are the covariates that encode seasonal indicators (e.g., temporal distance from holidays), and $g(d) = e^d$ or $\log(1+e^d)$ is the transfer function. The dynamics model is a local random walk term, and $z_t = z_{t-1} + \alpha\mathcal{N}(0,1)$, to capture serial correlation in the data.

However, sometimes we observe zero counts, $y_t = 0$, not because there is no demand, but because there is no supply (i.e., we are out of stock). If we do not model this properly, we may incorrectly infer that $z_t = 0$, thus underestimating demand, which may result in not ordering enough inventory for the future, further compounding the error.

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**, in which $y_t = 0$ is emitted with probability $\sigma(d_t^0)$; otherwise $y_t = 1$ is emitted with probability $\sigma(d_t^1)$; otherwise $y_t =$ is emitted with probability $\text{Poi}(d_t^2)$. This generalizes the scheme in [SOB12].

29.12.5.4 Example: hierarchical SSM for 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 $N_t^j - y_t^j$ vote Republican.) It is natural to want to model the dependencies in this data both across time (longitudinally) and across space (this is an example of **panel data**).

We can do this using a hierarchical SSM, as illustrated in Figure 29.38. The top level Markov chain, z_t^0, models national-level trends, and the state-specific chains, z_t^j, model local "random effects". In practice we would usually also include covariates at the national level, $\boldsymbol{u}_t^0$ and state level, $\boldsymbol{u}_t^j$.

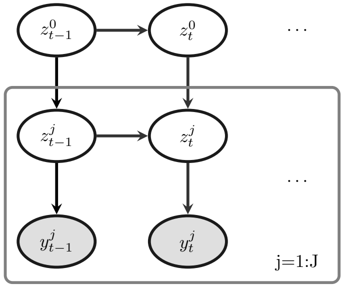


Figure 29.38: Illustration of a hierarchical state-space model.

Thus the model becomes

$$y^j_t \sim \text{Bin}(y^j_t|\pi^j_t, N^j_t) \tag{29.152}$$

$$\pi^j_t = \sigma\left[(\boldsymbol{z}^0_t)^{\mathsf{T}}\boldsymbol{u}^0_t + (\boldsymbol{z}^j_t)^{\mathsf{T}}\boldsymbol{u}^j_t\right] \tag{29.153}$$

$$\boldsymbol{z}^0_t = \boldsymbol{z}^0_{t-1} + \mathcal{N}(\boldsymbol{0}, \sigma^2\mathbf{I}) \tag{29.154}$$

$$\boldsymbol{z}^j_t = \boldsymbol{z}^j_{t-1} + \mathcal{N}(\boldsymbol{0}, \tau^2\mathbf{I}) \tag{29.155}$$

For more details, see [Lin13b].

29.12.6 Causal impact of a time series intervention

In this section, we discuss how to estimate the causal effect on an intervention given some **quasi-experimental** time series data. (The term "quasi-experimental" means the data was collected under an intervention but without using random assignment.) For example, suppose y_t is the click through rate (CTR) of the web page of some company at time t. The company launches an ad campaign at time n, and observes outcomes $\boldsymbol{y}_{1:n}$ before the intervention and $\boldsymbol{y}_{n+1:m}$ after the intervention. (This is an example of an **interrupted time series**, since the "natural" process was perturbed at some point.) A natural question to ask is: what would the CTR have been had the company not run the ad campaign? This is a **counterfactual** question. (We discuss counterfactuals in Section 4.7.4.) If we can predict this counterfactual time series, $\tilde{\boldsymbol{y}}_{n+1:m}$, then we compare the actual y_t to the predicted $\tilde{y}_t$, and use this to estimate the **causal impact** of the intervention.

To predict the counterfactual outcome, we will use a structural time series (STS) model (see Section 29.12), following [Bro+15]. An STS model is a linear-Gaussian state-space model, where arrows have a natural causal interpretation in terms of the **arrow of time**; thus a STS is a kind of structural equation model, and hence a structural causal model (see Section 4.7). The use of an SCM allows us to infer the latent state of the noise variables given the observed data; we can then "roll back time" to the point of intervention, where we explore an alternative "fork in the road" from the one we actually took by "rolling forwards in time" in a new version of the model, using the **twin network** approach to counterfactual inference (see Section 4.7.4).

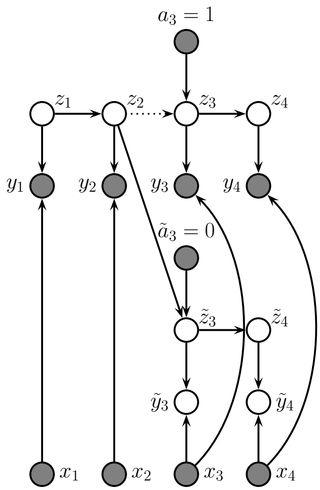


Figure 29.39: Twin network state-space model for estimating causal impact of an intervention that occurs just after time step $n = 2$. We have $m = 4$ actual observations, denoted $\boldsymbol{y}_{1:4}$. We cut the incoming arcs to $\boldsymbol{z}_3$ since we assume $\boldsymbol{z}_{3:T}$ comes from a different distribution, namely the post-intervention distribution. However, in the counterfactual world, shown at the bottom of the figure (with tilde symbols), we assume the distributions are the same as in the past, so information flows along the chain uninterrupted.

29.12.6.1 Computing the counterfactual prediction

To explain the method in more detail, consider the twin network in Figure 29.39. The intervention occurs after time $n = 2$, and there are $m = 4$ observations in total. We observe 2 datapoints before the intervention, $\boldsymbol{y}_{1:2}$, and 2 datapoints afterwards, $\boldsymbol{y}_{3:4}$. We assume observations are generated by latent states $\boldsymbol{z}_{1:4}$, which evolve over time. The states are subject to exogeneous noise terms, which can represent any set of unmodeled factors, such as the state of the economy. In addition, we have exogeneous covariates, $\boldsymbol{x}_{1:m}$.

To predict what would have happened if we had not performed the intervention, (an event denoted by $\tilde{a} = 0$), we replicate the part of the model that occurs after the intervention, and use it to make forecasts. The goal is to compute the counterfactual distribution, $p(\tilde{\boldsymbol{y}}_{n+1:m}|\boldsymbol{y}_{1:n}, \boldsymbol{x}_{1:m})$, where $\tilde{y}_t$ represents counterfactual outcomes if the action had been $\tilde{a} = 0$. We can compute this counterfactual distribution as follows:

$$p(\tilde{\boldsymbol{y}}_{n+1:m}|\boldsymbol{y}_{1:n}, \boldsymbol{x}_{1:m}) = \int p(\tilde{\boldsymbol{y}}_{n+1:m}|\tilde{\boldsymbol{z}}_{n+1:m}, \boldsymbol{x}_{n+1:m}, \boldsymbol{\theta}) p(\tilde{\boldsymbol{z}}_{n+1:m}|\boldsymbol{z}_n, \boldsymbol{\theta}) \times \tag{29.156}$$

$$p(\boldsymbol{z}_n, \boldsymbol{\theta}|\boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n}) d\boldsymbol{\theta} d\boldsymbol{z}_n d\tilde{\boldsymbol{z}}_{n+1:m} \tag{29.157}$$

where

$$p(\boldsymbol{z}_n, \boldsymbol{\theta}|\boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n}) = p(\boldsymbol{z}_n|\boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n}) \tag{29.158}$$

For linear Gaussian SSMs, the term $p(\boldsymbol{z}_n|\boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n}, \boldsymbol{\theta})$ can be computed using Kalman filtering (Section 8.2.2), and the term $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n}, \boldsymbol{x}_{1:n})$, can be computed using MCMC or variational inference.

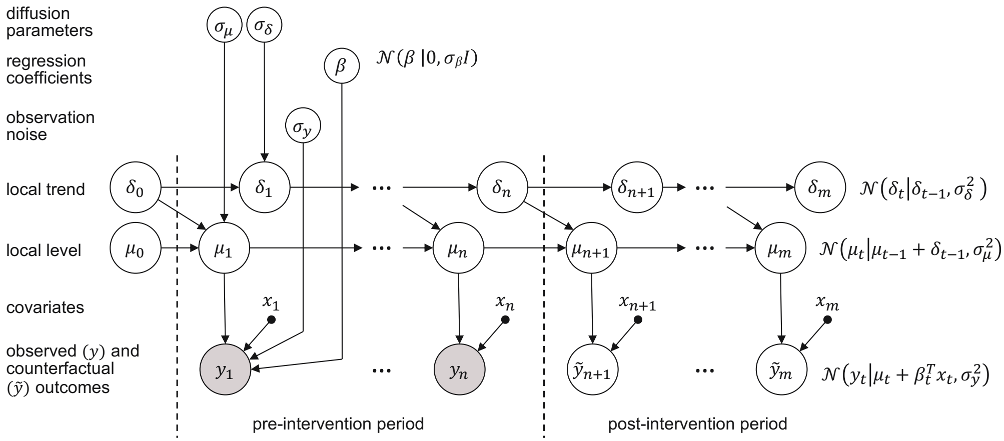


Figure 29.40: A graphical model representation of the local level causal impact model. The dotted line represents the time n at which an intervention occurs. Adapted from Figure 2 of [Bro+15]. Used with kind permission of Kay Brodersen.

We can use samples from the above posterior predictive distribution to compute a Monte Carlo approximation to the distribution of the **treatment effect** per time step, $\tau_t^i = y_t - \tilde{y}_t^i$, where the i index refers to posterior samples. We can also approximate the distribution of the cumulative causal impact using $\sigma_t^i = \sum_{s=n+1}^t \tau_t^i$. (There will be uncertainty in these quantities arising both from epistemic uncertainty, about the true parameters controlling the model, and aleatoric uncertainty, due to system and observation noise.)

29.12.6.2 Assumptions needed for the method to work

The validity of the method is based on 3 assumptions: (1) Predictability: we assume that the outcome can be adequately predicted by our model given the data at hand. (We can check this by using **backcasting**, in which we make predictions on part of the historical data.) (2) Unaffectedness: we assume that the intervention does not change future covariates $\boldsymbol{x}_{n+1:m}$. (We can potentially check this by running the method with each of the covariates as an outcome variable.) (3) Stability: we assume that, had the intervention not taken place, the model for the outcome in the pre-treatment period would have continued in the post-treatment period. (We can check this by seeing if we predict an effect if the treatment is shifted earlier in time.)

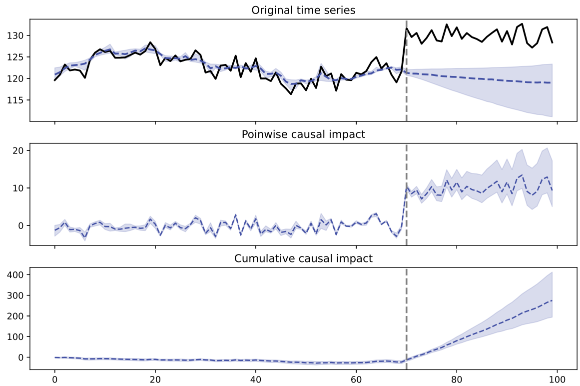


Figure 29.41: Some simulated time series data which we use to estimate the causal impact of some intervention, which occurs at time $n = 70$, Generated by causal_impact_jax.ipynb.

29.12.6.3 Example

As a concrete example, let us assume we have a local level model and we use linear regression to model the dependence on the covariates, as in Section 29.12.2.3. That is,

$$y_t = \mu_t + \boldsymbol{\beta}^\mathsf{T} \boldsymbol{x}_t + \mathcal{N}(0, \sigma_y^2) \tag{29.159}$$

$$\mu_t = \mu_{t-1} + \delta_{t-1} + \mathcal{N}(0, \sigma_\mu^2) \tag{29.160}$$

$$\delta_t = \delta_{t-1} + \mathcal{N}(0, \sigma_\delta^2) \tag{29.161}$$

See the graphical model in Figure 29.40. The static parameters of the model are $\boldsymbol{\theta} = (\boldsymbol{\beta}, \sigma_y^2, \sigma_\mu^2, \sigma_\delta^2)$, the other terms are state or observation variables. (Note that we are free to use any kind of STS model; the local level model is just a simple default.)

For simplicity, let us assume we have a single scalar input x_t, in addition to the scalar output y_t. We create some synthetic data using an autoregressive process on x_t, and then set $y_t = 1.2x_t + \epsilon_t$. We then manually intervene at timestep $t = 70$ by increasing the y_t values by 10. In Figure 29.41, we show the output of the causal impact procedure when applied to this dataset. In the top row, we see that the forecasted output $\tilde{y}_t$ (blue line) at times $t \geq 70$ follows the general AR trend learned by the model on the pre-interventional period, whereas the actual observations y_t (black line) are quite different. Thus the posterior over the pointwise causal impact, $\tau_t = y_t - \tilde{y}_t$, has a value of about 10 for $t \geq 70$.

29.12.6.4 Comparison to synthetic control

The use of a linear combination of other "**donor**" time series $\boldsymbol{\beta}^\mathsf{T} \boldsymbol{x}_t$ is similar in spirit to the concept of a "**synthetic control**" [Aba; Shi+21]. However we do not restrict ourselves to a convex combination of donors. Indeed, when we have many covariates, we can use a spike-and-slab prior (Section 15.2.5) or horseshoe prior (Section 15.2.7) to select the relevant ones. Furthermore, the STS method can be applied even if we just observe the outcome series y_t, without any other parallel time series.

29.12.7 Prophet

Prophet [TL18a] is a popular time series forecasting library from Facebook. It fits a generalized additive model of the form

$$y(t) = g(t) + s(t) + h(t) + \boldsymbol{w}^\mathsf{T}\boldsymbol{x}(t) + \epsilon_t \tag{29.162}$$

where $g(t)$ is a trend function, $s(t)$ is a seasonal fluctuation (modeled using linear regression applied to a sinusoidal basis set), $h(t)$ is an optional set of sparse "holiday effects", $\boldsymbol{x}(t)$ are an optional set of (possibly lagged) covariates, $\boldsymbol{w}$ are the regression coefficients, and $\epsilon(t)$ is the residual noise term, assumed to be iid Gaussian.

Prophet is a regression model, not an auto-regressive model, since it predicts the time series $\boldsymbol{y}_{1:T}$ given the time stamp t and the covariates $\boldsymbol{x}_{1:T}$, but without conditioning on past observations of y. To model the dependence on time, the trend function is assumed to be a piecewise linear trend with S changepoints, uniformly spaced in time. (See Section 29.5.6 for a discussion of changepoint detection.) That is, the model has the form

$$g(t) = (k + \boldsymbol{a}(t)^T\boldsymbol{\delta})t + (m + \boldsymbol{a}(t)^\mathsf{T}\boldsymbol{\gamma}) \tag{29.163}$$

where k is the growth rate, m is the offset, $a_j(t) = \mathbb{I}(t \geq s_j)$, where s_j is the time of the j'th changepoint, $\delta_t \sim \text{Laplace}(\tau)$ is the magnitude of the change, and $\gamma_j = -s_j\delta_j$ to make the function continuous. The Laplace prior on δ ensures the MAP parameter estimate is sparse, so the difference across change point boundaries is usually 0.

For an interactive visualization of how Prophet works, see `https://github.com/MBrouns/timeseers`

29.12.8 Neural forecasting methods

Classical time series methods work well when there is little data (e.g., short sequences, or few covariates). However, in some cases, we have a lot of data. For example, we might have a single, but very long sequence, such as in anomaly detection from real-time sensors [Ahm+17]. Or we may have multiple, related sequences, such as sales of related products [Sal+19b]. In both cases, larger data means we can afford to fit more complex parametric models. Neural networks are a natural choice, because of their flexibility. Until recently, their performance in forecasting tasks was not competitive with classical methods, but this has recently started to change, as described in [Ben+22; LZ20].

A common benchmark in the univariate time series forecasting literature is the **M4 forecasting competition** [MSA18], which requires participants to make forecasts on many different kinds of (univariate) time series (without covariates). This was recently won by a neural method [Smy20]. More precisely, the winner of the 2019 M4 competition was a *hybrid* RNN-classical method called **ES-RNN** [Smy20]. The exponential smoothing (ES) part allows data-efficient adaptation to the observed past of the current time series; the recurrent neural network (RNN) part allows for learning of nonlinear components from multiple related time series. (This is known as a **local+global** model, since the ES part is "trained" just on the local time series, whereas the RNN is trained on a global dataset of related time series.)

In [Ran+18] they adopt a different approach for combining RNNs and classical methods, called **DeepSSM**. In particular, they train a single RNN to predict the parameters of a state-space model (see Main Section 29.1). In more detail, let $\boldsymbol{x}^n_{1:T}$ represent the n'th time series, and let $\boldsymbol{\theta}^n_t$ represent the non-stationary parameters of a linear-trend SSM model (see Section 29.12.1). We train an RNN to compute $\boldsymbol{\theta}_t^n = f(\boldsymbol{c}_{1:T}^n; \boldsymbol{\phi})$, where $\boldsymbol{\phi}$ are the RNN parameters shared across all sequences. We can use the predicted parameters to compute the log likelihood of the sequence, $L_n = \log p(\boldsymbol{x}_{1:T}^n | \boldsymbol{c}_{1:T}^n, \boldsymbol{\theta}_{1:T}^n)$, using the Kalman filter. These two modules can be combined to allow for end-to-end training of $\boldsymbol{\phi}$ to maximize $\sum_{n=1}^N L_n$.

In [Wan+19c], they propose a different hybrid model known as **Deep Factors**. The idea is to represent each time series (or its latent function, for non-Gaussian data) as a weighted sum of a global time series, coming from a neural model, and a stochastic local model, such as an SSM or GP. The **DeepGLO** (global-local) approach of [SYD19] proposes a related hybrid method, where the global model uses matrix factorization to learn shared factors. This is then combined with temporal convolutional networks.

It is also possible to train a purely neural model, without resorting to classical methods. For example, the **N-BEATS** model of [Ore+20] trains a residual network to predict the weights of a set of basis functions, corresponding to a polynomial trend and a periodic signal. The weights for the basis functions are predicted for each window of input using the neural network. Another approach is the **DeepAR** model of [Sal+19b], which fits a single RNN to a large number of time series. The original paper used integer (count) time series, modeled with a negative binomial likelihood function. This is a unimodal distribution, which may not be suitable for all tasks. More flexible forms, such as mixtures of Gaussians, have also been proposed [Muk+18].

A popular alternative is to use **quantile regression** [Koe05], in which the model is trained to predict quantiles of the distribution, which can be done by optimizing the pinball loss (see Section 14.3.2.1). For example, [Gas+19] proposed **SQF-RNN**, which uses splines to represent the quantile function. They used **CRPS** or **continuous-ranked probability score** as the loss function, which trains the model to predict all the quantiles. In particular, for a quantile predictor $F^{-1}(\alpha)$, the CRPS loss is defined as

$$\text{CRPS}(F^{-1}, y) = \int_0^1 2\ell_\alpha(y, F^{-1}(\alpha))d\alpha$$

where the inner loss function is the pinball loss defined in Equation (14.53). CRPS is a proper scoring rule, but is less sensitive to outliers, and is more "distance aware", than log loss. For determiistic predictions, the CRPS reduces to the absolute error.

The above methods all predict a single output (per time step). If there are multiple simultaneous observations, it is best to try to model their interdependencies. In [Sal+19a], they use a (low-rank) **Gaussian copula** for this, and in [Tou+19], they use a **nonparametric copula**.

In [Wen+17], they simultaneously predict quantiles for multiple steps ahead using dilated causal convolution (or an RNN). They call their method **MQ-CNN**. In [WT19], they extend this to predict the full quantile function, taking as input the desired quantile level α, rather than prespecifying a fixed set of levels. They also use a copula to learn the dependencies among multiple univariate marginals.

29.13 Deep SSMs

Traditional state-space model 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 model**, also known as **dynamical variational autoencoders**. (To be consistent with the literature on VAEs, we denote the observations by $\boldsymbol{x}_t$ instead of $\boldsymbol{y}_t$.) For a detailed review, see [Ged+20; Gir+21].

29.13.1 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 29.13.4.)

We can fit a DMM using SVI (Section 10.1.4). The key is to infer the posterior over the latents. From the first-order Markov properties, the exact posterior is given by

$$p(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \cancel{\boldsymbol{x}_{1:t-1}}, \boldsymbol{x}_{t:T}) \tag{29.164}$$

where we define $p(\boldsymbol{z}_1|\boldsymbol{z}_0, \boldsymbol{x}_{1:T}) = p(\boldsymbol{z}_1|\boldsymbol{x}_{1:T})$, and the cancelation follows since $\boldsymbol{z}_t \perp \boldsymbol{x}_{1:t-1}|\boldsymbol{z}_{t-1}$, as pointed out in [KSS17].

In general, it is intractable to compute $p(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T})$, so we approximate it with an inference network. There are many choices for q. A simple one is a fully factorized model, $q(\boldsymbol{z}_{1:T}) = \prod_t q(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})$. This is illustrated in Figure 29.42a. Since $\boldsymbol{z}_t$ only depends on past data, $\boldsymbol{x}_{1:t}$ (which is accumulated in the RNN hidden state $\boldsymbol{h}_t$), 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

$$q(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \cancel{\boldsymbol{x}_{1:t-1}}, \boldsymbol{x}_{t:T}) \tag{29.165}$$

Note that the dependence on past observation $\boldsymbol{x}_{1:t-1}$ is already captured by $\boldsymbol{z}_{t-1}$, as in Equation (29.164). The dependencies on future observations, $\boldsymbol{x}_{t:T}$, can be summarized by a backwards RNN, as shown in Figure 29.42b. Thus

$$q(\boldsymbol{z}_{1:T}, \boldsymbol{h}_{1:T}|\boldsymbol{x}_{1:T}) = \prod_{t=T}^{1} \mathbb{I}\left(\boldsymbol{h}_t = f(\boldsymbol{h}_{t+1}, \boldsymbol{x}_t)\right) \prod_{t=1}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{h}_t) \tag{29.166}$$

(a) (b)

Figure 29.42: Inference networks for deep Markov model. (a) Fully factorized causal posterior $q(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) = \prod_t q(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})$*. The past observations* $\boldsymbol{x}_{1:t}$ *are stored in the RNN hidden state* $_t$*. (b) Markovian posterior* $q(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) = \prod_t q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}_{t:T})$*. The future observations* $\boldsymbol{x}_{t:T}$ *are stored in the RNN hidden state* $_t$*.*

Given a fully factored $q(\boldsymbol{z}_{1:T})$, we can compute the ELBO as follows.

$$\log p(\boldsymbol{x}_{1:T}) = \log \left[\sum_{\boldsymbol{z}_{1:T}} p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T}) p(\boldsymbol{z}_{1:T}) \right] \tag{29.167}$$

$$= \log \mathbb{E}_{q(\boldsymbol{z}_{1:T})} \left[p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T}) \frac{p(\boldsymbol{z}_{1:T})}{q(\boldsymbol{z}_{1:T})} \right] \tag{29.168}$$

$$= \log \mathbb{E}_{q(\boldsymbol{z}_{1:T})} \left[\prod_{t=1}^{T} \frac{p(\boldsymbol{x}_t|\boldsymbol{z}_t) p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})}{q(\boldsymbol{z}_t)} \right] \tag{29.169}$$

$$\geq \mathbb{E}_{q(\boldsymbol{z}_{1:T})} \left[\sum_{t=1}^{T} \log p(\boldsymbol{x}_t|\boldsymbol{z}_t) + \log p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) - \log q(\boldsymbol{z}_t) \right] \tag{29.170}$$

$$= \sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{z}_t)} \left[\log p(\boldsymbol{x}_t|\boldsymbol{z}_t) \right] - \mathbb{E}_{q(\boldsymbol{z}_{t-1})} \left[D_{\mathbb{KL}} \left(q(\boldsymbol{z}_t) \parallel p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) \right) \right] \tag{29.171}$$

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.

29.13.2 Recurrent SSM

In a DMM, the observation model $p(\boldsymbol{x}_t|\boldsymbol{z}_t)$ is first-order Markov, as is the dynamics model $p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$. 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 $\boldsymbol{z}_{1:t}$ instead of just $\boldsymbol{z}_t$ by using $p(\boldsymbol{x}_t|\boldsymbol{h}_t)$, where $\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{z}_t)$, so $\boldsymbol{h}_t$ records all the stochastic choices. This is illustrated in Figure 29.43a. We can also make the dynamical prior depend on $\boldsymbol{z}_{1:t-1}$ by replacing $p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$ with $p(\boldsymbol{z}_t|\boldsymbol{h}_{t-1})$. as is illustrated in Figure 29.43b. 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 $q(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{x}_{1:t})$.

Figure 29.43: Recurrent state-space models. (a) Prior is first-order Markov, $p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$, but observation model is not Markovian, $p(\boldsymbol{x}_t|\boldsymbol{h}_t) = p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t})$, where $\boldsymbol{h}_t$ summarizes $\boldsymbol{z}_{1:t}$. (b) Prior model is no longer first-order Markov either, $p(\boldsymbol{z}_t|\boldsymbol{h}_{t-1}) = p(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1})$. Diamonds are deterministic nodes, circles are stochastic.

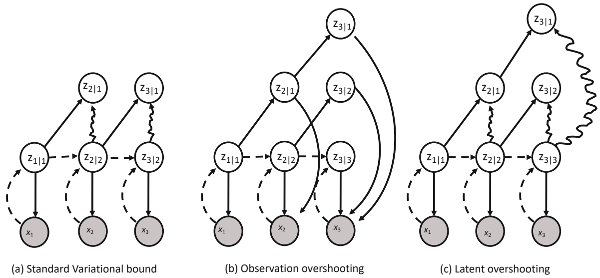


Figure 29.44: Unrolling schemes for SSMs. The labels $z_{i|j}$ is shorthand for $p(\boldsymbol{z}_i|\boldsymbol{x}_{1:j})$. Solid lines denote the generative process, dashed lines the inference process. Arrows pointing at shaded circles represent log-likelihood loss terms. Wavy arrows indicate KL divergence 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 need to care about future observations. Adapted from Figure 3 of [Haf+19].

29.13.3 Improving multistep predictions

In Figure 29.44(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 $\mathbb{E}_{q(\boldsymbol{z}_{t-1})}\left[D_{\mathbb{KL}}\left(q(\boldsymbol{z}_t) \parallel p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})\right)\right]$, and the solid edge $z_{t|t} \rightarrow \boldsymbol{x}_t$ corresponds to $\mathbb{E}_{q(\boldsymbol{z}_t)}\left[\log p(\boldsymbol{x}_t|\boldsymbol{z}_t)\right]$. We see that the dynamics model, $p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$, is only ever penalized in terms of how it differs from the one-step-ahead posterior $q(\boldsymbol{z}_t)$, which can hurt the ability of the model to make long-term predictions.

One solution to this is to make multistep forwards 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 29.44(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 $p(\boldsymbol{z}_t|\boldsymbol{z}_{t-d})$. We can then compute the ELBO

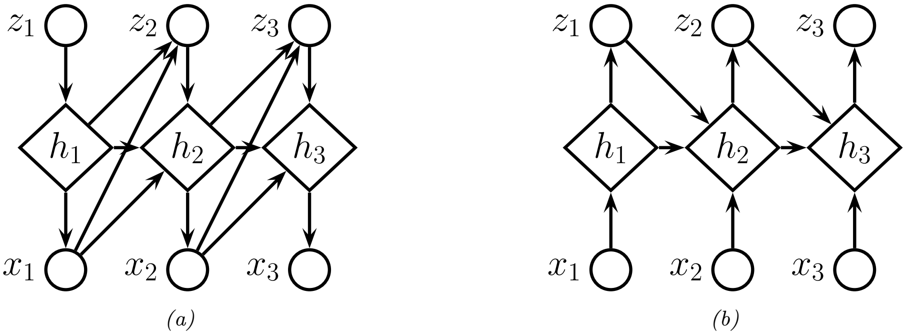


Figure 29.45: Variational RNN. (a) Generative model. (b) Inference model. The diamond-shaped nodes are deterministic.

for this as follows:

$$\log p_d(\boldsymbol{x}_{1:T}) \triangleq \log \int \prod_{t=1}^{T} p(\boldsymbol{z}_t|\boldsymbol{z}_{t-d})p(\boldsymbol{x}_t|\boldsymbol{z}_t)d\boldsymbol{z}_{1:T} \tag{29.172}$$

$$\geq \sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{z}_t)}\left[\log p(\boldsymbol{x}_t|\boldsymbol{z}_t)\right] - \mathbb{E}_{p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_{t-d})q(\boldsymbol{z}_{t-d})}\left[D_{\mathbb{KL}}\left(q(\boldsymbol{z}_t) \parallel p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})\right)\right] \tag{29.173}$$

To train the model so it is good at predicting at different future horizon depths d, we can average the above over all $1 \leq d \leq D$. However, for computational reasons, we can instead just average the KL terms, using weights β_d. This is called **latent overshooting** [Haf+19], and is illustrated in Figure 29.44(c). The new objective becomes

$$\frac{1}{D}\sum_{d=1}^{D} \log p_d(\boldsymbol{x}_{1:T}) \geq \sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{z}_t)}\left[\log p(\boldsymbol{x}_t|\boldsymbol{z}_t)\right] \tag{29.174}$$

$$- \frac{1}{D}\sum_{d=1}^{D} \beta_d \mathbb{E}_{p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_{t-d})q(\boldsymbol{z}_{t-d})}\left[D_{\mathbb{KL}}\left(q(\boldsymbol{z}_t) \parallel p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})\right)\right] \tag{29.175}$$

29.13.4 Variational RNNs

A **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 $\boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+2}, \ldots$ to "feed into" the hidden states, which controls the dynamics. This makes the model less useful for forecasting and model-based RL (see Section 35.4.5.2).

More precisely, the generative model is as follows:

$$p(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{h}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{z}_t|\boldsymbol{h}_{t-1}, \boldsymbol{x}_{t-1})\mathbb{I}\left(\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{x}_{t-1}, \boldsymbol{z}_t)\right) p(\boldsymbol{x}_t|\boldsymbol{h}_t) \tag{29.176}$$

where $p(\boldsymbol{z}_1|\boldsymbol{h}_0, \boldsymbol{x}_0) = p(\boldsymbol{z}_0)$ and $\boldsymbol{h}_1 = f(\boldsymbol{h}_0, \boldsymbol{x}_0, \boldsymbol{z}_1) = f(\boldsymbol{z}_1)$. Thus $\boldsymbol{h}_t = (\boldsymbol{z}_{1:t}, \boldsymbol{x}_{1:t-1})$ 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 $p(\boldsymbol{z}_t|\boldsymbol{h}_{t-1}, \boldsymbol{x}_{t-1}) = p(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:t-1})$, whereas in a DMM, it is $p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$, and in an RSSM, it is $p(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1})$. See Figure 29.45a for an illustration.

We can train VRNNs using SVI. In [Chu+15], they use the following inference network:

$$q(\boldsymbol{z}_{1:T}, \boldsymbol{h}_{1:T}|\boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} \mathbb{I}\left(\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{z}_{t-1}, \boldsymbol{x}_t\right) q(\boldsymbol{z}_t|\boldsymbol{h}_t) \tag{29.177}$$

Thus $\boldsymbol{h}_t = (\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:t})$. Marginalizing out these deterministic nodes, we see that the filtered posterior has the form $q(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) = \prod_t q(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:t})$. See Figure 29.45b for an illustration. (We can also optionally replace $\boldsymbol{x}_t$ with the output of a bidirectional RNN to get the smoothed posterior, $q(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) = \prod_t q(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:T})$.)

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.

30 Graph learning

30.1 Introduction

Graphs are a very common way to represent data. In this chapter we discuss probability models for graphs. In Section 30.2, 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 30.3, we assume the graph structure G is unknown and needs to be inferred from correlated data, $\boldsymbol{x}_n \in \mathbb{R}^D$; for this, we will use probabilistic graphical models with unknown topology. See also Section 16.3.6, where we discuss graph neural networks, for performing supervised learning using graph-structured data.

30.2 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. To try to find "interesting structure" in such graphs, such as clusters or communities, it is common to fit latent variable generative models of various forms, such as the **stochastic blocks model**.

More details on this topic can be found in Supplementary Section 30.1 in the online supplementary material.

30.3 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 N_G nodes. (Usually $N_G = D$, but we also consider the case where we learn extra latent nodes that are not present in the input.)

30.3.1 Methods

There are many different methods for learning PGM graph structures. See e.g., [VCB22] for a recent review. More details on this topic can be found in Supplementary Chapter 30 in the online supplementary material.

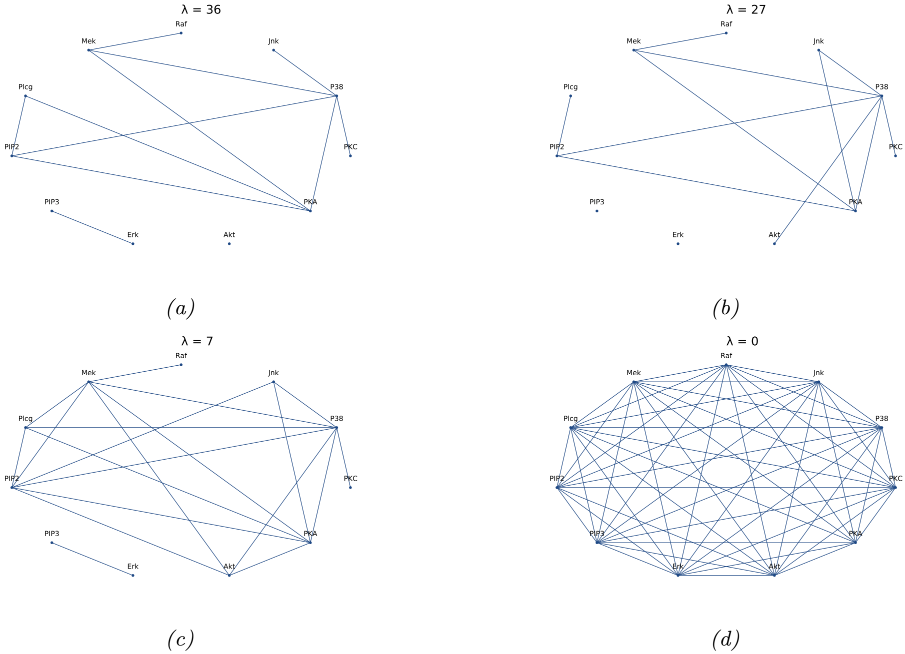


Figure 30.1: A sparse undirected Gaussian graphical model learned using graphical lasso applied to some flow cytometry data (from [Sac+05]), which measures the phosphorylation status of 11 proteins. The sparsity level is controlled by λ. (a) $\lambda = 36$. (b) $\lambda = 27$. (c) $\lambda = 7$. (d) $\lambda = 0$. Adapted from Figure 17.5 of [HTF09]. Generated by ggm_lasso_demo.ipynb.

30.3.2 Applications

In terms of 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 30.1 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 Supplementary Section 30.4.2. 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 this part of the bird brain.

In some cases, we are not interested in interpreting the graph structure, we just want to use it to make predictions. One example of this is in financial portfolio management, where accurate models of the covariance between large numbers of different stocks is important. [CW07] show that by learning a sparse graph, and then using this as the basis of a trading strategy, it is possible to outperform (i.e., make more money than) methods that do not exploit sparse graphs. Another example is predicting traffic jams on the freeway. [Hor+05] describe a deployed system called JamBayes for predicting traffic flow in the Seattle area, using a directed graphical model whose structure was learned from data.

Structure learning is also an important pre-requisite for causal inference. In particular, to predict the effects of interventions on a system, or to perform counterfactual reasoning, we need to know the structural causal model (SCM), as we discuss in Section 4.7. An SCM is a kind of directed graphical model where the relationships between nodes are deterministic (functional), except for stochastic root (exogeneous) variables. Consequently one can use techniques for learning DAG structures as a way to learn SCMs, if we make some assumptions about (lack of) confounders. This is called **causal discovery**. See Supplementary Section 30.5 in the online supplementary material for details.

31 Nonparametric Bayesian models

This chapter is written by Vinayak Rao.

31.1 Introduction

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, neural networks have a fixed number of parameters, independent of the dataset. In a **parametric Bayesian model**, a prior probability distribution on these parameters is used to define a prior distribution on the objects of interest. By contrast, in a **Bayesian nonparametric** (**BNP**) model (also called a **non-parametric Bayesian** model) we directly place prior distributions on the objects of interest, such as functions, graphs, probability distributions, etc. This is usually done via some kind of **stochastic process**, which is a probability distribution over a potentially infinite set of random variables.

One example is a **Gaussian process**. As explained in Chapter 18, this defines a probability distribution over an unknown function $f : \mathcal{X} \to \mathbb{R}$, such that the joint distribution of $f(\mathbf{X}) = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N))$ is jointly Gaussian for any finite set of values $\mathbf{X} = \{\boldsymbol{x}_n \in \mathcal{X}\}_{n=1}^N$ i.e., $p(f(\mathbf{X})) = \mathcal{N}(f(\mathbf{X})|\boldsymbol{\mu}(\mathbf{X}), \mathbf{K}(\mathbf{X}))$ where $\boldsymbol{\mu}(\mathbf{X}) = [\mu(\boldsymbol{x}_1), \ldots, \mu(\boldsymbol{x}_N)]$ is the mean, $\mathbf{K}(\mathbf{X}) = [\mathcal{K}(\boldsymbol{x}_i), \mathcal{K}(\boldsymbol{x}_j)]$ is the $N \times N$ Gram matrix, and $\mathcal{K}$ is a positive definite kernel function. The complexity of the posterior over functions can grow with the amount of data, avoiding underfitting, since we maintain a full posterior distribution over the infinite set of unknown "parameters" (i.e., function evaluations at all points $\boldsymbol{x} \in \mathcal{X}$). But by taking a Bayesian approach, we avoid overfitting this infinitely flexible model. 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. We can also define probability distributions over probability distributions, as well as other kinds of objects.

We discuss these topics in more detail in Supplementary Chapter 31. For even more information, see e.g., [Hjo+10; GV17].

32 Representation learning

This chapter is written by Ben Poole and Simon Kornblith.

32.1 Introduction

Representation learning is a paradigm for training machine learning models to transform raw inputs into a form that makes it easier to solve new tasks. Unlike supervised learning, where the task is known at training time, representation learning often assumes that we do not know what task we wish to solve ahead of time. Without this knowledge, are there transformations of the input we can learn that are useful for a variety of tasks we might care about?

One point of evidence that representation learning is possible comes from us. Humans can rapidly form rich representations of new classes [LST15] that can support diverse behaviors: finding more instances of that class, decomposing that instance into parts, and generating new instances from that class. However, it is hard to directly specify what representations we would like our machine learning systems to learn. We may want it make it easy to solve new tasks with small amounts of data, we may want to construct novel inputs or answer questions about similarities between inputs, and we may want the representation to encode certain information while discarding other information.

In building methods for representation learning, the goal is to design a task whose solution requires learning an improved representation of the input instead of directly specifying what the representation should do. These tasks can vary greatly, from building generative models of the dataset to learning to cluster datapoints. Different methods often involve different assumptions on the dataset, different kinds of data, and induce different biases on the learned representation. In this chapter we first discuss methods for evaluating learned representations, then approaches for learning representations based on supervised learning, generative modeling, and self-supervised learning, and finally the theory behind when representation learning is possible.

32.2 Evaluating and comparing learned representations

How can we make sense of representations learned by different neural networks, or of the differences between representations learned in different layers of the same network? Although it is tempting to imagine representations of neural networks as points in a space, this space is high-dimensional. In order to determine the quality of representations and how different representations differ, we need ways to summarize these high-dimensional representations or their relationships with a few

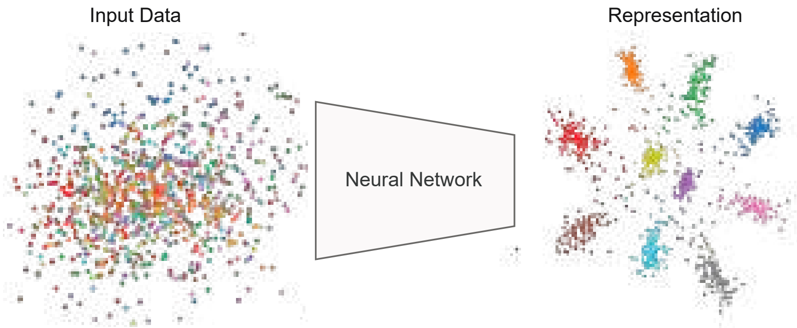


Figure 32.1: Representation learning transforms input data (left) where data from different classes (color) are mixed together to a representation (right) where attributes like class are more easily distinguished. Generated by vib_ demo.ipynb.

relevant scalars. Doing so requires making assumptions about what structure in the representations is important.

32.2.1 Downstream performance

The most common way to evaluate the quality of a representation is to adapt it to one or more downstream tasks thought to be representative of real-world scenarios. In principle, one could choose any strategy to adapt the representation, but a small number of adaptation strategies dominate the literature. We discuss these strategies below.

Clearly, downstream performance can only differ from pretraining performance if the downstream task is different from the pretraining task. Downstream tasks can differ from the pretraining task in their input distributions, target distributions, or both. The downstream tasks used to evaluate unsupervised or self-supervised representation learning often involve the same distribution of inputs as the pretraining task, but require predicting targets that were not provided during pretraining. For example, in self-supervised visual representation learning, representations learned on the ImageNet dataset without using the accompanying labels are evaluated on ImageNet using labels, either by performing linear evaluation with all the data or by fine-tuning using subsets of the data. By contrast, in **transfer learning** (Section 19.5.1), the input distribution of the downstream task differs from the distribution of the pretraining task. For example, we might pretrain the representation on a large variety of natural images and then ask how the representation performs at distinguishing different species of birds not seen during pretraining.

32.2.1.1 Linear classifiers and linear evaluation

Linear evaluation treats the trained neural network as a fixed feature extractor and trains a linear classifier on top of fixed features extracted from a chosen network layer. In earlier work, this linear classifier was often a support vector machine [Don+14; SR+14; Cha+14], but in more recent work, it is typically an L^2-regularized multinomial logistic regression classifier [ZIE17; KSL19; KZB19]. The process of training this classifier is equivalent to attaching a new layer to the chosen representation layer and training only this new layer, with the rest of the network's weights frozen and any normalization/regularization layers set to "inference mode" (see Figure 32.2).

Although linear classifiers are conceptually simple compared to deep neural networks, they are not necessarily simple to train. Unlike deep neural network training, objectives associated with commonly-used linear classifiers are convex and thus it is possible to find global minima, but it can be challenging to do so with stochastic gradient methods. When using SGD, it is important to tune both the learning rate schedule and weight decay. Even with careful tuning, SGD may still require substantially more epochs to converge when training the classifier than when training the original neural network [KZB19]. Nonetheless, linear evaluation with SGD remains a commonly used approach in the representation learning literature.

When it is possible to maintain all features in memory simultaneously, it is possible to use full-batch optimization method with line search such as L-BFGS in place of SGD [KSL19; Rad+21]. These optimization methods ensure that the loss decreases at every iteration of training, and thus do not require manual tuning of learning rates. To obtain maximal accuracy, it is still important to tune the amount of regularization, but this can be done efficiently by sweeping this hyperparameter and using the optimal weights for the previous value of the hyperparameter as a warm start. Using a full-batch optimizer typically implies precomputing the features before performing the optimization, rather than recomputing features on each minibatch. Precomputing features can save a substantial amount of computation, since the forwards passes through the frozen model are typically much more expensive than computing the gradient of the linear classifier. However, precomputing features also limits the number of augmentations of each example that can be considered.

It is important to keep in mind that the accuracy obtainable by training a linear classifier on a finite dataset is only a lower bound on the accuracy of the Bayes-optimal linear classifier. The datasets used for linear evaluation are often small relative to the number of parameters to be trained, and the classifier typically needs to be regularized to obtain maximal accuracy. Thus, linear evaluation accuracy depends not only on whether it is possible to linearly separate different classes in the representation, but also on how much data is required to find a good decision boundary with a given training objective and regularizer. In practice, even an invertible linear transformation of a representation can affect linear evaluation accuracy.

32.2.1.2 Fine-tuning

It is also possible to adapt all layers from the pretraining task to the downstream task. This process is typically referred to as **fine-tuning** [HS06b; Gir+14]. In its simplest form, fine-tuning, like linear evaluation, involves attaching a new layer to a chosen representation layer, but unlike linear evaluation, all network parameters, and not simply those of the new layer, are updated according to gradients computed on the downstream task. The new layer may be initialized with zeros or using the solution obtained by training it with all other parameters frozen. Typically, the best results are obtained when the network is fine-tuned at a lower learning rate than was used for pretraining.

Fine-tuning is substantially more expensive than training a linear classifier on top of fixed feature representations, since each training step requires backpropagating through multiple layers. However, fine-tuned networks typically outperform linear classifiers, especially when the pretraining and downstream tasks are very different [KSL19; AGM14; Cha+14; Azi+15]. Linear classifiers perform better only when the number of training examples is very small (~5 per class) [KSL19].

Fine-tuning can also involve adding several new network layers. For detection and segmentation tasks, which require fine-grained knowledge of spatial position, it is common to add a *feature pyramid network* (FPN) [Lin+17b] that incorporates information from different feature maps in the pretrained network. Alternatively, new layers can be interspersed between old layers and initialized to preserve the network's output. *Net2Net* [CGS15] follows this approach to construct a higher-capacity network that makes use of representations contained in the pretrained weights of a smaller network, whereas *adapter modules* [Hou+19] incorporate new, parameter-efficient modules into a pretrained network and freeze the old ones to reduce the number of parameters that need to be stored when adapting models to different tasks.

32.2.1.3 Disentanglement

Given knowledge about how a dataset was generated, for example that there are certain factors of variation such as position, shape, and color that generated the data, we often wish to estimate how well we can recover those factors in our learned representation. This requires using disentangled representation learning methods (see Section 21.3.1.1). While there are a variety of metrics for disentanglement, most measure to what extent there is a one-to-one correspondence between latent factors and dimensions of the learned representation.

32.2.2 Representational similarity

Rather than measure compatibility between a representation and a downstream task, we might seek to directly examine relationships between two fixed representations without reference to a task. In this section, we assume that we have two sets of fixed representations corresponding to the same n examples. These representations could be extracted from different layers of the same network or layers of different neural networks, and need not have the same dimensionality. For notational convenience, we assume that each set of representations has been stacked row-wise to form matrices $\boldsymbol{X} \in \mathbb{R}^{n \times p_1}$ and $\boldsymbol{Y} \in \mathbb{R}^{n \times p_2}$ such that $\boldsymbol{X}_{i,:}$ and $\boldsymbol{Y}_{i,:}$ are two different representations of the same example.

32.2.2.1 Representational similarity analysis and centered kernel alignment

Representational similarity analysis (RSA) is the dominant technique for measuring similarity of representations in neuroscience [KMB08], but has also been applied in machine learning. RSA reduces the problem of measuring similarity between representation matrices to measuring the similarities between representations of individual examples. RSA begins by forming representational similarity matrices (RSMs) from each representation. Given functions $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ and $k' : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ that measure the similarity between pairs of representations of individual examples $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$, and $\boldsymbol{y}, \boldsymbol{y}' \in \mathcal{Y}$, the corresponding representational similarity matrices $\boldsymbol{K}, \boldsymbol{K}' \in \mathbb{R}^{n \times n}$ contain the similarities between the representations of all pairs of examples $\boldsymbol{K}_{ij} = k(\boldsymbol{X}_{i,:}, \boldsymbol{X}_{j,:})$ and $\boldsymbol{K}'_{ij} = k'(\boldsymbol{Y}_{i,:}, \boldsymbol{Y}_{j,:})$. These representational similarity matrices are transformed into a scalar similarity value by applying a matrix similarity function $s(\boldsymbol{K}, \boldsymbol{K}')$.

The RSA framework can encompass many different forms of similarity through the selection of the similarity functions $k(\cdot, \cdot)$, $k'(\cdot, \cdot)$, and $s(\cdot, \cdot)$. How these functions should be selected is a contentious topic [BS+20; Kri19]. In practice, it is common to choose $k(\boldsymbol{x}, \boldsymbol{x}') = k'(\boldsymbol{x}, \boldsymbol{x}') = \text{corr}[\boldsymbol{x}, \boldsymbol{x}']$, the Pearson correlation coefficient between examples. $s(\cdot, \cdot)$ is often chosen to be the Spearman rank correlation between the representational similarity matrices, which is computed by reshaping $\boldsymbol{K}$ and

$\boldsymbol{K}'$ to vectors, computing the rankings of their elements, and measuring the Pearson correlation between these rankings.

Centered kernel alignment (CKA) is a technique that was first proposed in machine learning literature [Cri+02; CMR12] but that can be interpreted as a form of RSA. In centered kernel alignment, the per-example similarity functions k and k' are chosen to be positive semi-definite kernels so that $\boldsymbol{K}$ and $\boldsymbol{K}'$ are kernel matrices. The matrix similarity function s is the cosine similarity between centered kernel matrices

$$s(\boldsymbol{K}, \boldsymbol{K}') = \frac{\langle \boldsymbol{HKH}, \boldsymbol{HK}'\boldsymbol{H} \rangle_{\mathrm{F}}}{\|\boldsymbol{HKH}\|_{\mathrm{F}} \|\boldsymbol{HK}'\boldsymbol{H}\|_{\mathrm{F}}}, \tag{32.1}$$

where $\langle \boldsymbol{A}, \boldsymbol{B} \rangle_{\mathrm{F}} = \mathrm{vec}(\boldsymbol{A})^\top \mathrm{vec}(\boldsymbol{B}) = \mathrm{tr}(\boldsymbol{A}^\top \boldsymbol{B})$ is the Frobenius product, and $\boldsymbol{H} = \boldsymbol{I} - \frac{1}{n}\boldsymbol{1}\boldsymbol{1}^\top$ is the centering matrix. As it is applied above, the centering matrix subtracts the means from the rows and columns of the similarity index.

A special case of centered kernel alignment arises when k and k' are chosen to be the linear kernel $k(\boldsymbol{x}, \boldsymbol{x}') = k'(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}'$. In this case, $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^\top$ and $\boldsymbol{K}' = \boldsymbol{Y}\boldsymbol{Y}^\top$, allowing for an alternative expression for CKA in terms of the similarities between pairs of *features* rather than pairs of examples. The representations themselves must first be centered by subtracting the means from their columns, yielding $\tilde{\boldsymbol{X}} = \boldsymbol{HX}$ and $\tilde{\boldsymbol{Y}} = \boldsymbol{HY}$. Then, so-called linear centered kernel alignment is given by

$$s(\boldsymbol{K}, \boldsymbol{K}') = \frac{\langle \tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^\top, \tilde{\boldsymbol{Y}}\tilde{\boldsymbol{Y}}^\top \rangle_{\mathrm{F}}}{\|\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^\top\|_{\mathrm{F}} \|\tilde{\boldsymbol{Y}}\tilde{\boldsymbol{Y}}^\top\|_{\mathrm{F}}} = \frac{\|\tilde{\boldsymbol{X}}^\top \tilde{\boldsymbol{Y}}\|_{\mathrm{F}}^2}{\|\tilde{\boldsymbol{X}}^\top \tilde{\boldsymbol{X}}\|_{\mathrm{F}} \|\tilde{\boldsymbol{Y}}^\top \tilde{\boldsymbol{Y}}\|_{\mathrm{F}}}. \tag{32.2}$$

Linear centered kernel alignment is equivalent to the RV coefficient [RE76] between centered features, as shown in [Kor+19].

32.2.2.2 Canonical correlation analysis and related methods

Given two datasets (in this case, the representation matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$), canonical correlation analysis or CCA (Section 28.3.4.3) seeks to map both datasets to a shared latent space such that they are maximally correlated in this space. The i^{th} pair of canonical weights $(\boldsymbol{w}_i, \boldsymbol{w}_i')$ maximize the correlation between the corresponding canonical vectors $\rho_i = \mathrm{corr}[\boldsymbol{X}\boldsymbol{w}_i, \boldsymbol{Y}\boldsymbol{w}_i']$ subject to the constraint that the new canonical vectors are orthogonal to previous canonical vectors,

$$\begin{aligned}
&\text{maximize } \mathrm{corr}[\boldsymbol{X}\boldsymbol{w}_i, \boldsymbol{Y}\boldsymbol{w}_i'] \\
&\text{subject to} \quad \forall_{j<i} \;\; \boldsymbol{X}\boldsymbol{w}_i \perp \boldsymbol{X}\boldsymbol{w}_j \\
&\qquad\qquad\quad\; \forall_{j<i} \;\; \boldsymbol{Y}\boldsymbol{w}_i' \perp \boldsymbol{Y}\boldsymbol{w}_j' \\
&\qquad\qquad\quad\; \|\boldsymbol{X}\boldsymbol{w}_i\| = \|\boldsymbol{Y}\boldsymbol{w}_i'\| = 1.
\end{aligned} \tag{32.3}$$

The maximum number of non-zero canonical correlations is the minimum of the ranks, $p = \min(\mathrm{rk}(\boldsymbol{X}), \mathrm{rk}(\boldsymbol{Y}))$.

The standard algorithm for computing the canonical weights and correlations [BG73] first decomposes the individual representations as the product of an orthogonal matrix and a second matrix, $\tilde{\boldsymbol{X}} = \boldsymbol{QR} : \boldsymbol{Q}^\top \boldsymbol{Q} = \boldsymbol{I}$ and $\tilde{\boldsymbol{Y}} = \boldsymbol{Q}'\boldsymbol{R}' : \boldsymbol{Q}'^\top \boldsymbol{Q}' = \boldsymbol{I}$. These decompositions can be obtained either by QR factorization or singular value decomposition. A second singular value decomposition of $\boldsymbol{Q}^\top \boldsymbol{Q}' = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ provides the quantities needed to determine the canonical weights and correlations.

Specifically, the canonical correlations are the singular values $\boldsymbol{\rho} = \text{diag}(\boldsymbol{\Sigma})$; the canonical vectors are the columns of $\boldsymbol{XW} = \boldsymbol{QU}$ and $\boldsymbol{YW}' = \boldsymbol{Q}'\boldsymbol{V}$; and the canonical weights are $\boldsymbol{W} = \boldsymbol{R}^{-1}\boldsymbol{U}$ and $\boldsymbol{W}' = \boldsymbol{R}'^{-1}\boldsymbol{V}$.

Two common strategies to turn the resulting vector of canonical correlations into a scalar are to take the **mean squared canonical correlation**,

$$R^2_{\text{CCA}}(\boldsymbol{X}, \boldsymbol{Y}) = \|\boldsymbol{\rho}\|_2^2/p = \|\boldsymbol{Q}^\top \boldsymbol{Q}'\|_{\text{F}}^2/p, \tag{32.4}$$

or the **mean canonical correlation**,

$$\bar{\rho} = \sum_{i=1}^{p} \boldsymbol{\rho}_i/p = \|\boldsymbol{Q}^\top \boldsymbol{Q}'\|_*/p, \tag{32.5}$$

where $\|\cdot\|_*$ denotes the nuclear norm. The mean squared canonical correlation has several alternative names, including Yanai's GCD [Yan74; RBS84], Pillai's trace, or the eigenspace overlap score [May+19].

Although CCA is a powerful tool, it suffers from the curse of dimensionality. If at least one representation has more neurons than examples and each neuron is linearly independent of the others, then all canonical correlations are 1. In practice, because neural network representations are high-dimensional, we can find ourselves in the regime where there are not enough data to accurately estimate the canonical correlations. Moreover, even when we can accurately estimate the canonical correlations, it may be desirable for a similarity measure to place less emphasis on the similarity of low-variance directions.

Singular vector CCA (SVCCA) mitigates these problems by retaining only the largest principal components of $\tilde{\boldsymbol{X}}$ and $\tilde{\boldsymbol{Y}}$ when performing CCA. Given the singular value decomposition of the representations $\tilde{\boldsymbol{X}} = \boldsymbol{U\Sigma V}^\top$ and $\tilde{\boldsymbol{Y}} = \boldsymbol{U}'\boldsymbol{\Sigma}'\boldsymbol{V}'^\top$, SVCCA retains only the first k columns of $\boldsymbol{U}$ corresponding to the largest k singular values of $\boldsymbol{\sigma} = \text{diag}(\boldsymbol{\Sigma})$ (i.e., the k largest principal components) and the first k' columns of $\boldsymbol{U}'$ corresponding to the largest k' singular values $\boldsymbol{\sigma}' = \text{diag}(\boldsymbol{\Sigma}')$. With these singular value decompositions, the canonical correlations, vectors, and weights can then be computed using the algorithm of Björck and Golub [BG73] described above, by setting

$$\boldsymbol{Q} = \boldsymbol{U}_{:,1:k}, \qquad \boldsymbol{R} = \boldsymbol{\Sigma}_{1:k,1:k}\boldsymbol{V}_{:,1:k}^\top, \qquad \boldsymbol{Q}' = \boldsymbol{U}'_{:,1:k}, \qquad \boldsymbol{R}' = \boldsymbol{\Sigma}'_{1:k,1:k}{\boldsymbol{V}'}^\top_{:,1:k}. \tag{32.6}$$

Raghu et al. [Rag+17] suggest retaining enough components to explain 99% of the variance, i.e., setting k to the minimum value such that $\|\boldsymbol{\sigma}_{1:k}\|^2/\|\boldsymbol{\sigma}\|^2 = 0.99$ and k' to the minimum value such that $\|\boldsymbol{\sigma}'_{1:k'}\|^2/\|\boldsymbol{\sigma}'\|^2 = 0.99$. However, for a fixed value of $\min(k, k')$, CCA-based similarity measures increase with the value of $\max(k, k')$. Representations that require more components to explain 99% of the variance of representations may thus appear "more similar" to all other representations than representations with more rapidly decaying singular value spectra. In practice, it is often better to set k and k' to the same fixed value.

Projection-weighted CCA (PWCCA) [MRB18] instead weights the correlations by a measure of the variability in the original representation that they explain. The resulting similarity measure is

$$\text{PWCCA}(\boldsymbol{X}, \boldsymbol{Y}) = \frac{\sum_{i=1}^{p} \alpha_i \rho_i}{\sum_{i=1}^{p} \alpha_i}, \qquad \alpha_i = \|(\boldsymbol{Y}\boldsymbol{w}'_i)^\top \boldsymbol{Y}\|_1. \tag{32.7}$$

PWCCA enjoys somewhat widespread use in representational similarity literature, but it has potentially undesirable properties. First, it is asymmetric; its value depends on which of the two representations is used for the weighting. Second, it is unclear why weightings should be determined using the L^1 norm, which is not invariant to rotations of the representation. It is arguably more intuitive to weight the correlations directly by the amount of variance in the representation that the canonical component explains, which corresponds to replacing the L^1 norm with the squared L^2 norm. The resulting similarity measure is

$$R^2_{\text{LR}}(\boldsymbol{X}, \boldsymbol{Y}) = \frac{\sum_{i=1}^{p} \alpha'_i \rho_i^2}{\sum_{i=1}^{p} \alpha'_i}, \qquad \alpha'_i = \sum_{j=1}^{p_2} \|(\boldsymbol{Y}\boldsymbol{w}'_i)^\top \boldsymbol{Y}\|_2^2. \tag{32.8}$$

As shown by Kornblith et al. [Kor+19], when $p_2 \geq p_1$, this alternative similarity measure is equivalent to the overall variance explained by using linear regression to fit every neuron in $\tilde{\boldsymbol{Y}}$ using the representation $\tilde{\boldsymbol{X}}$,

$$R^2_{\text{LR}}(\boldsymbol{X}, \boldsymbol{Y}) = 1 - \sum_{i=1}^{p_2} \min_{\boldsymbol{\beta}} \|\tilde{\boldsymbol{Y}}_{:,i} - \tilde{\boldsymbol{X}}\boldsymbol{\beta}\|^2 / \|\tilde{\boldsymbol{Y}}\|_{\text{F}}^2. \tag{32.9}$$

Finally, there is also a close relationship between CCA and linear CKA. This relationship can be clarified by writing similarity indexes directly in terms of the singular value decompositions $\tilde{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ and $\tilde{\boldsymbol{Y}} = \boldsymbol{U}'\boldsymbol{\Sigma}'\boldsymbol{V}'^\top$. The left-singular vectors $\boldsymbol{u}_i = \boldsymbol{U}_{:,i}$ correspond to the principal components of $\boldsymbol{X}$ normalized to unit length, and the squared singular values $\lambda_i = \Sigma_{ii}^2$ are the amount of variance that those principal components explain (up to a factor of $1/n$). Given these singular value decompositions, R^2_{CCA}, R^2_{LR}, and linear CKA become:

$$R^2_{\text{CCA}}(\boldsymbol{X}, \boldsymbol{Y}) = \sum_{i=1}^{p_1} \sum_{j=1}^{p_2} \left(\boldsymbol{u}_i^\top \boldsymbol{u}'_j\right)^2 / p_1 \tag{32.10}$$

$$R^2_{\text{LR}}(\boldsymbol{X}, \boldsymbol{Y}) = \sum_{i=1}^{p_2} \sum_{j=1}^{p_2} \lambda'_j \left(\boldsymbol{u}_i^\top \boldsymbol{u}'_j\right)^2 / \sum_{j=1}^{p_2} \lambda'_j \tag{32.11}$$

$$\text{CKA}_{\text{linear}}(\boldsymbol{X}, \boldsymbol{Y}) = \frac{\sum_{i=1}^{p_1} \sum_{j=1}^{p_2} \lambda_i \lambda'_j \left(\boldsymbol{u}_i^\top \boldsymbol{u}'_j\right)^2}{\sqrt{\sum_{i=1}^{p_1} \lambda_i^2} \sqrt{\sum_{j=1}^{p_2} {\lambda'_j}^2}}. \tag{32.12}$$

Thus, these similarity indexes all involve the similarities between all pairs of principal components from $\boldsymbol{X}$ and $\boldsymbol{Y}$, but place different weightings on these similarities according to the fraction of variance that these principal components explain.

32.2.2.3 Comparing representational similarity measures

What properties are desirable in a representational similarity measure is an open question, and this question may not have a unique answer. Whereas evaluations of downstream accuracy approximate real-world use cases for neural network representations, the goal of representational similarity is instead to develop understanding of how representations evolve across neural networks, or how they differ between neural networks with different architectures or training settings.

One way to taxonomize different similarity measures is through the transformations of a representation that they are invariant to. The minimum form of invariance is invariance to permutation of a representation's constituent neurons, which is needed because neurons in neural networks generally have no canonical ordering: for commonly-used initialization strategies, any permutation of a given initialization is equiprobable, and nearly all architectures and optimizers produce training trajectories that are equivariant under permutation. On the other hand, invariance to arbitrary invertible transformations, as provided by mutual information, is clearly undesirable, since many realistic neural networks are injective functions of the input [Gol+19] and thus there always exists an invertible transformation between any pair of representations. In practice, most similarity measures in common use are invariant to rotations (orthogonal transformations) of representations, which implies invariance to permutation. Similarity measures based solely on CCA correlations, such as R^2_{CCA} and $\bar{\rho}$, are invariant to all invertible linear transformations of representations. However, SVCCA and PWCCA are not.

A different way to distinguish similarity measures is to investigate situations where we know the relationships among representations and to empirically evaluate their ability to recover these relationships. Kornblith et al. [Kor+19] propose a simple "sanity check": Given two architecturally identical networks A and B trained from different random initializations, any layer in network A should be more similar to the architecturally corresponding layer in network B than to any other layer. They show that, when considering flattened representations of CNNs, similarity measures based on centered kernel alignment satisfy this sanity check whereas other similarity measures do not. By contrast, when considering representations of individual tokens in Transformers, all similarity measures perform reasonably well. However, Maheswaranathan et al. [Mah+19] show that both CCA and linear CKA are highly sensitive to seemingly innocuous RNN design choices such as the activation function, even though analysis of the fixed points of the dynamics of different networks suggests they all operate similarly.

32.3 Approaches for learning representations

The goal of representation learning is to learn a transformation of the inputs that makes it easier to solve future tasks. Typically the tasks we want the representation to be useful for are not known when learning the representation, so we cannot directly train to improve performance on the task. Learning such generic representations requires collecting large-scale unlabeled or weakly-labeled datasets, and identifying tasks or priors for the representations that one can solve without direct access to the downstream tasks. Most methods focus on learning a parametric mapping $z = f_\theta(x)$ that takes an input x and transforms it into a representation z using a neural network with parameters θ.

The main challenge in representation learning is coming up with a task that requires learning a good representation to solve. If the task is too easy, then it can be solved without learning an interesting transformation of the inputs, or by learning a *shortcut*. If a task is too different from the downstream task that the representation will be evaluated on, then the representation may also not be useful. For example, if the downstream task is object detection, then the representation needs to encode both the identity and location of objects in the image. However, if we only care about classification, then the representation can discard information about position. This leads to approaches for learning representations that are often not generic: different training tasks may perform better for different downstream tasks.

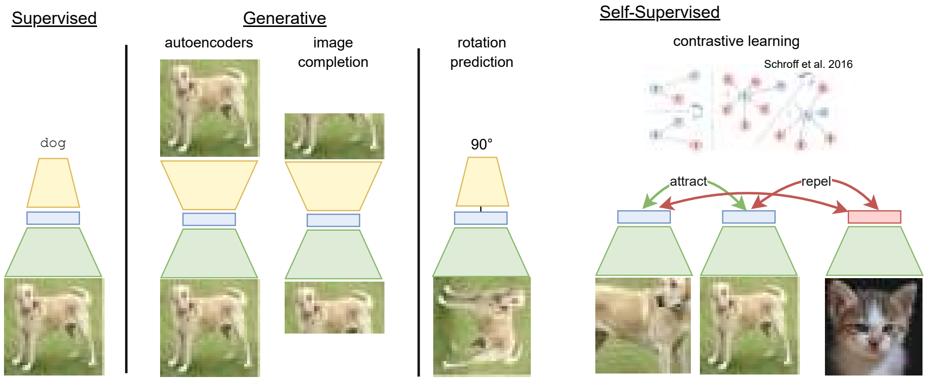


Figure 32.2: Approaches for representation learning from images. An input image is encoded through a deep neural network (green) to produce a representation (blue). An additional shallow or deep neural network (yellow) is often used to train the representation, but is thrown out after the representation is learned when solving downstream tasks. In the supervised case, the mapping from the representation to logits is typically linear, while for autoencoders the mapping from representation to images can be highly complex and stochastic. Unlike supervised or generative approaches, contrastive methods rely on other datapoints in the form of positive pairs (often created through data augmentation) and negative pairs (typically other datapoints) to learn a representation.

In Figue 32.2, we outline three approaches we will discuss for representation learning. **Supervised** approaches train on large-scale supervised or weakly-supervised data using standard supervised losses. **Generative** approaches aim to learn generative models of the dataset or parts of a dataset. **Self-supervised** approaches are based on transformation prediction or multi-view learning, where we design a task that where labels can be easily synthesized without needing human input.

32.3.1 Supervised representation learning and transfer

The first major successes in visual representation learning with deep learning came from networks trained on large labeled datasets. Following the discovery that supervised deep neural networks could outperform classical computer vision models for natural image classification [KSH12b; CMS12], it became clear that the representations learned by these networks could outperform handcrafted features used across a wide variety of tasks [Don+14; SR+14; Oqu+14; Gir+14]. Although unsupervised visual representation learning has recently achieved competitive results on many domains, supervised representation learning remains the dominant approach.

Larger networks trained on larger datasets generally achieve better performance on both pretraining and downstream tasks. When other design choices are held fixed, architectures that achieve higher accuracy during pretraining on natural image datasets such as ImageNet also learn better representations for downstream natural image tasks, as measured by both linear evaluation and fine-tuned accuracy [KSL19; TL19; Zha+19a; Zha+21; Abn+21]. However, when the domain shift from the pretraining task to the downstream task becomes larger (e.g., from ImageNet to medical imaging), the correlation between pretraining and downstream accuracy can be much lower [Rag+19;

Ke+21; Abn+21]. Studies that vary pretraining dataset size generally find that larger pretraining datasets yield better representations for downstream tasks [HAE16; Mah+18; Kol+20; Zha+21; Abn+21], although there is an interaction between model size and dataset. When training small models with the intention of transferring to a specific downstream task, it is sometimes preferable to pretrain on a smaller dataset that is more closely related to that task rather than a larger dataset that is less closely related [Cui+18; Mah+18; Ngi+18; Kol+20], but larger models seem to derive greater benefit from larger, more diverse datasets [Mah+18; Kol+20].

Whereas scaling the architecture and dataset size generally improves both pretraining and downstream accuracy, other design choices can improve pretraining accuracy at the expense of transfer, or vice versa. Regularizers such as penultimate layer dropout and label smoothing improve accuracy on pretraining tasks but produce worse representations for downstream tasks [KSL19; Kor+21]. Although most convolutional neural networks are trained with batch normalization, Kolesnikov et al. [Kol+20] find that the combination of group normalization and weight standardization leads to networks that perform similarly on pretraining tasks but substantially better on transfer tasks. Adversarial training produces networks that perform worse on pretraining tasks as compared to standard training, but these representations transfer better to other tasks [Sal+20]. For certain combinations of pretraining and downstream datasets, increasing the amount of weight decay on the network's final layer can improve transferability at the cost of pretraining accuracy [Zha+21; Abn+21].

The challenge of collecting ever-larger pretraining datasets has led to the emergence of **weakly-supervised representation learning**, which eschews the expensive human annotations of datasets such as ImageNet and instead relies on data that can be readily collected from the Internet, but which may have greater label noise. Supervision sources include hashtags accompanying images on websites such as Instagram and Flickr [CG15; Iza+15; Jou+16; Mah+18], image labels obtained automatically using proprietary algorithms involving user feedback signals [Sun+17; Kol+20], or image captions/alt text [Li+17a; SPL20; DJ21; Rad+21; Jia+21]. Hashtags and automatic labeling give rise to image classification problems that closely resemble their more strongly supervised counterparts. The primary difference versus standard supervised representation learning is that the data are noisier, but in practice, the benefits of more data often outweigh the detrimental effects of the noise.

Image-text supervision has provided more fertile ground for innovation, as there are many different ways of jointly processing text and images. The simplest approach is again to convert the data into an image classification problem, where the network is trained to predict which words or n-grams appear in the text accompanying a given image [Li+17a]. More sophisticated approaches train image-conditional language models [DJ21] or masked language models [SPL20], which can make better use of the structure of the text. Recently, there has been a surge in interest in *contrastive* image/text pretraining models such as CLIP [Rad+21] and ALIGN [Jia+21], details of which we discuss in Section 32.3.4. These models process images and text independently using two separate "towers", and learn an embedding space where embeddings of images lie close to the embeddings of the corresponding text. As shown by Radford et al. [Rad+21], contrastive image/text pretraining learns high-quality representations faster than alternative approaches.

Beyond simply learning good visual representations, pretrained models that embed image and text in a common space enable **zero-shot transfer** of learned representations. In zero-shot transfer, an image classifier is constructed using only textual descriptions of the classes of interest, without any images from the downstream task. Early co-embedding models relied on pretrained image models and word embeddings that were then adapted to a common space [Fro+13], but contrastive image/text pretraining provides a means to learn co-embedding models end-to-end. Compared to linear classifiers trained using image embeddings, zero-shot classifiers typically perform worse, but zero-shot classifiers are far more robust to distribution shift [Rad+21].

32.3.2 Generative representation learning

Supervised representation learning often fails to learn representations for tasks that differ significantly from the task the representation was trained on. How can we learn representations when the task we wish to solve differs a lot from tasks where we have large labeled datasets?

Generative representation learning aims to model the entire distribution of a dataset $q(x)$ with a parametric model $p_\theta(x)$. The hope of generative representation learning is that. if we can build models that can create all the data that we have seen, then we implicitly may learn a representation that can be used to answer any question about the data, not just the questions that are related to a supervised task for which we have labels. For example, in the case of digit classification, it is hard to collect labels for the style of a handwritten digit, but if the model has to product all possible handwritten digits in our dataset it needs to learn to produce digits with different styles. On the other hand, supervised learning to classify digits aims to learn a representation that is invariant to style.

There are two main approaches for learning representations with generative models: (1) latent-variable models that aim to capture the underlying factors of variation in data with latent variables z that act as the representation (see the chapter on VAEs, Chapter 21), and (2) fully-observed models where a neural architecture is trained with a tractable generative objective (see the chapters on AR models, Chapter 22, and flow models, Chapter 23), and then a representation is extracted from the learned architecture.

32.3.2.1 Latent-variable models

One criterion for learning a good representation of the world is that it is useful for synthesizing observed data. If we can build a model that can create new observations, and has a simple set of latent variables, then hopefully this model will learn variables that are related to the underlying physical process that created the observations. For example, if we are trying to model a dataset of 2d images of shapes, knowing the position, size, and type of the shape would enable easy synthesis of the image. This approach to learning is known as **analysis-by-synthesis**, and is a theory of perception that aims at identifying a set of underlying latent factors (analysis) that could be used to synthesize observations [Rob63; Bau74; LM03]. Our goal is to learn a generative model $p_\theta(x, z)$ over the observations x and latents z, with parameters θ. Given an observation x, performing the analysis step to extract a representation requires running inference to sample or compute the posterior mean of $p_\theta(z|x)$. Different choices for the model $p_\theta(x, z)$ and inference procedure for $p_\theta(z|x)$ represent different ways of learning representations from a dataset.

Early work on deep latent-variable generative models aimed to learn stacks of features often based on training simple energy-based models or directed sparse coding models, each of which could explain the previous set of latent factors, and which learned increasingly abstract representation [HOT06b; Lee+09; Ran+06]. Bengio, Courville, and Vincent [BCV13] provide an overview of several methods based on stacking latent-variable generative modeling approaches to learn increasingly abstract representation. However greedy approaches to generative representation learning have failed to scale to larger natural datasets.

If the generative process that created the data is simple and can be described, then encoding that structure into a generative model is a tremendously powerful way of learning useful and robust representations. Lake, Salakhutdinov, and Tenenbaum [LST15] and George et al. [Geo+17] use knowledge of how characters are composed of strokes to build hierarchical generative models with representations that excel at several downstream tasks. However, for many real-world datasets the generative structure is not known, and the generative model must also be learned. There is often a tradeoff between imposing structure in the generative process (such as sparsity) vs. learning that structure from data.

Directed latent-variable generative models have proven easier to train and scale to natural datasets. Variational autoencoders (Chapter 21) train a directed latent-variable generative model with variational inference, and learn a prior $p_\theta(z)$, decoder $p_\theta(x|z)$, and an amortized inference network $q_\phi(z|x)$ that can be used to extract a representation on new datapoints. Higgins et al. [Hig+17b] show β-VAEs (Section 21.3.1) are capable of learning latent variables that correspond to factors of variation on simple synthetic datasets. Kingma et al. [Kin+14b] and Rasmus et al. [Ras+15] demonstrate improved performing on semi-supervised learning with VAEs. While there have been several recent advances to scale up VAEs to natural datasets [VK20b; Chi21b], none of these methods have yet led to representations that are competitive for downstream tasks such as classification or segmentation.

Adversarial methods for training directed latent-variable models have also proven useful for representation learning. In particular, GANs (Chapter 26) trained with encoders such as BiGAN [DKD17], ALI [Dum+17], and [Che+16] were able to learn representations on small scale datasets that performed well at object classification. The discriminators from GANs have also proven useful for learning representations [RMC16b]. More recently, these methods were scaled up to ImageNet in BigBiGAN [DS19], with learned representations that performed strongly on classification and segmentation tasks.

32.3.2.2 Fully observed models

The neural network architectures used in fully observed generative models can also learn useful representations without the presence of latent-variables. ImageGPT [Che+20a] demonstrate that an autoregressive model trained on pixels can learn internal representations that excel at image classification. Unlike with latent-variable models where the representation is often thought of as the latent variables, ImageGPT extracted representations from the deterministic layers of the transformer architecture used to compute future tokens. Similar approaches have shown progress for learning features in language modeling [Raf+20b], however alternative objectives, based on masked training (as in BERT, [Dev+19]), often leads to better performance.

32.3.2.3 Autoencoders

A related set of methods for representation learning are based on learning a representation from which the original data can be reconstructed. These methods are often called **autoencoders** (see Section 16.3.3), as the data is encoded in a way such that the input data itself can be recreated. However, unlike generative models, they cannot typically be used to synthesize observations from scratch or assign likelihoods to observations. Autoencoders learn an encoder that outputs a representation $z = f_\theta(x)$, and a decoder $g_\phi(z)$ that takes the representation z and tries to recreate the input data, x. The quality of the approximate reconstruction , $\hat{x} = g_\phi(z)$ is often measured using a domain-specific loss, for example mean-squared error for images:

$$\mathcal{L}(\theta, \phi) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \|x - g_\phi(f_\theta(x))\|_2^2. \tag{32.13}$$

If there are no constraints on the encoder or decoder, and the dimensionality of the representation z matches the dimensionality of the input x, then there exists a trivial solution to minimize the autoencoding objective: set both f_θ and g_ϕ to identity functions. In this case the representation has not learned anything interesting, and thus in practice an additional regularizer is often placed on the learned representation.

Reducing the dimensionality of the representation z is one effective mechanism to avoid trivial solutions to the autoencoding objective. If both the encoder and decoder networks are linear, and the loss is mean-squared-error, then the resulting linear autoencoder model can learn the principal components of a dataset [Pla18].

Other methods maintain higher-dimensional representations by adding sparsity (for example, penalties on $\|z\|_1$ in Ng et al. [Ng+11]) or smoothness regularizers [Rif+11], or adding noise to the input [Vin+08] or intermediate layers of the network [Sri+14b; PSDG14]. These added regularizers aim to bias the encoder and decoder to learn representations that are not just the identity function, but instead are nonlinear transformations of the input that may be useful for downstream tasks. See Bengio, Courville, and Vincent [BCV13] for a more detailed discussion of regularized autoencoders and their applications. A recent re-evaluation of several algorithms based on iteratively learning features by stacked regularized autoencoders have been shown to degrade performance versus training end-to-end from scratch [Pai+14]. However, we will see in Section 32.3.3.1 that denoising autoencoders have shown promise for representation learning in discrete domains and when applied with more complex noise and masking patterns.

32.3.2.4 Challenges in generative representation learning

Despite several success in generative representation learning, they have empirically fallen behind. Generative methods for representation learning have to learn to match complex high-dimensional and diverse training datasets, which requires modeling all axis of variation of the inputs, regardless of whether they are semantically relevant for downstream tasks. For example, the exact pattern of blades of grass in an image matter for generation quality, but are unlikely to be useful for many of the semantic evaluations that are typically used. Ways to bias generative models to focus on the semantic features and ignore "noise" in the input is an open area of research.

32.3.3 Self-supervised representation learning

When given large amounts of labeled data, standard supervised learning is a powerful mechanism for training deep neural networks. When only presented with unlabeled data, building generative models requires modeling all variations in a dataset, and is often not explicit about what is the signal and noise that we aim to capture in a representation. The methods and architectures for building these generative models also differs substantially from those of supervised learning, where largely feedforward architectures are used to predict low-dimensional representations. Instead of trying to model all aspects of variation, self-supervised learning aims to design tasks where labels can be generated cheaply, and help to encode the structure of what we may care about for other downstream tasks. Self-supervised learning methods allow us to apply the tools and techniques of supervised learning to unlabeled data by designing a task for which we can cheaply produce labels.

In the image domain, several self-supervised tasks, also known as **pretext tasks**, have been proven effective for learning representations. Models are trained to perform these tasks in a supervised fashion using data generated by the pretext task, and then the learned representation is transferred to a target task of interest (such as object recognition), by training a linear classifier or fine-tuning the model in a supervised fashion.

32.3.3.1 Denoising and masked prediction

Generative representation learning is challenging because generative models must learn to produce the entire data distribution. A simpler option is *denoising*, in which some variety of noise is added to the input and the model is trained to reconstruct the noiseless input. A particularly successful variant of denoising is *masked prediction*, in which input patches or tokens are replaced with uninformative masks and the network is trained to predict only these missing patches or tokens.

The **denoising autoencoder** [Vin+08; Vin+10a] was the first deep model to exploit denoising for representation learning. A denoising autoencoder resembles a standard autoencoder architecturally, but it is trained to perform a different task. Whereas a standard autoencoder attempts to reconstruct its input exactly, a denoising autoencoder attempts to produce a noiseless output from a noisy input. Vincent et al. [Vin+08] argue that the network must learn the structure of the data manifold in order to solve the denoising task.

Newer approaches retain the conceptual approach of the denoising autoencoder, but adjust the masking strategy and objective. **BERT** [Dev+18] introduced the **masked language modeling** task, where 15% of the input tokens are selected for masking and the network is trained to predict them. 80% of the time, these tokens are replaced with an uninformative `[MASK]` token. However, the `[MASK]` token does not appear at fine-tuning time, producing some domain shift between pretraining and fine-tuning. Thus, 10% of the time, tokens are replaced with random tokens, and 10% of the time, they are left intact. BERT and the masked language modeling task have been extremely influential for representation learning in natural language processing, inspiring substantial follow-up work [Liu+19c; Jos+20].

Although denoising-based approaches to representation learning were first employed for computer vision, they received little attention for the decade that followed. Vincent et al. [Vin+08] greedily trained stacks of up to three denoising autoencoders that were then fine-tuned end-to-end to perform digit classification, but greedy unsupervised pretraining was abandoned as it was shown that it was possible to attain good performance using CNNs and other architectures trained end-to-end. Context encoders [Pat+16] mask contiguous image regions and train models to perform inpainting, achieving transfer learning performance competitive with other contemporary unsupervised visual representation learning methods. The use of image colorization as a pretext task [ZIE16; ZIE17] is also related to denoising in that colorization involves reconstructing the original image from a corrupted input, although generally color is dropped in a deterministic fashion rather than stochastically.

Recently, the success of BERT in NLP has inspired new approaches to visual representation learning based on masked prediction. Image GPT [Che+20a] trained a transformer directly upon pixels to perform a BERT-style masked image modeling task. While the resulting model achieves very high accuracy when fine-tuned CIFAR-10, the cost of self-attention is quadratic in the number of pixels,

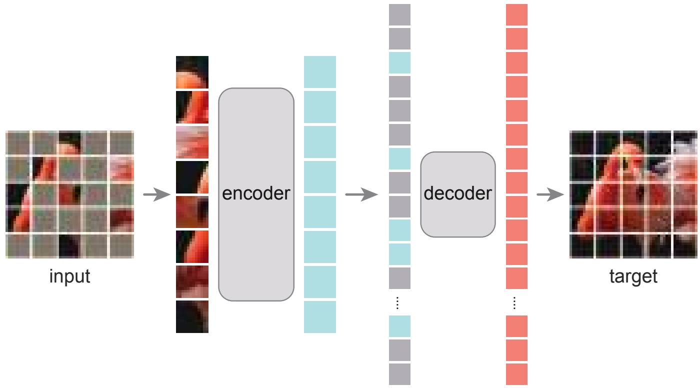


Figure 32.3: Masked autoencoders learn a representation of images by randomly masking out input patches and trying to predict them (from He et al. [He+21]).

limiting applicability to larger image sizes. **BEiT** [Bao+22b] addresses this challenge by combining the idea of masked image modeling with the patch-based architecture of vision transformers [Dos+21]. BEiT splits images into 16×16 pixel image patches and then discretizes these patches using a discrete VAE [Ram+21b]. At training time, 40% of tokens are masked. The network receives continuous patches as input and is trained to predict the discretized missing tokens using a softmax over all possible tokens.

The **masked autoencoder** or **MAE** [He+22] further simplifies the masked image modeling task (see Figure 32.3). The MAE eliminates the need to discretize patches and instead predicts the constituent pixels of each patch directly using a shallow decoder trained with L_2 loss. Because the MAE encoder operates only on the unmasked tokens, it can be trained efficiently even while masking most (75%) of the tokens. Models pretrained using masked prediction and then fine-tuned with labels currently hold the top positions on the ImageNet leaderboard among models trained without additional data [He+22; Don+21].

32.3.3.2 Transformation prediction

An even simpler approach to representation learning involves applying a transformation to the input image and then predicting the transformation that was applied (see Figure 32.4). This prediction task is usually formulated as a classification problem. For visual representation learning, transformation prediction is appealing because it allows reusing exactly the same training pipelines as standard supervised image classification. However, it is not clear that networks trained to perform transformation prediction tasks learn rich visual representations. Transformation prediction tasks are potentially susceptible to "shortcut" solutions, where networks learn trivial features that are nonetheless sufficient to solve the task with high accuracy. For many years, self-supervised learning methods based on transformation prediction were among the top-performing methods, but they have since been displaced by newer methods based on contrastive learning and masked prediction.

Some pretext tasks operate by cutting images into patches and training networks to recover the spatial arrangement of the patches. In context prediction [DGE15], a network receives two adjacent

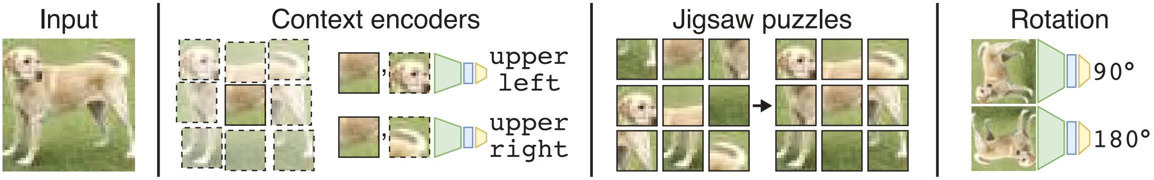


Figure 32.4: Transformation prediction involves training neural networks to predict a transformation applied to the input. Context encoders predict the position of a second crop relative to the first. The jigsaw puzzle task involves predicting the way in which patches have been permuted. Rotation prediction involves predicting the rotation that was applied to the input.

image patches as input and is trained to recover their spatial relationship by performing an eight-class classification problem. To prevent the network from directly matching the pixels at the patch borders, the two patches must be separated by a small variable gap. In addition, to prevent networks from using chromatic aberration to localize the patches relative to the lens, color channels must be distorted or stochastically dropped. Other work has trained networks to solve jigsaw puzzles by splitting images into a 3×3 grid of patches [NF16]. The network receives shuffled patches as input and learns to predict how they were permuted. By limiting the permutations to a subset of all possibilities, the jigsaw puzzle task can be formulated as a standard classification task [NF16].

Another widely used pretext task is rotation prediction [GSK18], where input images are rotated 0, 90, 180, or 270 degrees and networks are trained to classify which rotation was applied. Although this task is extremely simple, the learned representations often perform better than those learned using patch-based methods [GSK18; KZB19]. However, all approaches based on transformation prediction currently underperform masked prediction and multiview approaches on standard benchmark datasets such as CIFAR-10 and ImageNet.

32.3.4 Multiview representation learning

The field of **multiview representation learning** aims to learn a representation where "similar" inputs or *views* of an input are mapped nearby in the representation space, and "dissimilar" inputs are mapped further apart. This representation space is often high-dimensional, and relies on collecting data or designing a task where one can generative "positive" pairs of examples that are similar, and "negative" pairs of examples that are dissimilar. There are many motivations and objectives for multiview representation learning, but all rely on coming up with sets of positive pairs, and a mechanism to prevent all representations from collapsing to the same point. Here we use the term multiview representation learning to encompass **contrastive learning** which combines positive and negative pairs, metric learning, and "non-contrastive" learning which eliminates the need for negative pairs.

Unlike generative methods for representation learning, multiview representation learning makes it easy to incorporate prior knowledge about what inputs should be closer in the embedding space. Furthermore, these inputs need not be from the same modality, and thus multiview representation learning can be applied with rich multimodal datasets. The simplicity of the way in which prior knowledge can be incorporated into a model through data has made multiview representation learning one of the most powerful and performant methods for learning representations.

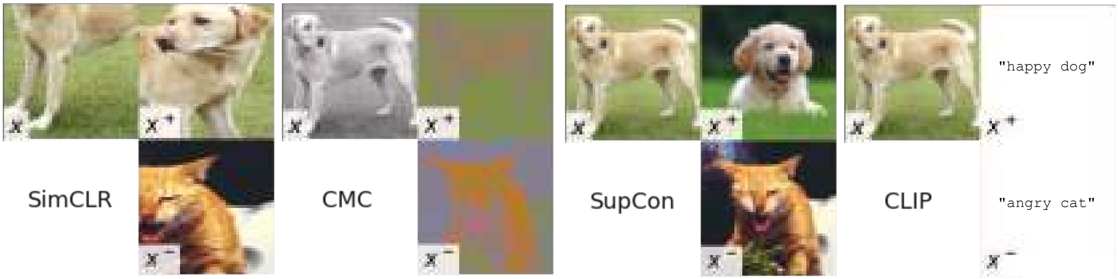


Figure 32.5: Positive and negative pairs used by different multiview representation learning methods.

While there are a variety of methods for multiview representation learning, they all involve a repulsion component that pulls positive pairs closer together in embedding space, and a mechanism to prevent collapse of the representation to a single point in embedding space. We begin by describing loss functions for multiview representation learning and how they combine attractive and repulsive terms to shape the representation, then discuss the role of view generation, and finally practical considerations in deploying multiview representation learning.

32.3.4.1 View selection

Multiview representation learning depends on a datapoint or "anchor" x, a positive example x^+ that x will be attracted to, and zero or more negative examples x^- that x is repelled from. We assume access to a data-generating process for the positive pair: $p^+(x, x^+)$, and a process that generates the negative examples given the datapoint x: $p^-(x^-|x)$. Typically $p^+(x, x^+)$ generate (x, x^+) that are different augmentations of an underlying image from the dataset, and x^- represents an augmented view of a different random image from the dataset. The generative process for x^- is then independent of x, i.e., $p^-(x^-|x) = p^-(x^-)$.

The choice of views used to generate positive and negative pairs is critical to the success of representation learning. Figure 32.5 shows the positive pair (x, x^+) and negative x^- for several methods which we discuss below: SimCLR, CMC, SupCon, and CLIP.

SimCLR [Che+20c] creates positive pairs by applying two different data augmentations defined by transformations t and t' to an initial image x_0 twice: $x = t(x_0), x^+ = t(x_0)$. The data augmentations used are random crops (with horizontal flips and resize), color distortion, and Gaussian blur. The strengths of these augmentations (e.g., the amount of blur) impact performance and are typically treated as a hyperparameter.

If we access to additional information, such as a categorical label, we can use this to select positive pairs with the same label, and negative pairs with different labels. The resulting objective, when used with a contrastive loss, is called **SupCon** [Kho+20], and resembles neighborhood component analysis [Gol+04]. It was shown to improve robustness when compared to standard supervised learning.

Contrastive multiview coding (CMC) [TKI20] generates views by splitting an initial image into orthogonal dimensions, such as the luma and chroma dimensions. These views are now no longer in the same space (or same dimensionality), and thus we must learn different encoders for the different inputs. However, the output of these encoders all live in the same-dimensional embedding space, and can be used in contrastive losses. At test-time, we can then combine embeddings from these different views through averaging or concatenation.

Views do not need to be from the same modality. **CLIP** [Rad+21] uses contrastive learning on image-text pairs, where x is an image, and x^+ and x^- are text descriptions. When applied to massive datasets of image-text pairs scraped from the Internet, CLIP is able to learn robust representations without any of the additional data augmentation needed by SimCLR or other image-only contrastive methods.

In most contrastive methods, negative examples are selected by randomly choosing x^+ from other elements in a minibatch. However, if the batch size is small it may be the case that none of the negative examples are close in embedding space to the positive example, and so learning may be slow. Instead of randomly choosing negatives, they may be chosen more intelligently through **hard negative mining** that selects negative examples that are close to the positive example in embedding space [Fag+18]. This typically requires maintaining and updating a database of negative examples over the course of training; this incurs enough computational overhead that the technique is infrequently used. However, reweighting examples within a minibatch can also lead to improved performance [Rob+21].

The choice of positive and negative views directly impacts what features are learned and what invariances are encouraged. Tian et al. [Tia+20] discusses the role of view selection on the learned representations, showing how choosing positives based on shared attributes (as in SupCon) can lead to learning those attributes or ignoring them. They also present a method for learning views (whereas all prior approaches fix views) based on targeting a "sweet spot" in the level of mutual information between the views that is neither too high or too low. However, understanding what views will work well for what downstream tasks remains an open area of study.

32.3.4.2 Contrastive losses

Given p^+ and p^-, we seek loss functions that learn an embedding $f_\theta(x)$ where x and x^+ are close in the embedding space, while x and x^- are far apart. This is called **metric learning**.

Chopra, Hadsell, LeCun, et al. [CHL+05] present a family of objectives that implements this intuition by enforcing the distance between negative pairs to always be at least ϵ bigger than the distance between positive pairs. The contrastive loss as instantiated in [HCL06] is:

$$\mathcal{L}_{\text{contrastive}} = \mathbb{E}_{x,x^+,x^-}\left[\|f_\theta(x) - f_\theta(x^+)\|^2 + \max(0, \epsilon - \|f_\theta(x) - f_\theta(x^-)\|^2\right]. \tag{32.14}$$

This loss pulls together the positive pairs by making the squared ℓ_2 distance between them small, and tries to ensure that negative pairs are at least a distance of ϵ apart. One challenge with using the contrastive loss in practice is tuning the hyperparameter ϵ.

Similarly, the **triplet loss** [SKP15] tries to ensure that the positive pair (x, x^+) is always at least some distance ϵ closer to each other than the negative pair (x, x^-):

$$\mathcal{L}_{\text{triplet}} = \mathbb{E}_{x,x^+,x^-}\left[\max(0, \|f_\theta(x) - f_\theta(x^+)\|^2 - \|f_\theta(x) - f_\theta(x^-)\|^2 + \epsilon)\right]. \tag{32.15}$$

A downside to the triplet loss approach is that one has to be careful about choosing hard negatives: if the negative pair is already sufficiently far away then the objective function is zero and no learning occurs.

An alternative contrastive loss which has gained popularity due to its lack of hyperparameters and empirical effectiveness is known as the **InfoNCE loss** [OLV18b] or the **multiclass N-pair loss**

[Soh16]:

$$\mathcal{L}_{\text{InfoNCE}} = -\mathbb{E}_{x,x^+,x^-_{1:M}} \left[\log \frac{\exp f_\theta(x)^T g_\phi(x^+)}{\exp f_\theta(x)^T g_\phi(x^+) + \sum_{i=1}^M \exp f_\theta(x)^T g_\phi(x_i^-)} \right], \tag{32.16}$$

where M are the number of negative examples. Typically the embeddings $f(x)$ and $g(x')$ are ℓ_2-normalized, and an additional hyperparameter τ can be introduced to rescale the inner products [Che+20c]. Unlike the triplet loss, which uses a hard threshold of ϵ, $\mathcal{L}_{\text{InfoNCE}}$ can always be improved by pushing negative examples further away. Intuitively, the InfoNCE loss ensures that the positive pair is closer together than any of the M negative pairs in the minibatch. The InfoNCE loss can be related to a lower bound on the mutual information between the input x and the learned representation z [OLV18b; Poo+19a]:

$$I(X;Z) \geq \log M - \mathcal{L}_{\text{InfoNCE}}, \tag{32.17}$$

and has also been motivated as a way of learning representations through the InfoMax principle [OLV18b; Hje+18; BHB19]. When applying the InfoNCE loss to parallel views that are the same modality and dimension, the encoder f_θ for the anchor x and the positive and negative examples g_ϕ can be shared.

32.3.4.3 Negative-free losses

Negative-free representation learning (sometimes called **non-contrastive representation learning**) learns representations using only positive pairs, without explicitly constructing negative pairs. Whereas contrastive methods prevent collapse by enforcing that positive pairs are closer together than negative pairs, negative-free methods make use of other mechanisms. One class of negative-free objectives includes both attractive terms and terms that prevent collapse. Another class of methods uses objectives that include only attractive terms, and instead relies on the learning dynamics to prevent collapse.

The **Barlow Twins** loss [Zbo+21] is

$$\mathcal{L}_{\text{BT}} = \sum_{i=1}^{p} (1 - \boldsymbol{C}_{ii})^2 + \lambda \sum_{i=1}^{p} \sum_{j \neq i} \boldsymbol{C}_{ij}^2 \tag{32.18}$$

where $\boldsymbol{C}$ is the cross-correlation matrix between two batches of features that arise from the two views. The first term is an attractive term that encourages high similarity between the representations of the two views, whereas the second term prevents collapse to a low-rank representation. The loss is minimized when $\boldsymbol{C}$ is the identity matrix. Similar losses based on ensuring the variance of features being non-zero have also been useful for preventing collapse [BPL21b]. The Barlow Twins loss can be related to kernel-based independence criterion such as HSIC which have also been useful as losses for representation learning [Li+21; Tsa+21].

BYOL (bootstrap your own latents) [Gri+20] and SimSiam [Che+20c] simply minimize the mean squared error between two representations:

$$\mathcal{L}_{\text{BYOL}} = \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}^+} \left[\|g_\phi(f_\theta(\boldsymbol{x})) - f_{\theta'}(\boldsymbol{x}^+)\|^2 \right]. \tag{32.19}$$

Following Grill et al. [Gri+20], g_ϕ is known as the *predictor*, f_θ is the *online network*, and $f_{\theta'}$ is the *target network*. When optimizing this loss function, weights are backpropagated to update ϕ and θ, but optimizing θ' directly leads the representation to collapse [Che+20c]. Instead, BYOL sets θ' as an exponential moving average of θ, and SimSiam sets $\theta' \leftarrow \theta$ at each iteration of training. The reasons why BYOL and SimSiam avoid collapse are not entirely clear, but Tian, Chen, and Ganguli [TCG21] analyze the gradient flow dynamics of a simplified linear BYOL model and show that collapse can indeed be avoided given properly set hyperparameters.

DINO (self-distillation with no labels) [Car+21] is another non-contrastive loss that relies on the dynamics of learning to avoid collapse. Like BYOL, DINO uses a loss that consists only of an attractive term between an online network and a target network formed by an exponential moving average of the online network weights. Unlike BYOL, DINO uses a cross-entropy loss where the target network produces the targets for the online network, and avoids the need for a predictor network. The DINO loss is:

$$\mathcal{L}_{\text{DINO}} = \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}^+} \left[H(f_{\theta'}(\boldsymbol{x})/\tau, \text{center}(f_\theta(\boldsymbol{x}^+))/\tau') \right] . \tag{32.20}$$

where, with some abuse of notation, center is a mean-centering operation applied across the minibatch that contains x^+. Centering the output of the target network is necessary to prevent collapse to a single "class", whereas using a lower temperature $\tau' < \tau$ for the target network is necessary to prevent collapse to a uniform distribution. The DINO loss provides marginal gains over the BYOL loss when performing self-supervised representation learning with vision transformers on ImageNet [Car+21].

32.3.4.4 Tricks of the trade

Beyond view selection and losses, there are a number of useful architectures and modifications that enable more effective multiview representation learning.

Normalizing the output of the encoders and computing cosine similarity instead of predicting unconstrained representations has shown to improve performance [Che+20c]. This normalization bounds the similarity between points between -1 and 1, so an additional temperature parameter τ is typically introduced and fixed or annealed over the course of learning.

While the learned representation with multiview learning are often useful for downstream tasks, the losses when combined with data augmentation typically lead to too much invariance for some tasks. Instead, one can extract an earlier layer in the encoder as the representation, or alternatively, add an additional layer known as a projection head to the encoder before computing the loss [Che+20c]. When training we compute the loss on the output of the projection head, but when evaluating the quality of the representation we discard this additional layer.

Given the summation over negative examples in the denominator of the InfoNCE loss, it is often sensitive to the batch size used for training. In practice, large batch sizes of 4096 or more are needed to achieve good performance with this loss, which can be computationally burdensome. **MoCo** (momentum contrast) [He+20] introduced a memory queue to store negative examples from previous minibatches to expand the size of negatives at each iteration. Additionally, they use a momentum encoder, where the encoder for the positive and negative examples uses an exponential moving average of the anchor encoder parameters. This momentum encoder approach was also found useful in BYOL to prevent collapse. As in BYOL, adding an extra predictor network that maps from the online network to the target network has shown to improve the performance of MoCo, and removes the requirement of a memory queue [CXH21].

The backbone architectures of the encoder networks play a large role in the quality of representations. For representation learning in vision, recent work has switched from ConvNet-based backbones to vision transformers, resulting in larger-scale models with improved performance on several downstream tasks [CXH21].

32.4 Theory of representation learning

While deep representation learning has replaced hand-designed features for most applications, the theory behind what features are learned and what guarantees these methods provide are limited. Here we review several theoretical directions in understanding representation learning: identifiability, information maximization, and transfer bounds.

32.4.1 Identifiability

In this section, we assume a latent-variable generative model that generated the data, where $z \sim p(z)$ are the latent variables, and $x = g(z)$ is a deterministic generator that maps from the latent variables to observations. Our goal is to learn a representation $h = f_\theta(x)$ that inverts the generative model and recovers $h = z$. If we can do this, we say the model is **identifiable**. Oftentimes we are not able to recover the true latent variables exactly, for example the dimensions of the latent variables may be permuted, or individual dimensions may be transformed version of an underlying latent variable: $h_i = f_i(z_i)$. Thus most theoretical work on identifiability focuses on the case of learning a representation that can be permuted and elementwise transformed to match the true latent variables. Such representations are referred to as **disentangled** as the dimensions of the learned representation do not mix together multiple dimensions of the true latent variables.

Methods for recovering are typically based around latent-variable models such as VAEs combined with various regularizers (see Section 21.3.1.1). While several publications showed promising empirical progress, a large-scale study by Locatello et al. [Loc+20a] on disentangled representation learning methods showed that several existing approaches cannot work without additional assumptions on the data or model. Their argument relies on the observation that we can form a bijection f that takes samples from a factorial prior $p(z) = \prod_i p_i(z_i)$ and maps to $z' = f(z)$ that (1) preservers the marginal distribution, and (2) has entirely entangled latents (each dimension of z influences every dimension of z'). Transforming the marginal in this way changes the representation, but preserves the marginal likelihood of the data, and thus one cannot use marginal likelihood alone to identify or distinguish between the entangled and disentangled model. Empirically, they show that past methods largely succeeded due to careful hyperparameter selection on the target disentanglement metrics that require supervised labels. While further work has developed unsupervised methods for hyperparameter that address several of these issues [Dua+20], at this point there are no known robust methods for learning disentangled representations without further assumptions.

To address the empirical and theoretical gap in learning disentangled representations, several papers have proposed using additional sources of information in the form of weakly-labeled data to provide guarantees. In theoretical work on nonlinear ICA [RMK21; Khe+20; Häl+21], this information comes in the form of additional observations for each datapoint that are related to the underlying latent variable through an exponential family. Work on **causal representation learning** has expanded the applicability of these methods and highlighted the settings where such strong assumptions on weakly-labeled data may be attainable [Sch+21c; WJ21; Rei+22]

Alternatively, one can assume access to pairs of observations where the relationship between latent variables is known. In Shu et al. [Shu+19b], they show that one can provably learn a disentangled representation of data when given access to pairs of data where only one of the latent variables is changed at a time. In real world datasets, having access to pairs of data like this is challenging, as not all the latent-variables of the model may be under the control of the data collector, and covering the full space of settings of the latent variable may be prohibitively expensive. Locatello et al. [Loc+20b] develops this method further but leverages a heuristic to detect which latent variable has changed, and shows this performs empirically well, and under some restricted settings may lead to learning disentangled representations.

More recently, [Kiv+21; Kiv+22] showed that it is possible to identify deep latent variable models, such as VAEs, without any side information, provided the latent space prior has the form of a mixture.

32.4.2 Information maximization

When learning representations of an input x, one desideratum is to preserve as much information about x as possible. Any information we discard cannot be recovered, and if that information is useful for a downstream task then performance will decrease. Early work on understanding biological learning by Linsker [Lin88c] and Bell and Sejnowski [BS95b] argued that information maximization or **InfoMax** is a good learning principle for biological systems as it enables the downstream processing systems access to as much sensory input as possible. However, these biological systems aim to communicate information subject to strong constraints, and these constraints can likely be tuned over time by evolution to sculpt the kinds of representations that are learned.

When applying information maximization to neural networks, we are often able to realize trivial solutions which biological systems may not face: being able to losslessly copy the input. Information theory does not "color" the bits, it does not tell us which bits of an input are more important than others. Simply sending the image losslessly maximizes information, but does not provide a transformation of the input that can improve performance according to the metrics in Section 32.2. Architectural and optimization constraints can guide the bits we learn and the bits we dispose of, but we can also leverage additional sources of information, for example labels, to identify which bits to extract.

The **information bottleneck** method (Section 5.6) aims to learn representations Z of an input X that are predictive of another observed variable Y, while being as compressed as possible. The observed variable Y guides the bits learned in the representation Z towards those that are predictive, and penalizes content that does not predict Y. We can formalize the information bottleneck as an optimization problem [TPB00]:

$$\text{maximize}_\theta I(Z;Y) - \beta I(X;Z). \tag{32.21}$$

Estimating mutual information in high dimensions is challenging, but we can form variational bounds on mutual information that are amenable to optimization with modern neural networks, such as variational information bottleneck (VIB, see Section 5.6.2). Approaches built on VIB have shown improved robustness to adversarial examples and natural variations [FA20].

Unlike information bottleneck methods, many recent approaches motivated by InfoMax have no explicit compression objective [Hje+18; BHB19; OLV18b]. They aim to maximize information subject to constraints, but without any explicit penalty on the information contained in the representation.

In spite of the appeal of explaining representation learning with information theory, there are a number of challenges. One of the greatest challenges in applying information theory to understand the content in learned representations is that most learned representations have determinstic encoders, $z = f_\theta(x)$ that map from a continuous input x to a continuous representation z. These mappings can typically preserve infinite information about the input. As mutual information estimators scale poorly with the true mutual information, estimating MI in this setting is difficult and typically results in weak lower bounds.

In the absence of constraints, maximizing information between an input and a learned representation has trivial solutions that do not result in any interesting transformation of the input. For example, the identity mapping $z = x$ maximizes information but does not alter the input. Tschannen et al. [Tsc+19] show that for invertible networks where the true mutual information between the input and representation is infinite, maximizing estimators of mutual information can result in meaningful learned represenations. This highlights that the geometric dependence and bias of these estimators may have more to do with their success for representation learning than the information itself (as it is infinite throughout training).

There have been several proposed methods for learning stochastic representations that constrain the amount of information in learned representations [Ale+17]. However, these approaches have not yet resulted in improved performance on most downstream tasks. Fischer and Alemi [FA20] shows that constraining information can improve robustness on some benchmarks, but scaling up models and datasets with determinstic representations currently presents the best results [Rad+21]. More work is needed to identify whether constraining information can improve learned representations.

33 Interpretability

This chapter is written by Been Kim and Finale Doshi-Velez.

33.1 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 (e.g., processing paperwork, flagging potential fraud) do not harm their users or society at large. Models with large impacts on health and welfare (e.g., recommending treatment, driving autonomously) must not only be safe but often also function collaboratively with their users.

However, determining whether a model is harmful is not easy. Specific performance metrics may be too narrowly focused—e.g., just because an autonomous car stays in lane does not mean it is safe. Indeed, the narrow objectives used in common decision formalisms such as Bayesian decision theory (Section 34.1), multi-step decision problems (Chapter 34), and reinforcement learning (Chapter 35) can often be easily exploited (e.g.„ reward hacking). Incomplete sets of metrics also result in models that learn shortcuts that do not generalize to new situations (e.g., [Gei+20b]). Even when one knows the desired metrics, those metrics can be hard to estimate with limited data or a distribution shift (Chapter 19). Finally, normative concepts, such as fairness, may be impossible to fully formalize. As a result, not only may unexpected and irreversible harms occur (e.g., an adverse drug reaction) but more subtle harms may go unnoticed until sufficient reporting data accrues [Amo+16].

Interpretability allows human experts to inspect a model. Alongside traditional statistical measures of performance, this human inspection can help expose issues and thus mitigate potential harms. Exposing the workings of a model can also help people identify ways to incorporate information they have into a final decision. More broadly, even when we are satisfied with a model's performance, we may be interested in understanding *why* they work to gain scientific and operational insights. For example, one might gain insights in language structure 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 helps humans to communicate better with machines to accomplish our tasks better.

In this chapter, we lay out the role and terminologies in interpretable ML before introducing methods, properties, and evaluation of interpretability methods.

33.1.1 The role of interpretability: unknowns and under-specifications

As noted above, ensuring that models behave as desired is challenging. In some cases, the desired behavior can be guaranteed by design, such as certain notions of privacy via differentially-private learning algorithms or some chosen mathematical metric of fairness. In other cases, tracking various metrics, such as adverse events or subgroup error rates, may be the appropriate and sufficient way to identify concerns. Much of this textbook deals with uncertainty quantification: basic models in Chapter 3, Bayesian neural networks in Chapter 17, Gaussian processes in Chapter 18). When well-calibrated uncertainties can be computed, they may provide sufficient warning that a model's output may be suspect.

However, in many cases, the ultimate goal may be fundamentally impossible to fully specify and thus formalize. For example, Section 20.4.8 discusses the challenge of evaluating the quality of samples from a generative model. In such cases, human inspection of the machine learning model may be necessary. Below we describe several examples.

Blindspot discovery. Inspection may reveal **blindspots** in our modeling, objective, or data [Bad+18; Zec+18b; Gur+18]. For example, suppose a company has trained a machine learning system for credit scoring. The model was trained on a relatively affluent, middle-aged population, and now the company is considering using it on a different, college-aged population. Suppose that inspection of the model reveals that it relies heavily on the applicant's mortgage payments. Not only might this suggest that the model might not transfer well to the college population, but it might encourage us to check for bias in the existing application because we know historical biases have prevented certain populations from achieving home ownership (something that a purely quantitative definition of fairness may not be able to recognize). Indeed, the most common application of interpretability in industry settings is for engineers to debug models and make deployment decisions [Pai].

Novel insights. Inspection may catalyze the discovery of **novel insights.** For example, suppose an algorithm determines that surgical procedures fall into three clusters. The surgeries in one of the clusters of patients seem to consistently take longer than expected. A human inspecting these clusters may determine that a common factor in the cluster with the delays is that those surgeries occur in a different part of the hospital, a feature not in the original dataset. This insight may result in ideas to improve on-time surgery performance.

Human+ML teaming. Inspection may empower effective **human+ML interaction and teaming**. For example, suppose an anxiety treatment recommendation algorithm reveals the patient's comorbid insomnia constrained its recommendations. If the patient reports that they no longer have trouble sleeping, the algorithm could be re-run with that constraint removed to get additional treatment options. More broadly, inspection can reveal places where people may wish to adjust the model, such as correcting an incorrect input or assumption. It can also help people use only part of a model in their own decision-making, such as using a model's computation of which treatments unsafe vs. which treatments are best. In these ways, the human+ML team may be able to produce better combined performance than either alone (e.g., [Ame+19; Kam16]).

Individual-level recourse. Inspection can help determine whether a specific harm or error happened in a specific context. For example, if a loan applicant knows what features were used to deny them a loan, they have a starting point to argue that an error might have been made, or that the algorithm denied them unjustly. For this reason, inspectability is sometimes a legal requirement [Zer+19; GF17; Cou16].

As we look at the examples above, we see that one common element is that *interpretability is needed when we need to combine human insights with the ML algorithm to achieve the ultimate goal.*[1] However, looking at the list above also emphasizes that beyond this very basic commonality, *each application and task represents very different needs.* A scientist seeking to glean insights from a clustering on molecules may be interested in global patterns — such as all molecules with certain loop structures are more stable — and be willing to spend hours puzzling over a model's outputs. In contrast, a clinician seeking to make a specific treatment decision may only care about aspects of the model relevant to the specific patient; they must also reach their decision within the time-pressure of an office visit. This brings us to our most important point: the best form of explanation depends on the *context*; interpretability is a means to an end.

33.1.2 Terminology and framework

In broad strokes, "to interpret means to explain or present in understandable terms" [Mer]. Understanding, in turn, involves an alignment of mental models. In interpretable machine learning, that alignment is between what (perhaps part of) the machine learning model is doing and what the user thinks the model is doing.

As a result, interpretable machine learning ecosystem includes not only standard machine learning (e.g.,, a prediction task) but also what information is provided to the human user, in what context, and the user's ultimate goal. The broader *socio-technical system* — the collection of interactions between human, social, organizational, and technical (hardware and software) factors — cannot be ignored [Sel+19]. The goal of interpretable machine learning is to help a user do *their* task, with *their* cognitive strengths and weaknesses, with *their* focus and distractions [Mil19]. Below we define the key terms of this expanded ecosystem and describe how they relate to each other. Before continuing, however, we note that the field of interpretable machine learning is relatively new, and a consensus around terminology is still evolving. Thus, it is always important to define terms.

Two core **social** or **human factors** elements in interpretable machine learning are the *context* and the *end-task*.

Context. We use the term *context* to describe the setting in which an interpretable machine learning system will be used. Who is the user? What information do they have? What constraints are present on their time, cognition, or attention? We will use the terms context and application interchangeably [Sta].

End-task. We use the term *end-task* to refer to the user's ultimate goal. What are they ultimately trying to achieve? We will use the terms end-task and downstream tasks interchangeably.

Three core **technical** elements in interpretable machine learning are the *method*, the *metrics*, and the *properties* of the methods.

1. We emphasize that interpretability is different from manipulation or persuasion, where the goal is to intentionally deceive or convince users of a predetermined choice.

Method. How do we does the interpretability happen? We use the term *explanation* to mean the output provided by the method to the user: interpretable machine learning *methods* provide *explanations* to the users. If the explanation is the model itself, we call the method *inherently interpretable* or *interpretable by design*. In other cases, the model may be too complex for a human to inspect it in its entirety: perhaps it is a large neural network that no human could expect to comprehend; perhaps it is a medium-sized decision tree that could be inspected if one had twenty minutes but not if one needs to make a decision in two minutes. In such cases, the explanation may be a *partial view* of the model, one that is ideally suited for performing the end-task in the given context. Finally, we note that even inherently interpretable models do not reveal everything: one might be able to fully inspect the function (e.g., a two-node decision tree) but not know what data it was trained on or which datapoints were most influential.

Metrics. How is the interpretability method evaluated? Evaluation is one of the most essential and challenging aspects of interpretable machine learning, because we are interested in the **end-task** performance of the *human*, when explanation is provided. We call this the *downstream performance*. Just as different goals in ML require different metrics (e.g., positive predictive value, log likelihood, AUC), different **contexts** and **end-tasks** will have different metrics. For example, the model with the best predictive performance (e.g., log likelihood loss) may not be the model that results in the best downstream performance.

Properties. What characteristics does the explanation have in relation to the model, the context and the end-tasks? Different **contexts** and different **end-tasks** might require different properties. For example, suppose that an explanation is being used to identify ways in which a denied loan applicant could improve their application. Then, it may be important that the explanation only include factors that, if changed, would change the outcome. In contrast, suppose the explanation is being used to determine if the denial was fair. Then, it may be important that the explanation does not leave out any relevant factors. In this way, properties serve as a glue between interpretability methods, contexts and end-tasks: properties allow us to specify and quantify aspects of the explanation relevant to our ultimate end-task goals. Then we can make sure that our interpretability method has those properties.

How they all relate. Formulating an interpretable machine learning problem generally starts by specifying the context and the end-task. Together the context and the end-task imply what metrics are appropriate to evaluate the downstream performance on the end-task and suggest what properties will be important in the explanation. Meanwhile, the context also determines the data and training metric for the ML model. The appropriate choice of explanation methods will depend on the model and properties desired, and it will be evaluated with respect to the end-task metric to determine the downstream performance. Figure 33.1 shows these relationships.

Interpretable machine learning involves many challenges, from computing explanations and optimizing interpretable models and creating explanations with certain properties to understanding the associated human factors. That said, the grand challenge is to (1) understand what properties are needed for different contexts and end-tasks and (2) identify and create interpretable machxine learning methods that have those properties.

A simple example In the following sections, we will expand upon methods for interpretability,

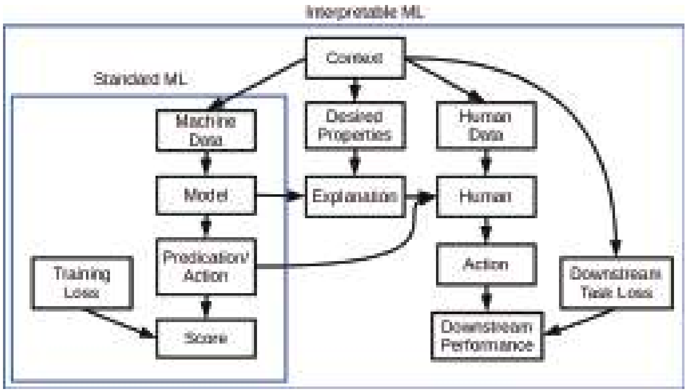


Figure 33.1: The interpretable machine learning ecosystem. While standard machine learning can often abstract away elements of the context and consider only the process of learning models given a data distribution and a loss, interpretable machine is inextricably tied to a socio-technical context.

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 the model before using it to inform our business strategy.) Our *metric* for the downstream performance is whether we correctly determine if the model can be trusted; 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. Two models seem to fit the data approximately equally well:

Model 1:

$$p(\texttt{profit}) = .9 * (\texttt{temperature} > 75) + .1(\texttt{howCleanSidewalk}) \tag{33.1}$$

Model 2:

$$p(\texttt{profit}) = \sigma(.9(\texttt{temperature} - 75)/\texttt{maxTemperature} + .1(\texttt{howCleanSidewalk} - .5)) \tag{33.2}$$

These models are illustrated in Figure 33.2. Both of these models are inherently interpretable in the sense that they are easy to inspect and understand. While we were not explicitly seeking causal models (for that, see Chapter 36), both rely mostly on the temperature, which seems reasonable.

For the sake of this example, suppose that the models above were black boxes, and we could only request partial views of it. We decide to ask the model for the most important features. Let us see what happens when we consider two different ways of computing important features.[2]

2. In the remainder of the chapter we will describe many other ways of creating and computing explanations.

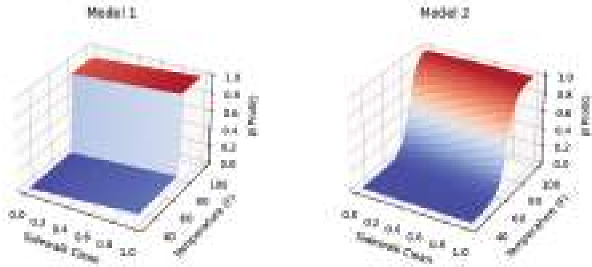


Figure 33.2: Models described in the simple example. Both of these models have the same qualitative characteristics, but different explanation methods will describe these models quite differently, potentially causing confusion.

Our first (feature-based) explanation method computes, for each training point, whether individually changing each feature to its max or min value changes the prediction. Important features are those that change the prediction for many training points. One can think of this explanation method as a variant of computing feature importance based on how important a feature is to the coalition that produces the prediction. In this case, both models will report temperature to be the dominating feature. If we used this explanation to vet our models, we would correctly conclude that both models use the features in a sensible way (and thus may be worth considering for deciding when to open our lemonade stand).

Our second (feature-based) explanation method computes the magnitude of the derivatives of the output with respect to the inputs for each training point. Important features are those that have a large sum of absolute derivatives across the training set. One can think of this explanation method as a variant of computing feature importances based on local geometry. In this case, Model 2 will still report that temperature has higher derivatives. However, Model 1, which has very similar behavior to Model 2, will report that sidewalk cleanliness is the dominating feature because the derivative with respect to temperature is zero nearly everywhere. If we used this explanation to vet our models, we would incorrectly conclude that Model 1 relies on an unimportant feature (and that Model 1 and 2 rely on different features).

What happened? The different explanations had different properties. The first 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 that have the most local curvature. In this example, the first property is more important for the task of determining whether our model can be used to determine our business strategy. [3]

33.2 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 33.1.2, the grand challenge in interpretable machine learning is determining what kinds of properties are needed for

3. Other properties may be important for this end-task. This example is just the simplest one.

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.

33.2.1 Inherently interpretable models: the model is its explanation

We consider certain classes of models inherently interpretable: a person can inspect the full model and, with reasonable effort, understand how inputs become outputs.[4] 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, suppose a model consists of a relatively small set of rules. Then, those rules might suffice as the explanation for end-tasks that do not involve extreme time pressure. (Note: in this way, a model might be inherently interpretable for one end-task and not another.)

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. Earlier in this textbook, we discussed simple models such as HMMs (Section 29.2), generalized linear models (Chapter 15), and various latent variable models (Chapter 28). When small enough, these are generally inherently interpretable. 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 a model picks only one of several correlated features, it may be harder to identify what signal is actually driving the prediction. A model might also assign correlated features different signs that ultimately cancel, rendering an interpretion of weights meaningless.

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]), generalized additive models (e.g., [HT17]) and monotonicity-enforced models (e.g., [Gup+16]).

Logic-based models use logical statements as basis. Models in this category include decision-trees [Bre+17], decision lists [Riv87; WR15; Let+15a; Ang+18; DMV15] , decision tables, decision sets [Hau+10; Wan+17a; LBL16; Mal+17; Bén+21], and logic programming [MDR94]. A broader discussion, as well as a survey of user studies on these methods, can be found in [Fre14]. Logic-based models easily model non-linear relationships but can have trouble modeling continuous relationships between the input and output (e.g., expressing a linear function vs. a step-wise constant function). Like the compact models, hierarchies and other forms of modularity can be used to extend the expressivity of the model while keeping it human-inspectable. For example, one can define a new concept as a formula based on some literals, and then use the new concept to build more complex rules.

When using inherently interpretable models, three key decisions need to be made: the choice of the model class, how to manage uninterpretable input features, and the choice of optimization method.

4. There may be other questions, such as how training data influenced the model, which may still require additional computation or information.

Decision: model class. Since the model is its own explanation, the decision on the model class becomes the decision on the form of explanation. Thus, we need to consider both whether the model class is a good choice for modeling the data as well as providing the necessary information to the user. For example, if one chooses to use a linear model to describe one's data, then it is important that the intended users can understand or manipulate the linear model. Moreover, if the fitting process produces a model that is too large to be human-inspectable, then it is no longer inherently interpretable, even if it belongs to one of the model classes described above.

Decision: optimization methods for training. The kinds of model classes that are typically inherently interpretable often require more advanced methods for optimization: compact, sparse, and logic-based models all involve learning discrete parameters. Fortunately, there is a long and continuing history of research for optimizing such models, including directly via optimization programs, relaxation and rounding techniques, and search-based approaches. Another popular optimization approach is via distillation or mimics: one first trains a complex model (e.g., a neural network) and then uses the complex model's output to train a simpler model to mimic the more complex model. The more complex model is then discarded. These optimization techniques are beyond the scope of this chapter but covered in optimization textbooks.

Decision: how to manage uninterpretable input features. Sometimes the input features themselves are not directly interpretable (e.g., pixels of an image or individual amplitudes spectrogram); only collections of inputs have semantic meaning for human users. This situation challenges our ability not only to create inherently interpretable models but also explanations in general.

To address this issue, more advanced methods attempt to add a "concept" layer that first converts the uninterpretable raw input to a set of human-interpretable concept features. Next, these concepts are mapped to the model's output [Kim+18a; Bau+17]. This second stage can still be inherently interpretable. For example, one could first map a pattern of spectrogram to a semantically meaningful sound (e.g., people chatting, cups clinking) and then from those sounds to a scene classification (e.g., in a 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. Creating and validating that machine-derived concepts correspond to a semantically meaningful human concepts remains 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. 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, waveforms, and language — in which deep models or other more complex models typically give significantly higher performance. Trying to fit complex behavior with a simple function may result not only in high bias in the trained model but also invite people to (incorrectly) rationalize why that highly biased model is sensible [Lun+20]. In an industry setting, seeking a migration away from a legacy, uninterpretable, business-critical model that has been tuned over decades would run into 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 33.1, interpretability is just one form of validation mechanism. For example, if the data distribution shifts, then one may observe unexpected model behavior.

33.2.2 Semi-inherently interpretable models: example-based methods

Example-based models use examples as their basis for their predictions. For example, an example-based classifier might predict the class of a new input by first identifying the outputs for similar instances in the training set and next taking a vote. K-nearest neighbors is one of the best known models in this class. Extensions include methods to identify exemplars for predicted classes and clusters (e.g., [KRS14; KL17b; JL15a; FD07b; RT16; Arn+10]), to generate exemplars (e.g., [Li+17d]), to define similarities between instances via sophisticated embeddings (e.g.,[PM18a]), and to first decompose an instance into parts and then find neighbors or exemplars between the parts (e.g., [Che+18b]). Like logic-based models, example-based models can describe highly non-linear boundaries.

On one hand, individual decisions made by example-based methods seem fully inspectable: one can provide the user with exactly the training instances (including their output labels) that were used to classify a particular input in a particular way. However, it may be difficult to convey a potentially complex distance metric used to define "similarity". As a result, the user may incorrectly infer what features or patterns made examples similar. It is also often difficult to convey the intuition behind the global decision boundary using examples.

33.2.3 Post-hoc or joint training: the explanation gives a partial view of the model

Inherently interpretable models are a subset of all machine learning models, and circumstances may require working with a model that is not inherently interpretable. As noted above, large neural models (Chapter 16) have demonstrated large performance benefits for certain kinds of data (e.g., images, waveform, and text); one might have to work with a legacy, business critical model that has been tuned for decades; one might be trying to understand a system of interconnected models.

In these cases, the view that the explanation gives into the model will necessarily be *partial*: the explanation may only be an approximation of the model or be otherwise incomplete. Thus, more decisions have to be made. Below, we split these decisions into two broad categories — what the explanation should consist of to best serve the context and how the explanation should be computed from the trained model. More detail on the abilities and limitations of these partial explanation methods can be found in [Sla+20; Yeh+19a; Kin+19; Ade+20a].

33.2.3.1 What does the explanation consist of?

One set of decisions center around the content of the explanation and what properties it should have. One choice is the form: Should the explanation be a list of important features? The top interactions? One must also choose the scope of the explanation: Is it trying to explain the whole model (global)?

The model's behavior near a specific input (local)? Something else? Determining what properties the explanation must have will help answer these and other questions. We expand on each of these points below; the right choice, as always, will depend on the user — whom the explanation is for —and their end-task.

Decision: form of the explanation. In the case of inherently interpretable models, the model class used to fit the data was also the explanation. Now, the model class and the explanation are two different entities. For example, the model could be a deep network and the explanation a decision tree.

Works in interpretable machine learning have used a large variety of forms of explanations. The form could be a list of "important" input features [RSG16b; Lun+20; STY17; Smi+17; FV17] or "important" concepts [Kim+18a; Bau+20; Bau+18]. Or it could be a simpler model that approximates the complex model (e.g., a local linear approximation, an approximating rule set)[FH17; BKB17; Aga+21b; Yin+19c]. Another choice could be a set of similar or prototypical examples [KRS14; AA18; Li+17d; 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].

Different forms of explanations will facilitate different tasks in different contexts. For example, a contrastive explanation of why treatment A is better than treatment B may help a clinician decide between treatments A and B. However, a contrast between treatments A and B may not be useful when comparing treatments A and C. Given the large number of choices, literature on how people communicate in the desired context can often provide some guidance. For example, if the domain involves making quick, high-stakes decisions, one might turn to how trauma nurses and firefighters explain their decisions (known as recognition-primed decision making, [Kle17]).

Decision: scope of the explanation: global or local. Another major decision regarding the parameters of the explanation is its scope.

Local explanation: In some cases, we may only need to interrogate an existing model about a specific decision. For example, why was this image predicted as a bird? Why was this patient predicted to have diabetes? Local explanations can help see if a consequential decision was made incorrectly or determine what could have been done differently to produce a different outcome (i.e., provide a recourse).

Local explanations can take many forms. A family of methods called saliency maps or attribution maps [STY17; Smi+17; ZF14; Sel+17; Erh+09; Spr+14; Shr+16] estimate the importances of each input dimension (e.g., via first-order derivatives with respect to the input). More generally, one might locally-fit simpler model in the neighborhood of the input of interest (e.g., LIME [RSG16b]). A local explanation may also consist of representative examples, including identifying which training points were most influential for a particular decision [KL17b] or identifying nearby datapoints with different predictions [MRW19; LHR20; Kar+20a].

All local explanation methods are partial views because they only attempt to explain the model around an input of interest. A key risk is that the user may overgeneralize the explanation to a wider region than it applies. They may also interpolate an incorrect mental model of the model based on a few local explanations.

Global explanation: In other cases, we may desire insight into the model as a whole or for a collection of datapoints (e.g., all inputs predicted to one class). For example, suppose that our end-task is to decide whether to deploy a model. Then, we care about understanding the *entire* model.

Global explanations can take many forms. One choice is to fit a simpler model (e.g., an inherently interpretable model) that approximates the original model (e.g., [HVD14]). One can also identify concepts or features that affect decisions across many inputs (e.g., [Kim+18b]). Another approach is to provide a carefully chosen set of representative examples [Yeh+18]. These examples might be chosen to be somehow characteristic of, or providing coverage of, a class (e.g., [AA18]), to draw attention to decision boundaries (e.g., [Zhu+18]), or to identify inputs particularly influential in training the model.

Unless a model is inherently interpretable, it is still important to remember that a global explanation is still a partial view. To make a complex model accessible to the user, the global explanation will need to leave some things out.

Decision: determining what properties the context needs. Different forms of explanations have different levels of expressivity. For example, an explanation listing important features, or fitting a local linear model around a particular input, does not expose interactions—but fitting a local decision tree would. For each form, there will also be many ways to compute an explanation of that form (more on this in Section 33.2.3.2). How do we choose amongst all of these different ways to compute the explanation? We suggest that the first step in determining the form and computation of an explanation should be to determine what properties are needed from it.

Specifying properties is especially important because not only may different forms of explanations have different intrinsic properties—e.g., can it model interactions?—but the properties may depend on the model being explained. For example, if the model is relatively smooth, then a feature-based explanation relying on local gradients may be fairly faithful to the original model. However if the model has spiky contours, the same explanation may not adequately capture the model's behavior. Once the desired properties are determined, one can determine what kind of computation is necessary to achieve them. We will list commonly desirable properties in Section 33.3.

33.2.3.2 How the explanation is computed

Another set of decisions has to do with how the explanation is computed.

Decision: computation of explanation. Once we make the decisions above, we must decide how the explanation will actually be computed. This choice will have a large impact on the explanation's properties. Thus, it is crucial to carefully choose a computational approach that provides the properties needed for the context and end-task.

For example, suppose one is seeking to identify the most "important" input features that change a prediction. Different computations correspond to different definitions of importance. One definition of importance might be the smallest region in an image that, when changed, changes the prediction—a perturbation-based analysis. Even within this definition, we would need to specify how that perturbation will be computed: Do we keep the pixel values within the training distribution? Do we preserve correlations between pixels? Different works take different approaches [SVZ13; DG17; FV17; DSZ16; Adl+18; Bac+15a].

A related approach is to define importance in terms of sensitivity (e.g., largest gradients of the output with respect to the input feature). Even then, there are many computational decisions to be made [STY17; Smi+17; Sel+17; Erh+09; Shr+16]. Yet another common definition of importance is how often the input feature is part of a "winning coalition" that drives the prediction, e.g., a Shapley or Banzaf score [LL17]. Each of these definitions have different properties, as well as require different amounts of computation.

Similar issues come up with other forms of explanations. For example, for an example-based explanation, one has to define what it means to be similar or otherwise representative: Is it the cosine similarity between activations? A uniform L2 ball of a certain size between inputs? Likewise, there are many different ways to obtain counterfactuals. One can rely on distance functions to identify nearby inputs that with different outputs [WMR17; LHR20], causal frameworks [Kus+18], or SAT formulations [Kar+20a], among other choices.

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. Note that this limitation is not because the partial explanation is wrong, but because the view that local gradients provide isn't sufficient if the true decision boundary is curvy.

One approach to getting explanations to have desired properties we 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 desired explanation [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, in some settings, we may desire the explanation use the same features that people do for the task (e.g., lower frequency vs. higher frequency features in image classifiers [Wan+20b]) — and still be faithful to the model. In other settings, we may want to control the input dimensions used or not used in the explanation, or for the explanation to be somehow compact (e.g., a small decision tree) while still being faithful to the underlying model, [RHDV17; Shu+19a; Vel+17; Nei+18; Wu+19b; Plu+20]. (Certain attention models fall into this category [JW19; WP19].) We may also have constraints on the properties of concepts or other intermediate features [AMJ18b; Koh+20a; Hen+16; BH20; CBR20; Don+17b]. In all of these cases, these desired properties could be included as a regularizer when training the model.

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 a prohibitively long time. 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 models are still 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 the model nor cover the model completely. As such, an explanation method tailored for one context may not be transferable in another; even in the intended context, there may be blindspots about the model that the explanation misses completely. For these reasons, in high stakes situations, one should attempt to use an inherently interpretable model first if possible [Rud19]. In all situations when post-hoc explanations are used, one must keep in mind that they are only one tool in a broader accountability toolkit and warn users appropriately.

33.2.4 Transparency and visualization

The scope of interpretable machine learning is around methods that expose the process by which a trained model makes a decision. However, the behavior of a model also depends on the objective function, the training data, how the training data were collected and processed, and how the model was trained and tested. Conveying to a human these other aspects of what goes into the creation of a model can be as important as explaining the trained model itself. While a full discussion of transparency and visualization is outside the scope of this chapter, we provide a brief discussion here to describe these important adjacent concepts.

Transparency is an umbrella term for the many things that one could expose about the modeling process and its context. Interpreting models is one aspect. However, one could also be transparent about other aspects, such as the data collection process or the training process (e.g., [Geb+21; Mit+19; Dnp]). There are also situations in which a trained model is released (whether or not it is inherently interpretable), and thus the software can be inspected and run directly.

Visualization is one way to create transparency. One can visualize the data directly or various aspects of the model's process (e.g., [Str+17]). Interactive visualizations can convey more than text or code descriptions [ZF14; OMS17; MOT15; Ngu+16; Hoh+20]. Finally, in the specific context of interpretable machine learning, how the explanation is presented — the visualization — can make a large difference in how easily users can consume it. Even something as simple as a rule list has many choices of layout, highlighting, and other organization.

When might we want to consider transparency and visualization? When not? In many cases, the trouble with a model comes not from the model itself, but parts of its training pipeline. The problem might be the training data. For example, since policing data contain historical bias, predictions of crime hot spots based on that data will be biased. Similarly, if clinicians only order tests when they are concerned about a patient's condition, then a model trained to predict risk based on tests ordered will only recapitulate what the clinicians already know. Transparency about the properties of the data, and how training and testing were performed, can help identify these issues.

Of course, inspecting the data and the model generation process is something that takes time and attention. Thus, visualizations of this kind and other descriptions to increase transparency are best-suited to situations in which a human inspector is not under time pressure to sift through potentially complex patterns for sources of trouble. These methods are not well-suited for situations in which a specific decision must be made in a relatively short amount of time, e.g., providing decision-support to a clinician at the bedside.

Finally, transparency in the form of making code available can potentially assist in understanding how a model works, identifying bugs, and allowing independent testing by a third party (e.g., testing with a new set of inputs, evaluating counterfactuals in different testing distributions). However, if a model is sufficiently complex, as many modern models are, then simply having access to the code is likely not be enough for a human to gain sufficient understanding for their task.

33.3 Properties: the abstraction between context and method

Recall from the terminology and framework in Section 33.1.2 that the context and end-task determine what properties are needed for the explanation. For example, in a high-stakes setting — such as advising on interventions for an unstable patient — it may be important that the explanation completely and accurately reflects the model (fidelity). In contrast, in a discovery-oriented setting, it might be more important for any explanation to allow for efficient iterative refinement, revealing different aspects of the model in turn (interactivity). Not all contexts and end-tasks need all properties, and the lack of a key property may result in poor downstream performance.

While the research is still evolving, there exists a growing informal understanding about how properties may work as an abstraction between methods and contexts. Many interpretability methods from Section 33.2 share the same properties, and methods with the same properties may have similar downstream performance in a specific end-task and context. If two contexts and end-tasks require the same properties, then a method that works well for one may work well for the other. A method with properties well-matched for one context could miserably fail in another context.

How to find desired properties? Of course, identifying what properties are important for a particular context and end-task is not trivial. Indeed, identifying what properties are important for what contexts, end-tasks, and downstream performance metrics is one facet of the grand challenge of interpretable machine learning. For the present, the process of identifying the correct properties will likely require iteration via user studies. However, iterating over properties is still a much smaller space than iterating over methods. For example, if one wants to test whether the sparsity of the explanation is key to good downstream performance, one could intentionally create explanations of varying levels of sparsity to test that hypothesis. This is a much more precise knob to test than exhaustively trying out different explanation methods with different hyperparameters.

Below, we first describe examples of properties that have been discussed in the interpretable machine learning literature. Many of these properties are purely computational — that is, they can be determined purely from the model and the explanation. A few have some user-centric elements. Next we list examples of properties of explanation from cognitive science (on human to human explanations) and human-computer interaction (on machine to human explanations). Some of these properties have analogs in the machine learning list, while others may serve as inspiration for areas to formalize.

33.3.1 Properties of explanations from interpretable machine learning

Many lists of potentially-important properties of interpretable machine learning models have been compiled, sometimes using different terms for similar concepts and sometimes using the similar terms for different concepts. Below we list some commonly-described properties of explanations, knowing that this list will evolve over time as the field advances.

Faithfulness, fidelity (e.g., as described in [JG20; JG21]). When the explanation is only a partial view of the model, how well does it match the model? There are many ways to make this notion precise. For example, suppose a mimic (simple model) is used to provide a global explanation of a more complex model. One possible measure of faithfulness could be how often the mimic gives the same outputs as the original. Another could be how often the mimic has the same first derivatives (local slope) as the original. In the context of a local explanation consisting of the 'key' features for a prediction, one could measure faithfulness by whether the prediction changes if the supposedly important features are flipped. Another measure could check to make sure the prediction does not change if a supposedly unimportant feature is flipped. The appropriate formalization will depend on the context.

Compactness, sparsity (e.g., as described in [Lip18; Mur+19]). In general, an explanation must be small enough such that the user can process it within the constraints of the task (e.g., how quickly a decision must be made). Sparsity generally corresponds to some notion of smallness (a few features, a few parameters, $L1$ norm etc.). Compactness generally carries an additional notion of not including anything irrelevant (that is, even if the explanation is small enough, it could be made smaller). Each must be formalized for the context

Completeness (e.g., as described in [Yeh+19b]). If the explanation is not the model, does it still include all of the relevant elements? For example, if an explanation consists of important features for a prediction, does it include all of them, or leave some out? Moreover, if the explanation uses derived quantities that are not the raw input features — for example, some notion of higher-level concepts — are they expressive enough to explain all possible directions of variation that could change the prediction? Note that one can have a faithful explanation in certain ways but not complete in others: Fore example, an explanation may be faithful in the sense that flipping features considered important flips the prediction and flipping features considered unimportant does not. However, the explanation may fail to include that flipping a set of unimportant features does change the prediction.

Stability (e.g., as described in [AMJ18a]) To what extent are the explanations similar for similar inputs? Note that the underlying model will naturally affect whether the explanation can be stable. For example, if the underlying model has high curvature and the explanation has limited expressiveness, then it may not be possible to have a stable explanation.

Actionability (e.g., as described in [Kar+20b; Poy+20]). Actionability implies filtering the content of the explanation to focus on only aspects of the model that the user might be able to intervene on. For example, if a patient is predicted to be at high risk of heart disease, an actionable explanation might only include mutable factors such as exercise and not immutable factors such as age or genetics. The notion of recourse corresponds to actionability in a justice context.

Modularity (e.g., as described in [Lip18; Mur+19]). Modularity implies that the explanation can be broken down into understandable parts. While modularity does not guarantee that the user can explain the system as a whole, for more complex models, modular explanations — where the user can inspect each part — can be an effective way to provide a reasonable level of insight into the model's workings.

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 their own way.

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 to a deep model at a particular input, 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. As with all accountability methods, the goal of the explanation 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 take the model and an input and compute the output (within any constraints of time and cognition). A simulable explanation is an explanation that is a simulable model. For example, a list of features is not simulable, because a list of features alone does not tell us how to compute the output. In contrast, an explanation in the form of a decision tree does include a computation process: the user can follow the logic of the tree, as long as it is not too deep. This example also points 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 other properties because its definition involves human input. However, in practice, we 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 be turned into a purely computational training constraint where we seek simulatable explanations.

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's vocabulary? For example, the explanation could be given in the semantics a user knows, such as medical conditions vs. raw sensor readings. Doing so can help the user more easily connect the explanation to their knowledge and existing decision-making guidelines [Clo+19]. Of course, the right vocabulary will depend on the user: an explanation in terms of parameter variances and influential points may be comprehensible to an engineer debugging a lending model but not to a loan applicant.

Like simulability, mental-model alignment is more human-centric. However, just as before, we can imagine an abstraction between eliciting vocabulary and mental models from users (i.e., 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 desired property, would using the L1 norm be enough? Or does a more sophisticated loss term need to be designed? This decision will necessarily be human-centric: how small an explanation needs to be, or in what ways it needs to be small, is a decision that needs to consider how people will be using the explanation. Once operationalized, most properties can be optimized computationally. That said, the properties should be evaluated with the context, end-task, model, and chosen explanation methods. Once evaluated, one may revisit the choice of the explanation and model.

Finally, we emphasize that the ability to achieve a particular property will depend on the ***intrinsic*** characteristics of the model. For example, the behavior of a highly nonlinear model with interactions between the inputs will, in general, be harder to understand than a linear model. No matter how we try to explain it, if we are trying to explain something complicated, then users will have a harder time understanding it.

33.3.2 Properties of explanations from cognitive science

Above we focused on computational properties between models and explanations. The fields of cognitive science and human-computer interaction have long examined what people consider good properties of an explanation. These more human-centered properties may be ones that researchers in machine learning may be less aware of, yet essential for communicating information to people.

Unsurprisingly, the literature on human explanation concurs that the explanation must fit the context [VF+80]; different contexts require different properties and different explanations. That said, human explanations are also social constructs, often including post-hoc rationalizations and other biases. We should focus on properties that help users achieve their goals, not ones simply "because people sometimes do it".

Below we list several of these properties.

Soundness (e.g., as described in [Kul+13]). Explanations should contain nothing but the truth with respect to whatever they are describing. Soundness corresponds to notions of *compactness* and *faithfulness* above.

Completeness (e.g., as described in [Kul+13]). Explanations should contain the whole truth with respect to whatever they are describing. Completeness corresponds to notions of *completeness* and *faithfulness* above.

Generality (e.g., as described in [Mil19]). Overall, people understand that an explanation for one context may not apply in another. That said, there is an expectation that an explanation should reflect some underlying mechanism or principle and will thus apply to similar cases — for whatever notion of similarity is in the person's mental model. Explanations that do not generalize to similar cases may be misinterpreted. Generality corresponds to notions of *stability* above.

Simplicity (e.g., as described in [Mil19]). All of the above being equal, simpler explanations are generally preferred. Simplicity relates to notions of *sparsity* and *complexity* above.

Contrastiveness (e.g., as described in [Mil19]). Contrastive explanations provide information of how something differs from an alternate decision or prediction. For example, instead of providing a list of features for why a particular drug is recommended, it might provide a list of features that explain why one drug is recommended over another. Contrastiveness relates to notions of *actionability* above, and more generally explanation types that include *counterfactuals*.

Finally, the cognitive science literature also notes that explanations are often goal directed. This matches the notion of explanation in ML as information that helps a person improve performance on their end-task. Different information may help with different goals, and thus human explanations take many forms. Examples include deductive-nomological forms (i.e. a logical proofs) [HO48], forms that provide a sense of an underlying mechanism [BA05; Gle02; CO06], and forms that conveying understanding [Kei06]. Knowing these forms can help us consider what options might be best among different sets of interpretable machine learning methods.

33.4 Evaluation of interpretable machine learning models

One cannot formalize the notion of interpretability without specifying the context, the end-task, and the downstream performance metric [VF+80]. If one explanation empowers the human to get better performance on their end-task over another explanation, then it is more useful. While the grand challenge of interpretable machine learning is to develop a general understanding of what properties are needed for good downstream performance on different end-tasks in different contexts, in this section, we will focus on rigorous evaluation within one context [DVK17].

Specifically, we describe two major categories for evaluating interpretable machine learning methods:

Computational evaluations of properties (without people). Computational evaluations of whether explanations have desired properties do not user studies. For example, one can computationally measure whether a particular explanation satisfies a definition of faithfulness under different training and test data distributions or whether the outputs of one explanation are more sparse than another. Such measures are valuable when one already knows that certain properties may be important for certain contexts. Computational evaluations also serve as intermediate evaluations and sanity checks to identify undesirable explanation behavior prior to a more expensive user study-based evaluation.

User studies (with people). Ultimately, user studies are needed to measure how well an interpretable machine learning method enables the user to complete their end-task in a given context. Performing a rigorous, well-designed user study in a real context is significant work — much more so than computing a test likelihood on benchmark datasets. It requires significant asks of not only the researchers but also the target users. Methods for different contexts will also have different evaluation challenges: while a system designed to assist with optimizing music recommendations might be testable on a wide population, a system designed to help a particle physicist identify new kinds of interactions might only be tested with one or two physicists because people with that expertise are hard to find. In all cases, the evaluation can be done rigorously given careful attention to experimental design.

33.4.1 Computational evaluation: does the method have desired properties?

While the ultimate measure of interpretability is whether the method successfully empowers the user to perform their task, properties can serve as a valuable abstraction. Checking whether an explanation has the right computational and desired properties can ensure that the method works as expected (e.g., no implementation errors, no obviously odd behaviors). One can iterate on novel, computationally-efficient methods to optimize the quantitative formalization of a property before conducting expensive human experiments. Computational checks can also ensure whether properties that held for one model continue to hold when applied to another model. Finally, checking for specific properties can also help pinpoint in what way an explanation is falling short, which may be less clear from a user study due to confounding.

In some cases, one might be able to prove mathematically that an explanation has certain properties, while in others the test must be empirical. For empirical testing, one umbrella strategy is to use a hypothesis-based sanity check; if we think a phenomenon X should never occur (hypothesis), we can test whether we can create situations where X may occur. If it does, then the method fails this sanity check. Another umbrella strategy is to create datasets with known characteristics or ground truth explanations. These could be purely synthetic constructions (e.g., generated tables with intentionally correlated features), semi-synthetic approaches (e.g., intentionally changing the labels on an image dataset), or taking slices of a real dataset (e.g., introduce intentional bias by only selecting real image, label pairs that are of outdoor environments). Note that these tests

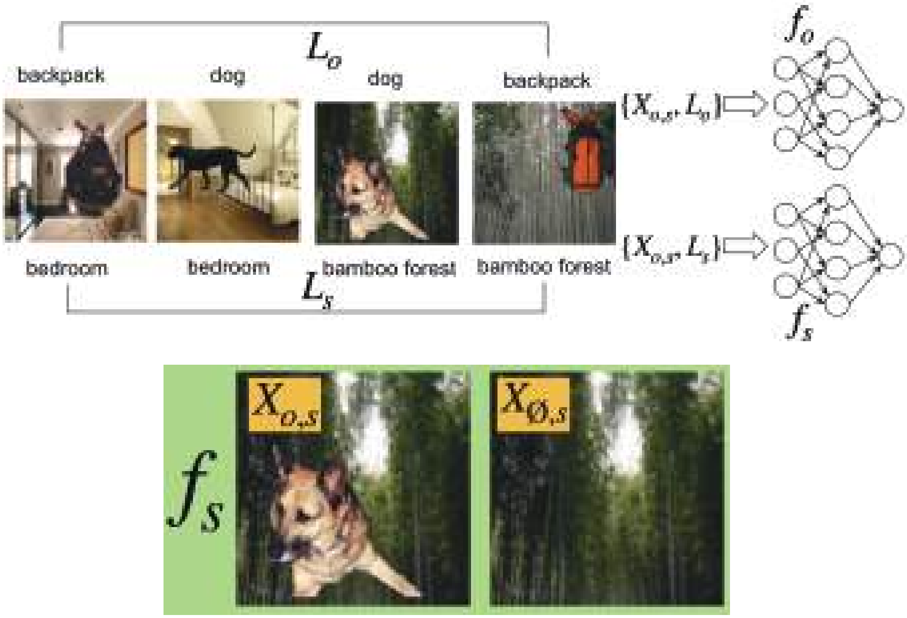


Figure 33.3: An example of computational evaluation using (semi-)synthetic datasets from [YK19]: foreground images (e.g., dogs, backpacks) are placed on different backgrounds (e.g., indoors, outdoors) to test what an explanation is looking for.

can only tell us if something is wrong; if a method passes a check, there may still be missing blindspots.

Examples of sanity checks. One strategy is to come up statements of the form "if this explanation is working, then phenomenon X should not be occurring" and then try to create a situation in which phenomenon X occurs. If we succeed, then the sanity check fails. By asking about out-of-the-box phenomena, this strategy can reveal some surprising failure modes of explanation methods.

For example, [Ade+20a] operates under the assumption that a faithful explanation should be a 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 33.4).

In another example, [Kin+19] hypothesize that a faithful explanation should be 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. Their work shows that some methods fail this sanity check.

Adversarial attacks on explanations also fall into this category. For example, [GAZ19] shows that two perceptively indistinguishable inputs with the same predicted label can be assigned very different explanations.

Examples using (semi)synthetic datasets. Constructed datasets can also help score properties

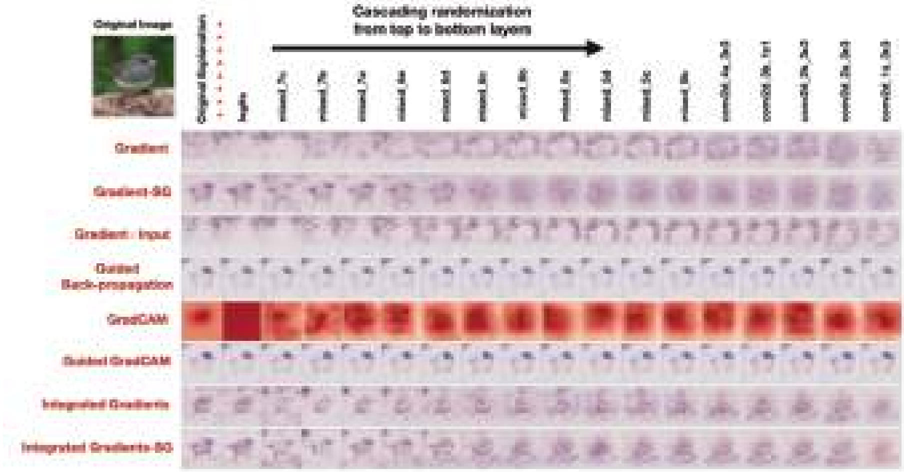


Figure 33.4: Interpretability methods (each row) and their explanations as we randomize layers starting from the logits, and cumulatively to the bottom layer (each column), in the context of image classification task. The rightmost column is showing a completely randomized network. Most methods output similar explanations for the first two columns; one predicts the bird, and the other predicts randomly. 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].

of various 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. To test for these properties, the authors generate images with known correlations. Specifically, they generate multiple datasets, each with a different rate of how often each particular foreground object co-occurs with each particular background (see Figure 33.3). Each dataset comes with two labels per image: for the object and the background.

Now, the authors compare two models: one trained to classify objects and one trained to classify backgrounds (left, Figure 33.3). 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 ([YK19] call this 'model contrast score'). They also checked that the model trained to predict backgrounds was not providing attributions to the foreground objects (see right Figure 33.3). Other works using similar strategies include [Wil+20b; Gha+21; PMT18; KPT21; Yeh+19b; Kim+18b].

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 an explanation ranks features from most to least important. We want to determine if this ranking is faithful. Further, suppose we can assume that the features do not interact. Then, one can attempt to make the prediction just with the top-1 most important feature, just the top-2 ranked features, etc. and observe if the change in prediction accuracy exhibits diminishing returns as more features are added. (If the features do interact, this test will not work. For example,

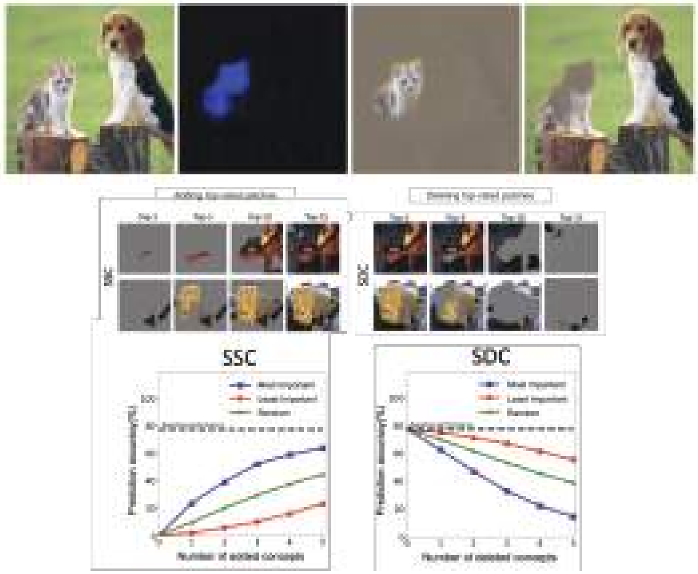


Figure 33.5: Examples of computational evaluation with real datasets. Top row is from Figure 1 of [DG17], used with kind permission of Yarin Gal. Bottom row is from Figure 4 of [Gho+19]. One would expect that adding or deleting patches rated as most 'relevant' for an image classification would have a large effect on the classification compared to patches not rated as important.

if features A, B, C are the top-3 features, but C is only important if feature B is present, the test above would over-estimate the importance of the feature C.)

Figure 33.5 shows an example of this kind of test [Gho+19]. Their method outputs a set of image patches (e.g., a set of connected pixels) that correlates with the prediction. They add top-n image patches provided by the explanation one by one and observe the desired trend in accuracy. A similar experiment in reverse direction (i.e., deleting top-n most important image patches one by one) provides additional evidence. Similar experiments are also conducted in [FV17; RSG16a].

For example, in [DG17], authors define properties in plain English first: smallest sufficient region (smallest region of the image that alone allows a confident classification) and smallest destroying region (smallest region of the image that when removed, prevents a confident classification), followed by careful operationalization of these properties such that they become the objective for optimization. Then, separately, an evaluation metric of saliency is defined to be "the tightest rectangular crop that contains the entire salient region and to feed that rectangular region to the classifier to directly verify whether it is able to recognise the requested class". While the "rectangular" constraint may introduce artifacts, it is a neat trick to make evaluation possible. By checking expected behavior as described above, authors confirm that methods' behavior on the real data is aligned with the defined property compared to baselines.

Evaluating the evaluations. As we have seen so far, there are many ways to formalize a given property and many empirical tests to determine whether a property is present. Each empirical test will have different qualities. As an illustration, in [Tom+20], the authors ask whether popular saliency metrics give consistent results across literature. They tested whether different metrics for assessing the quality of saliency-based explanations (explanations that identify important pixels or regions in images) is evaluating similar properties. In other words, this work tests consistency and stability properties of *metrics*. They show many metrics are statistically unreliable and inconsistent. While each metric may still have a particular use [Say+19], knowing this inconsistency exists helps us better understand the landscape and limitations of evaluation approaches. Developing good evaluations for computational properties is an ongoing area of research.

33.4.2 User study-based evaluation: does the method help a user perform a target task?

User study-based evaluations measure whether an interpretable machine learning method helps a human perform some task. This task could be the ultimate end-task of interest (e.g., does a method help a doctor make better treatment decisions) or a synthetic task that mirrors contexts of interest (e.g., a simplified situation with artificial diseases and symptoms). In both cases, rigorous experimental design is critical to ensuring that the experiment measures what we want it to measure. Understanding experimental design for user studies is essential for research in interpretable machine learning.

33.4.2.1 User studies in real contexts.

The gold standard for testing whether an explanation is useful is to test it in the intended context: Do clinicians make better decisions with a certain kind of decision support? Do programmers debug code faster with a certain kind of explanation about model errors? Do product teams create more fair models for their businesses? A complete guide on how to design and conduct user studies is out of scope for this chapter; below we point out some basic considerations.

33.4.2.2 Basic elements of user studies

Performing a high-quality user study is a nuanced and non-trivial endeavor. There are many sources of bias, some obvious (e.g., learning and fatigue effects during a study) and some less obvious (e.g., participants willing to work with us are more optimistic about AI technology than those we could not recruit, or different participants may have different needs for cognition).

Interface design. The explanation must be presented to the user. Unlike the *intrinsic* difficulty of explaining a model (i.e., complex models are harder to explain than simple ones), the design of the interface is an *extrinsic* source of difficulty that can confound the experimental results. For example, it may be easier, in general, to scan a list of features ordered by importance rather than alphabetically.

When we perform an evaluation with respect to an end-task, intrinsic and extrinsic difficulties can get conflated. Does one explanation type work better because it does a better job of explaining the complex system? Or does it work better simply because it was presented in a way that was easier for people to use? Especially if the goal is to test the difference between one explanation and another in the experiment, it is important that the interface for each is designed to tease out the effect from the explanations and their presentations. (Note that good presentations and visualization are an important but different object of study.) Moreover, if using the interface requires training, it is important to deliver the training to users in a way that is neutral in each testing condition. For example, how the end-task and goals of the study are described during training (e.g., with practice questions) will have a large impact on how users approach the task.

Baselines. Simply the presence of an explanation may change the way in which people interact with an ML system. Thus, it is often important to consider how a human performs with no ML system, with an ML system and no explanation, with an ML system and a placebo explanation (an explanation that provides no information), and with an ML system and hand-crafted explanations (manually generated by humans who are presumably good communicators).

Experimental design and hypothesis testing. Independent and dependent variables, hypotheses, and inclusion and exclusion criteria must be clearly defined prior to the start of the study. For example, suppose that one hypothesizes that a particular explanation will help a developer debug an image classifier. In this case, the independent variable would be a form of assistance: the particular explanation, competing explanation methods, and the baselines above. The dependent variable would be whether the developer can identify bugs. Inclusion and exclusion criteria might include a requirement that the developer has sufficient experience training image classifiers (as determined by an initial survey, or a pre-test), demonstrates engagement (as measured by a base level of performance on practice rounds), and does not have prior experience with the particular explanation types (as determined by an initial survey). Other exclusion criteria could be removing outliers. For example, one could decide, in advance, to exclude data from any participant that takes an unusually long or short time to perform task as a proxy for engagement.

As noted in Section 33.2, there are many decisions that go into any interpretable machine learning method, and each context is nuanced. Studies of the form "Does explanation X (computed via some pipeline Y) help users in context Z compared to explanation X'?" may not provide much insight as to *why* that particular explanation is better or worse — making it harder not only to iterate on a particular explanation but also to generalize to other explanations or contexts. There are many factors of potential variation in the results, ranging from the properties of the explanation and its presentation to the difficulty of the task.

To reduce this variance, and to get more useful and generalizable insights, we can manipulate some factors of variation directly. For example, suppose the research question is whether complete explanations are better than incomplete explanations in a particular context. One might write out hand-crafted explanations that are complete in what features they implicate, explanations in which one important feature is missing, and explanations in which several important features are missing. Doing so ensures even coverage of the different experimental regimes of interest, which may not occur if the explanations were simply output from a pipeline. As another example, one might intentionally create an image classifier with known bugs, or simply pretend to have an image classifier that makes certain predictions (as done in [Ade+20b]). These kinds of studies are called *wizard-of-Oz* studies, and they can help us more precisely uncover the science of why an explanation is useful (e.g., as done in [Jac+21]).

Once the independent and dependent variables, hypotheses, and participant criteria (including how the independent and dependent variables may be manipulated) are determined, the next step is setting up the study design itself. Broadly speaking, *randomization* marginalizes over potential confounds. For example, randomization in assigning subjects to tasks marginalizes the subject's prior knowledge; randomization in the order of tasks marginalizes out learning effects. *Matching* and *repeated measures* reduce variance. An example of matching would be asking the same subject to perform the same end-task with two different explanations. An example of repeated measures would be asking the subject to perform the end-task for several different inputs.

Other techniques for designing user studies include block randomized designs/Latin square designs that randomize the order of explanation types while keeping tasks associated with each explanation type grouped together. This can be used to marginalize the effects of learning and fatigue without too much context switching. Careful consideration should be given to what will be compared within subjects and across subjects. Comparisons of task performance within subjects will have lower variance but a potential bias from learning effects from the first task to the second. Comparisons across subjects will have higher variance and also potential bias from population shift during experimental recruitment. Finally, each of these study designs, as well as the choice of independent and dependent variables, will imply an appropriate significance test. It is essential to choose the right test and multiple hypothesis correction to avoid inflated significance values while retaining power.

Qualitative studies. So far, we have described the standard approach for the design of a quantitative user study–one in which the dependent variable is numerically measured (e.g., time taken to correctly identify a bug, % bugs detected). While quantitative studies provide value by demonstrating that there is a consistent, quantifiable effect across many users, they usually do not tell us *why* a certain explanation worked. In contrast, qualitative studies, often performed with a "think-aloud" or other discussion-based protocol in which users expose their thought process as they perform the experiment, can help identify why a particular form of explanation seems to be useful or not. The experimenter can gain insights by hearing how the user was using the information, and depending on the protocol, can ask for clarifications.

For example, suppose one is interested in how people use an example-based explanation to understand a video-game agent's policy. The idea is to show a few video clips of an automated agent in the video game, and then ask the user what the agent might do in novel situations. In a think-aloud study, the user would perform this task while talking through how they are connecting the videos they have seen to the new situation. By hearing these thoughts, a researcher might not only gain deeper insight into how users make these connections — e.g., users might see the agent collect coins in one video and presume that the agent will always go after coins — but they might also identify surprising bugs: for example, a user might see the agent fall into a pit and attribute it to a one-off sloppy fingers, not internalizing that an automated agent might make that mistake every time.

While a participant in a think-aloud study is typically more engaged in the study than they might be otherwise (because they are describing their thinking), knowing their thoughts can provide insight into the causal process between what information is being provided by the explanation and the action that the human user takes, ultimately helping advance the science of how people interact with machine-provided information.

Pilot studies: The above descriptions are just a very high-level overview of the many factors that must be designed properly for a high-quality evaluation. In practice, one does not typically get all of these right the first time. Small scale pilot studies are essential to checking factors such as whether participants attend to the provided information in unexpected ways or whether instructions are clear and well-designed. Modifying the experiments after iterative small scale pilot studies can save a lot of time and energy down the road. In these pilots, one should collect not only the usual information about users and the dependent variables, but also discuss with the participants how they approached the study tasks and whether any aspects of the study were confusing. These discussions will lead to insights and confidence that the study is testing what it is intended to test. The results from pilot studies should not be included in the final results.

Finally, as the number of factors to test increases (e.g., baselines, independent variables), the study design becomes more complex and may require more participants and longer participation times to determine if the results are significant — which can in turn increase costs and effects of fatigue. Pilots, think-aloud studies, and careful thinking about what aspects of the evaluation require user studies and what can be completed computationally can all help distill down a user-based evaluation to the most important factors.

33.4.2.3 User studies in synthetic contexts

It is not always appropriate or possible to test an interpretable machine learning method in the real context: for example, it would be unethical to test a prototype explanation system on patients each time one has a new way to convey information about a treatment recommendation. In such cases, we might want to run an experiment in which clinicians perform a task on made-up patients, or in some analogous non-medical context where the participant pool is bigger and more affordable. Similarly, one might create a relatively accessible image classification debugging context where one can control the incorrect labels, distribution shifts, etc. (e.g., [Ade+20b]) and see what explanations help users detect problems in this simpler setting. The convenience and scalability of using a simpler setting could shed light on what properties of explanations are important generally (e.g., for debugging image classification). For example, we can test how different forms of explanation have different cognitive loads or how a particular property affects performance with a relatively large pool of subjects (e.g., [Lag+19]). The same principles we outlined above for user studies in real contexts continue to apply, but there are some important cautions.

Cautions regarding synthetic contexts: While user studies with synthetic contexts can be valuable for identifying scientific principles, one must be cautious. For example, experimental subjects in a synthetic high-stakes context may not treat the stakes of the problem as seriously, may be relatively unburdened with respect to distractions or other demands on their time and attention (e.g., a quiet study environment vs. a chaotic hospital floor), and ignore important factors of the task (e.g., clicking through to complete the task as quickly as possible). Moreover, small differences in task definition can have big effects: even the difference between asking users to simply perform a task with an explanation available vs. asking users to answer some questions about the explanation first, may create very different results as the latter forces the user to pay attention to the explanation and the former does not. Priming users by giving them a specific scenario where they can put themselves into a mindset could help. For example: "Imagine now you are an engineer at a company selling a risk calculator. A deadline is approaching and your boss wants to make sure the product will work for a new client. Describe how you would use the following explanation".

33.5 Discussion: how to think about interpretable machine learning

Interpretable machine learning is a young, interdisciplinary field of study. As a result, consensus on definitions, evaluation methods, and appropriate abstractions is still forming. The goal of this section is to lay out a core set of principles about interpretable machine learning. While specifics in the previous sections may change, the principles below will be durable.

There is no universal, mathematical definition of interpretability, and there never will be. Defining a downstream performance metric (and justifying it) for each context is a must. The information that best communicates to the human what is needed to perform a task will necessarily vary: for example, what a clinical expert needs to determine whether to try a new treatment policy is very different than what a person to determine how to get a denied loan approved. Similarly, methods to communicate characteristics of models built on pixel data may not be appropriate for communicating characteristics of models built on language data. We may hope to identify desired properties in explanations to maximize downstream task performance for different classes of end tasks — that is the grand challenge of interpretable machine learning — but there will never be one metric for all contexts.

While this lack of a universal metric may feel disappointing, other areas of machine learning also lack universal metrics. For example, not only is it impossible to satisfy the many metrics on fairness at the same time [KMR16], but also in a particular situation, none may exactly match the desires of the stakeholders. Even in a standard classification setting, there are many metrics that correspond to making the predicted and true labels as close as possible. Does one care about overall accuracy? Precision? Recall? 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 of computation and statistics but also experimental design and user studies.

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* towards 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 a model satisfy certain fairness metrics), we do not need any human input. If a model behaves as expected 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 work in the identification of causal models (see Chapter 36), under many circumstances, it is not possible to learn a model that is *guaranteed* to be causal from a dataset alone. Here, interpretability could assist the end-task of "Is the model causal?" by allowing a human to inspect the model's prediction process.

As another example, one could measure the safety of a clinical decision support system by tracking how often its recommendations causes harm to patients — and stop using the system if it causes 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 inspect the model's decision process for red flags *prior* to deployment.

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.

Interpretability is not about understanding everything about the model; it is about understanding enough to do the end-task. The ultimate measure of an interpretable machine learning method is whether it helps the user perform their end-task. Suppose the end-task is to fix an overheating laptop. An explanation that lists the likely sources of heat is probably sufficient to address the issue, even if one does not know the chemical properties of its components. On the other hand, if the laptop keeps freezing up, knowing about the sources of heat may not be the right information. Importantly, both end-tasks have clear downstream performance metrics: we can observe whether the information helped the user perform actions that make the laptop overheat or freeze up less.

As another example, consider AlphaGo, Google DeepMind's AI go player that beat the human world champion, Lee SeDol. The model is so complex that one cannot fully understand its decision process, including surprising moves like its famous move 37[Met16]. That said, partial probes (e.g., does AlphaGo believe the same move would have made a different impact if it was made earlier but similar position in the game) might still help a go expert gain insights on the rationale for the move in the context of what they already know about the game.

Relatedly, 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 may or may not be sufficient for a user to accomplish their end-task. For example, a typical user will not be able to reason through $100K$ lines of parameters despite having all the pieces available.

That said, any partial view of a model is, necessarily, only a partial view; it does not tell the full story. While we just argued that many end-tasks do not require knowing everything about a model, we also must acknowledge that a partial view does not convey the full model. For example, the set of features needed to change a loan decision may be the right partial view for a denied applicant, but convey nothing about whether the model is discriminatory. Any probe will only return what it is designed to compute (e.g., an approximation of a complex function with a simpler one). Different probes may be able to reveal different properties at different levels of quality. Incorrectly believing the partial view is the full story could result in incorrect insights.

Partial views can lack stability and enable attacks. Relatedly, any explanation that reveals only certain parts of a model can lack stability (e.g., [AMJ18a]) and can be more easily attacked (e.g., see [Yeh+19a; GAZ19; Dom+19; Sla+20]). Especially when models are overparameterized such as neural networks, it is possible to learn models whose explanations say one thing (e.g., a feature is not important, according to some formalization of feature importance) while the model does another (e.g., uses the prohibited feature). Joint training can also exacerbate the issue, as it allows the model to learn boundaries that pass some partial-view test while in reality violating the underlying constraint. Other adversarial approaches can work on the input, minimally perturbing it to change the explanation's partial view while keeping the prediction constant or to change the prediction while keeping the explanation constant.

These concerns highlight an important open area: We need to improve ways to endow explanations with the property of translucence, that is, explanations that communicate what they can and cannot say about the model. Translucence is important because misinterpreted explanations that happen to favor a user's views create false basis for trust.

Trade-offs between inherently interpretable models and performance often do not exist; partial views can help when they do. While some have claimed that there exists an inherent trade-off between using an inherently-interpretable model and performance (defined as a model's performance on some test data), this trade-off does not always exist in practice for several reasons [Rud19].

First, in many cases, the data can be surprisingly well-fit by a fairly simple model (due to high noise, for example) or a model that can be decomposed into interpretable parts. One can often find a combination of architecture, regularizer, and optimizer that produces inherently interpretable models with performance comparable to, or sometimes even better than, blackbox approaches [Wan+17a; LCG12; Car+15; Let+15b; UR16; FHDV20; KRS14]. In fact, interpretability and performance can be synergistic: methods for encoding a preference for simpler models (e.g., L1 regularizer for sparsity property) were initially developed to increase performance and avoid overfitting, and interpretable models are often more robust [RDV18].

Second, a narrow focus on the trade-off between using inherently interpretable models and a predefined metric of performance, as usually measured on a validation set, overlooks a broader issue: that predefined metric of performance may not tell the full story about the quality of the model. For example, using an inherently interpretable model may enable a person to realize that a prediction is based on confounding, not causation—or other ways it might fail in deployment. In this way, one might get better performance with an inherently interpretable model in practice even if a blackbox appears to have better performance numbers in validation. An inherently interpretable model may also enable better human+model teaming by allowing the human user to step in and override the system appropriately.

Human factors are essential. All machine learning systems ultimately connect to broader socio-technical contexts. However, in many cases, many aspects of model construction and optimization can be performed in a purely computational setting: there are techniques to check for appropriate model capacity, techniques for tuning a gradient descent or convex optimization. In contrast, interpretable machine learning must consider human factors *from the beginning*: there is no point optimizing an explanation to have various properties if it still fails to improve the user's performance on the end-task.

Over-reliance. Just because an explanation is present, does not mean that the user will analytically and reasonably incorporate the information provided into their ultimate decision-making task. The presence of *any* explanation can increase a user's trust in the model, exacerbating the general issue

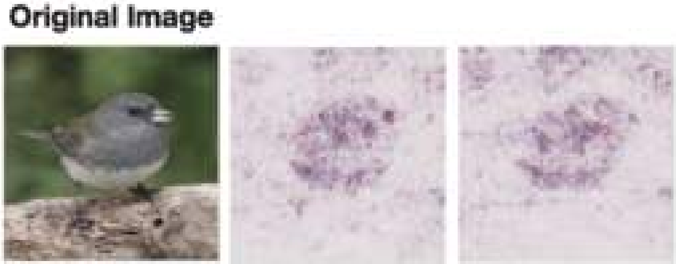


Figure 33.6: (Potential) perception issues: an explanation from a trained network (left) is visually indistinguishable to humans from one from an untrained network (right)—even if they are not exactly identical.

of over-trust in human+ML teams. Recent studies have found that even data scientists over-trust explanations in unintended ways [Kau+20]; their excitement about the tool led them to take it at face-value rather than dig deeper. [LM20] reports a similar finding, noting that inaccurate but evocative presentations can create a feeling of comprehension.

Over-reliance can be combated with explicit measures to force the user to engage analytically and skeptically with the information in the explanation. For example, one could ask the user to submit their decision first and only then show the recommendation and accompanying explanation to pique their interest in why their choice and the recommendation might disagree (and prompting whether they want to change their choice). Another option is to ask the user some basic questions about the explanation prior to submitting their decision to force them to look at the explanation carefully. Yet another option is to provide only the relevant information (the explanation) without the recommendation, forcing the user to synthesize the additional information on their own. However, in all these cases, there is a delicate balance: users will often be amenable to expending additional cognitive effort if they can see it achieves better results, but if they feel the effort is too much, they may start ignoring the information entirely.

Potential for misuse. A malicious version of over-reliance is when explanations are used to manipulate a user rather than facilitating the user's end-task. Further, users may report that they *like* explanations that are simple, require little cognitive effort, etc. even when those explanations do not help them perform their end-task. As creators of interpretable machine learning methods, one must be on alert to ensure that the explanations help the user achieve what they want to (ideally in a way that they also like).

Misunderstandings from a lack of understanding of machine learning. Even when correctly engaged, users in different contexts will have different levels of knowledge about machine learning. For example, not everyone may understand concepts such as additive factors or Shapley values [Sha16]. Users may also attribute more understanding to a model than it actually has. For example, if they see a set of pixels highlighted around a beak, or a set of topic model terms about a disease, they may mistakenly believe that the machine learning model has some notion of concepts that matches theirs, when the truth might be quite different.

Related: perception issues in image explanations. The nature of our visual processing system adds another layer of nuance when it comes to interpreting and misinterpreting explanations. In Figure 33.6, two explanations (in terms of important pixels in a bird image) seem to communicate a similar message; for most people, both explanations seem to suggest that the belly and cheek of the bird are the important parts for this prediction. However, one of them is generated from a trained network (left), but the other one is from a network that returns random predictions (right). While the two saliency maps aren't identical to machines, they look similar because humans don't parse an image as pixel values, but as whole, they see a bird in both pictures.

Another common issue with pixel-based explanations is that explanation creators often multiply the original image with an importance "mask" (black and clear saliency mask, where black pixel represents no importance and a clear pixel represents maximum importance), introducing the arbitrary artifact that black objects never appear important [Smi+17]. In addition, this binary mask is produced by clipping important pixels in a certain percentile (e.g., only taking 99−th percentile), which can also introduce another artifact [Sun+19c]. The balancing act between artifacts introduced by visualization for the ease of understanding and faithfully representing the explanation remains a challenge.

Together, all of these points on human factors emphasize what we said from the start: we cannot divorce the study and practice of interpretable machine learning from its intended socio-technical context.

Part VI

Action

34 Decision making under uncertainty

34.1 Statistical decision theory

Bayesian inference provides the optimal way to update our beliefs about hidden quantities H given observed data $\mathbf{X} = \boldsymbol{x}$ by computing the posterior $p(H|\boldsymbol{x})$. However, at the end of the day, we need to turn our beliefs into **actions** that we can perform in the world. How can we decide which action is best? This is where **decision theory** comes in. In this section, we give a brief introduction. For more details, see e.g., [DeG70; Ber85b; KWW22].

34.1.1 Basics

In **statistical decision theory**, we have an **agent** or decision maker, who wants to choose an **action** from has a set of possible actions, $a \in \mathcal{A}$, given some observations or data $\boldsymbol{x}$. We assume the data comes from some environment that is external to the agent; we characterize the state of this environment by a hidden or unknown variable $h \in \mathcal{H}$, known as the **state of nature**. Finally, we assume we know a **loss function** $\ell(h, a)$, that specifies the loss we incur if we take action a when the state of nature is h. The goal is to define a **policy**, also called an **estimator** or **decision procedure**, which specifies which action to take in response to each possible observation, $a = \delta(\boldsymbol{x})$, so as to minimize the expected loss, also called the **risk**:

$$\delta^*(\cdot) = \operatorname*{argmin}_{\delta} R(\delta) \tag{34.1}$$

where the risk is given by

$$R(\delta) = \mathbb{E}\left[\ell(h, \delta(\mathbf{X}))\right] \tag{34.2}$$

The key question is how to define the above expectation. We can use a frequentist or Bayesian approach, as we discuss below.

34.1.2 Frequentist decision theory

In **frequentist decision theory**, we treat the state of nature h as a fixed but unknown quantity, and treat the data $\mathbf{X}$ as random. Hence we take expectations wrt the data, which gives us the **frequentist risk**:

$$r(\delta|h) = \mathbb{E}_{p(\boldsymbol{x}|h)}\left[\ell(h, \delta(\boldsymbol{x}))\right] = \int p(\boldsymbol{x}|h)\ell(h, \delta(\boldsymbol{x}))d\boldsymbol{x} \tag{34.3}$$

The idea is that a good estimator will have low risk across many different datasets.

Unfortunately, the state of nature is not known, so the above quantity cannot be computed. There are several possible solutions to this. One idea is to put a prior distribution on h, and then to compute the **Bayes risk**, also called the **integrated risk**:

$$R_B(\delta) \triangleq \mathbb{E}_{p(h)}\left[r(\delta|h)\right] = \int p(h)p(\boldsymbol{x}|h)\ell(h,\delta(\boldsymbol{x}))\,dh\,d\boldsymbol{x} \tag{34.4}$$

A decision rule that minimizes the Bayes risk is known as a **Bayes estimator**.

Of course the use of a prior might seem undesirable in the context of frequentist statistics. We can therefore use the **maximum risk** instead. This is defined as follows:

$$R_{\max}(\delta) = \max_h r(\delta|h) \tag{34.5}$$

Minimizing the maximum risk gives rise to a **minimax estimator**:

$$\delta^* = \min_\delta \max_h r_h(\delta) \tag{34.6}$$

Minimax estimators have a certain appeal. However, computing them can be hard. And furthermore, they are very pessimistic. In fact, one can show that all minimax estimators are equivalent to Bayes estimators under a **least favorable prior**. In most statistical situations (excluding game theoretic ones), assuming nature is an adversary is not a reasonable assumption. See [BS94, p449] for further discussion of this point.

34.1.3 Bayesian decision theory

In **Bayesian decision theory**, we treat the data as an observed constant, $\boldsymbol{x}$, and the state of nature as an unknown random variable. The **posterior expected loss** for picking action a is defined as follows:

$$\rho(a|\boldsymbol{x}) \triangleq \mathbb{E}_{p(h|\boldsymbol{x})}\left[\ell(h,a)\right] = \int \ell(h,a)p(h|\boldsymbol{x})dh \tag{34.7}$$

We can define the posterior expected loss, or **Bayesian risk**, for an estimator using

$$\rho(\delta|\boldsymbol{x}) = \rho(\delta(\boldsymbol{x})|\boldsymbol{x}) \tag{34.8}$$

The **optimal policy** specifies what action to take so as to minimize the expected loss. This is given by

$$\delta^*(\boldsymbol{x}) = \operatorname*{argmin}_{a\in\mathcal{A}} \mathbb{E}_{p(h|\boldsymbol{x})}\left[\ell(h,a)\right] \tag{34.9}$$

An alternative, but equivalent, way of stating this result is as follows. Let us define a **utility function** $U(h,a)$ to be the desirability of each possible action in each possible state. If we set $U(h,a) = -\ell(h,a)$, then the optimal policy is as follows:

$$\delta^*(\boldsymbol{x}) = \operatorname*{argmax}_{a\in\mathcal{A}} \mathbb{E}_h\left[U(h,a)\right] \tag{34.10}$$

This is called the **maximum expected utility principle**.

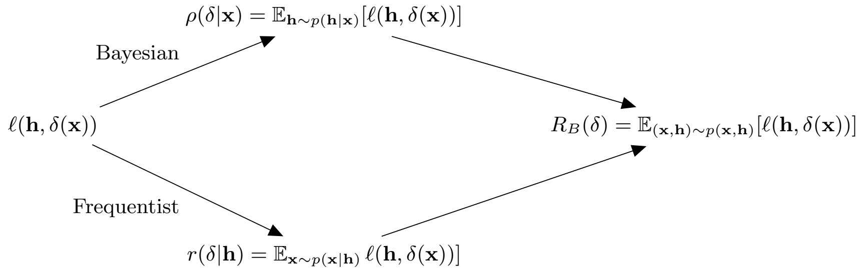


Figure 34.1: Illustration of how the Bayesian and frequentist approaches to decision making incur the same Bayes risk.

34.1.4 Frequentist optimality of the Bayesian approach

We see that the Bayesian approach, given by Equation (34.9), which picks the best action for each individual observation $\boldsymbol{x}$, will also optimize the Bayes risk in Equation (34.4), which picks the best policy for all possible observations. This follows from **Fubini's theorem** which lets us exchange the order of integration in a double integral (this is equivalent to the **law of iterated expectation**):

$$\begin{aligned} R_B(\delta) &= \mathbb{E}_{p(\boldsymbol{x})}\left[\rho(\delta|\boldsymbol{x})\right] = \mathbb{E}_{p(h|\boldsymbol{x})p(\boldsymbol{x})}\left[\ell(h, \delta(\boldsymbol{x}))\right] && (34.11) \\ &= \mathbb{E}_{p(h)}\left[r(\delta|h)\right] = \mathbb{E}_{p(h)p(\boldsymbol{x}|h)}\left[\ell(h, \delta(\boldsymbol{x}))\right] && (34.12) \end{aligned}$$

See Figure 34.1 for an illustration. The above result tells us that the Bayesian approach has optimal frequentist properties.

More generally, one can show that any **admissable policy**[1] is a Bayes policy with respect to some, possibly improper, prior distribution, a result known as **Wald's theorem** [Wal47]. (See [DR21] for a more general version of this result.) Thus we arguably lose nothing by "restricting" ourselves to the Bayesian approach (although we need to check that our modeling assumptions are adequate, a topic we discuss in Section 3.9). See [BS94, p448] for further discussion of this point.

Another advantage of the Bayesian approach is that is constructive, that is, it specifies how to create the optimal policy (estimator) given a particular dataset. By contrast, the frequentist approach allows you to use any estimator you like; it just derives the properties of this estimator across multiple datasets, but does not tell you how to create the estimator.

34.1.5 Examples of one-shot decision making problems

In the sections below, we give some common examples of **one-shot** decision making problems (i.e., making a single decision, not a sequence of decisions) that arise in ML applications.

1. An estimator is said to be **admissible** if it is not strictly dominated by any other estimator. We say that δ^1 **dominates** δ^2 if $R(\boldsymbol{\theta}, \delta^1) \leq R(\boldsymbol{\theta}, \delta^2)$ for all $\boldsymbol{\theta}$. The domination is said to be strict if the inequality is strict for some $\boldsymbol{\theta}^*$.

34.1.5.1 Classification

Suppose the states of nature correspond to class labels, so $\mathcal{H} = \mathcal{Y} = \{1, \ldots, C\}$. Furthermore, suppose the actions also correspond to class labels, so $\mathcal{A} = \mathcal{Y}$. In this setting, a very commonly used loss function is the **zero-one loss** $\ell_{01}(y^*, \hat{y})$, defined as follows:

$$\begin{array}{c|cc} & \hat{y}=0 & \hat{y}=1 \\ \hline y^*=0 & 0 & 1 \\ y^*=1 & 1 & 0 \end{array} \tag{34.13}$$

We can write this more concisely as follows:

$$\ell_{01}(y^*, \hat{y}) = \mathbb{I}\left(y^* \neq \hat{y}\right) \tag{34.14}$$

In this case, the posterior expected loss is

$$\rho(\hat{y}|\boldsymbol{x}) = p(\hat{y} \neq y^*|\boldsymbol{x}) = 1 - p(y^* = \hat{y}|\boldsymbol{x}) \tag{34.15}$$

Hence the action that minimizes the expected loss is to choose the most probable label:

$$\delta(\boldsymbol{x}) = \operatorname*{argmax}_{y \in \mathcal{Y}} p(y|\boldsymbol{x}) \tag{34.16}$$

This corresponds to the **mode** of the posterior distribution, also known as the **maximum a posteriori** or **MAP estimate**.

We can generalize the loss function to associate different costs for false positives and false negatives. We can also allow for a "**reject action**", in which the decision maker abstains from classifying when it is not sufficiently confident. This is called **selective prediction**; see Section 19.3.3 for details.

34.1.5.2 Regression

Now suppose the hidden state of nature is a scalar $h \in \mathbb{R}$, and the corresponding action is also a scalar, $y \in \mathbb{R}$. The most common loss for continuous states and actions is the ℓ_2 **loss**, also called **squared error** or **quadratic loss**, which is defined as follows:

$$\ell_2(h, y) = (h - y)^2 \tag{34.17}$$

In this case, the risk is given by

$$\rho(y|\boldsymbol{x}) = \mathbb{E}\left[(h-y)^2|\boldsymbol{x}\right] = \mathbb{E}\left[h^2|\boldsymbol{x}\right] - 2y\mathbb{E}\left[h|\boldsymbol{x}\right] + y^2 \tag{34.18}$$

The optimal action must satisfy the condition that the derivative of the risk (at that point) is zero (as explained in Chapter 6). Hence the optimal action is to pick the posterior mean:

$$\frac{\partial}{\partial y}\rho(y|\boldsymbol{x}) = -2\mathbb{E}\left[h|\boldsymbol{x}\right] + 2y = 0 \;\Rightarrow\; \delta(\boldsymbol{x}) = \mathbb{E}\left[h|\boldsymbol{x}\right] = \int h \, p(h|\boldsymbol{x}) dh \tag{34.19}$$

This is often called the **minimum mean squared error** estimate or **MMSE** estimate.

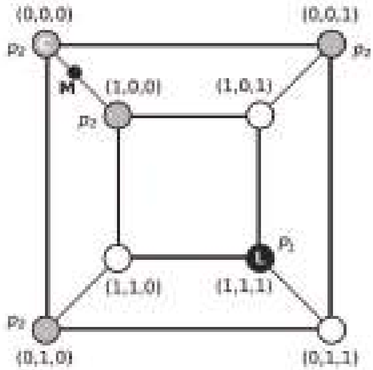


Figure 34.2: A distribution on a discrete space in which the mode (black point L, with probability p_1) is untypical of most of the probability mass (gray circles, with probability $p_2 < p_1$). The small black circle labeled M (near the top left) is the posterior mean, which is not well defined in a discrete state space. C (the top left vertex) is the centroid estimator, made up of the maximizer of the posterior marginals. See text for details. From Figure 1 of [CL07]. Used with kind permission of Luis Carvalho.

34.1.5.3 Parameter estimation

Suppose the states of nature correspond to unknown parameters, so $\mathcal{H} = \Theta = \mathbb{R}^D$. Furthermore, suppose the actions also correspond to parameters, so $\mathcal{A} = \Theta$. Finally, we assume the observed data (that is input to the policy/estimator) is a dataset, such as $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n) : n = 1 : N\}$. If we use quadratic loss, then the optimal action is to pick the posterior mean. If we use 0-1 loss, then the optimal action is to pick the posterior mode, i.e., the MAP estimate:

$$\delta(\mathcal{D}) = \hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmax}}\, p(\boldsymbol{\theta}|\mathcal{D}) \tag{34.20}$$

34.1.5.4 Estimating discrete parameters

The MAP estimate is the optimal estimate when the loss function is 0-1 loss, $\ell(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \mathbb{I}\left(\boldsymbol{\theta} \neq \hat{\boldsymbol{\theta}}\right)$, as we show in Section 34.1.5.1. However, this does not give any "partial credit" for estimating some of the components of $\boldsymbol{\theta}$ correctly. An alternative is to use the **Hamming loss**:

$$\ell(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \sum_{d=1}^{D} \mathbb{I}\left(\theta_d \neq \hat{\theta}_d\right) \tag{34.21}$$

In this case, one can show that the optimal estimator is the vector of **max marginals**

$$\hat{\boldsymbol{\theta}} = \left[\underset{\theta_d}{\operatorname{argmax}} \int_{\boldsymbol{\theta}_{-d}} p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}_{-d}\right]_{d=1}^{D} \tag{34.22}$$

This is also called the **maximizer of posterior marginals** or **MPM** estimate. Note that computing the max marginals involves marginalization and maximization, and thus depends on the whole distribution; this tends to be more robust than the MAP estimate [MMP87].

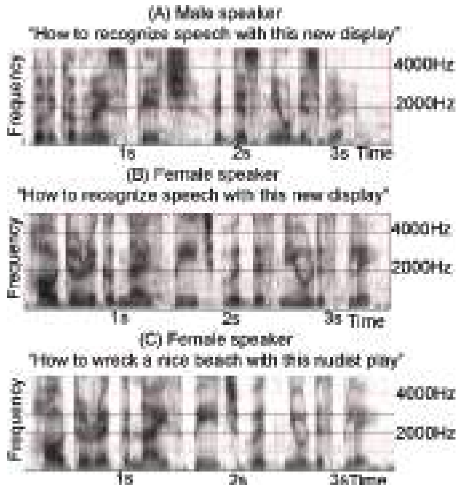


Figure 34.3: Spectograms for three different spoken sentences. The x-axis shows progression of time and the y-axis shows different frequency bands. The energy of the signal in different bands is shown as intensity in grayscale values with progression of time. (A) and (B) show spectrograms of the same sentence "How to recognize speech with this new display" spoken by two different speakers, male and female. Although the frequency characterization is similar, the formant frequencies are much more clearly defined in the speech of the female speaker. (C) shows the spectrogram of the utterance "How to wreck a nice beach with this nudist play" spoken by the same female speaker as in (B). (A) and (B) are not identical even though they are composed of the same words. (B) and (C) are similar to each other even though they are not the same sentences. From Figure 1.2 of [Gan07]. Used with kind permission of Madhavi Ganapathiraju.

For example, consider a problem in which we must estimate a vector of binary variables. Figure 34.2 shows a distribution on $\{0,1\}^3$, where points are arranged such that they are connected to their nearest neighbors, as measured by Hamming distance. The black state (circle) labeled L (configuration (1,1,1)) has probability p_1, and corresponds to the MAP estimate. The 4 gray states have probability $p_2 < p_1$; and the 3 white states have probability 0. Although the black state is the most probable, it is untypical of the posterior: all its nearest neighbors have probability zero, meaning it is very isolated. By contrast, the gray states, although slightly less probable, are all connected to other gray states, and together they constitute much more of the total probability mass.

In the example in Figure 34.2, we have $p(\theta_j = 0) = 3p_2$ and $p(\theta_j = 1) = p_2 + p_1$ for $j = 1:3$. If $2p_2 > p_1$, the vector of max marginals is $(0,0,0)$. This MPM estimate can be shown to be a **centroid estimator**, in the sense that it minimizes the squared distance to the posterior mean (the center of mass), yet it (usually) represents a valid configuration, unlike the actual mean (fractional estimates do not make sense for discrete problems). See [CL07] for further discussion of this point.

34.1.5.5 Structured prediction

In some problems, such as natural language processing or computer vision, the desired action is to return an output object $\boldsymbol{y} \in \mathcal{Y}$, such as a set of labels or body poses, that not only is probable given the input $\boldsymbol{x}$, but is also internally consistent. For example, suppose $\boldsymbol{x}$ is a sequence of phonemes and

$\boldsymbol{y}$ is a sequence of words. Although $\boldsymbol{x}$ might sound more like $\boldsymbol{y}$ = "How to wreck a nice beach" on a word-by-word basis, if we take the sequence of words into account then we may find (under a language model prior) that $\boldsymbol{y}$ = "How to recognize speech" is more likely overall. (See Figure 34.3.) We can capture this kind of dependency amongst outputs, given inputs, using a **structured prediction model**, such as a conditional random field (see Section 4.4).

In addition to modeling dependencies in $p(\boldsymbol{y}|\boldsymbol{x})$, we may prefer certain action choices $\hat{\boldsymbol{y}}$, which we capture in the loss function $\ell(\boldsymbol{y}, \hat{\boldsymbol{y}})$. For example, referring to Figure 34.3, we may be reluctant to assume the user said $\hat{y}_t$="nudist" at step t unless we are very confident of this prediction, since the cost of mis-categorizing this word may be higher than for other words.

Given a loss function, we can pick the optimal action using **minimum Bayes risk** decoding:

$$\hat{\boldsymbol{y}} = \min_{\hat{\boldsymbol{y}} \in \mathcal{Y}} \sum_{\boldsymbol{y} \in \mathcal{Y}} p(\boldsymbol{y}|\boldsymbol{x}) \ell(\boldsymbol{y}, \hat{\boldsymbol{y}}) \tag{34.23}$$

We can approximate the expectation empirically by sampling M solutions $\boldsymbol{y}^m \sim p(\boldsymbol{y}|\boldsymbol{x})$ from the posterior predictive distribution. (Ideally these are diverse from each other.) We use the same set of M samples to approximate the minimization to get

$$\hat{\boldsymbol{y}} \approx \min_{\boldsymbol{y}^i, i \in \{1,\ldots,M\}} \sum_{j \in \{1,\ldots,M\}} p(\boldsymbol{y}^j|\boldsymbol{x}) \ell(\boldsymbol{y}^j, \boldsymbol{y}^i) \tag{34.24}$$

This is called **empirical MBR** [Pre+17a], who applied it to computer vision problems. A similar approach was adopted in [Fre+22], who applied it to neural machine translation.

34.1.5.6 Fairness

Models trained with ML are increasingly being used to high-stakes applications, such as deciding whether someone should be released from prison or not, etc. In such applications, it is important that we focus not only on accuracy, but also on **fairness**. A variety of definitions for what is meant by fairness have been proposed (see e.g., [VR18]), many of which entail conflicting goals [Kle18]. Below we mention a few common definitions, which can all be interpreted decision theoretically.

We consider a binary classification problem with true label Y, predicted label $\hat{Y}$ and **sensitive attribute** S (such as gender or race). The concept of **equal opportunity** requires equal true positive rates across subgroups, i.e., $p(\hat{Y} = 1|Y = 1, S = 0) = p(\hat{Y} = 1|Y = 1, S = 1)$. The concept of **equal odds** requires equal true positive rates across subgroups, and also equal false positive rates across subgroups, i.e., $p(\hat{Y} = 1|Y = 0, S = 0) = p(\hat{Y} = 1|Y = 0, S = 1)$. The concept of **statistical parity** requires positive predictions to be unaffected by the value of the protected attribute, regardless of the true label, i.e., $p(\hat{Y} = 1|S = 0) = p(\hat{Y}|S = 1)$.

For more details on this topic, see e.g., [KR19].

34.2 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.

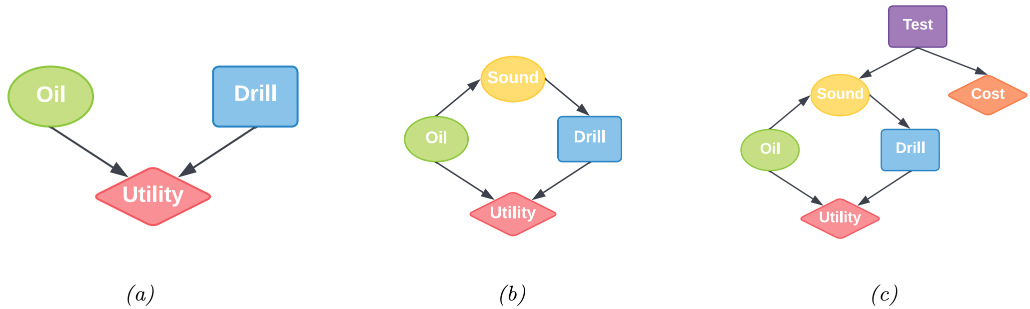


Figure 34.4: Influence diagrams for the oil wildcatter problem. Ovals are random variables (chance nodes), squares are decision (action) nodes, diamonds are utility (value) nodes. (a) Basic model. (b) An extension in which we have an information arc from the Sound chance node to the Drill decision node. (c) An extension in which we get to decide whether to perform a test or not, as well as whether to drill or not.

34.2.1 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 34.4(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

$$\mathrm{EU}(d) = \sum_{o=0}^{2} p(o) U(d, o) \tag{34.25}$$

We find $\mathrm{EU}(d = 0) = 0$ and $\mathrm{EU}(d = 1) = 20$ and hence the maximum expected utility is

$$\mathrm{MEU} = \max\{\mathrm{EU}(d = 0), \mathrm{EU}(d = 1)\} = \max\{0, 20\} = 20 \tag{34.26}$$

Thus the optimal action is to drill, $d^* = 1$.

34.2.2 Information arcs

Now let us consider a slight extension to the model, in which you have access to a measurement (called a "sounding"), which is a noisy indicator about the state of the oil well. Hence we add an $O \to S$ arc to the model. In addition, we assume that the outcome of the sounding test will be available before we decide whether to drill or not; hence we add an **information arc** from S to D. This is illustrated in Figure 34.4(b). Note that the utility depends on the action and the true state of the world, but not the measurement.

We assume the sounding variable can be in one of 3 states: $s = 0$ is a diffuse reflection pattern, suggesting no oil; $s = 1$ is an open reflection pattern, suggesting some oil; and $s = 2$ is a closed reflection pattern, indicating lots of oil. Since S is caused by O, we add an $O \to S$ arc to our model. Let us model the reliability of our sensor using the following conditional distribution for $p(S|O)$:

	$s = 0$	$s = 1$	$s = 2$
$o = 0$	0.6	0.3	0.1
$o = 1$	0.3	0.4	0.3
$o = 2$	0.1	0.4	0.5

Suppose the sounding observation is s. The posterior expected utility of performing action d is

$$\text{EU}(d|s) = \sum_{o=0}^{2} p(o|s)U(o, d) \tag{34.27}$$

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

$$\text{MEU} = \sum_s p(s)\text{EU}(d^*(s)|s) \tag{34.28}$$

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

$$\text{MEU} = 0.41 \times 0 + 0.35 \times 32.9 + 0.24 \times 87.5 = 32.2 \tag{34.29}$$

These numbers can be summarized in the **decision tree** shown in Figure 34.5.

34.2.3 Value of information

Now suppose you can choose whether to do the test or not. This can be modelled as shown in Figure 34.4(c), where we add a new test node T. If $T = 1$, we do the test, and S can enter states $\{0, 1, 2\}$, determined by O, exactly as above. If $T = 0$, we don't do the test, and S enters a special unknown state. There is also some cost associated with performing the test.

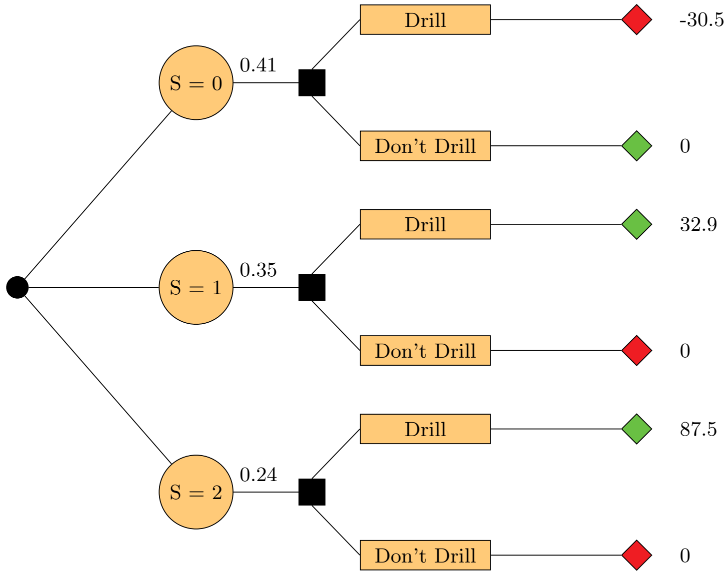


Figure 34.5: Decision tree for the oil wildcatter problem. Black circles are chance variables, black squares are decision nodes, diamonds are the resulting utilities. Green leaf nodes have higher utility than red leaf nodes.

Is it worth doing the test? This depends on how much our MEU changes if we know the outcome of the test (namely the state of S). If you don't do the test, we have MEU = 20 from Equation (34.26). If you do the test, you have MEU = 32.2 from Equation (34.29). So the improvement in utility if you do the test (and act optimally on its outcome) is \$12.2. This is called the **value of perfect information** (VPI). So we should do the test as long as it costs less than \$12.2.

In terms of graphical models, the VPI of a variable S can be determined by computing the MEU for the base influence diagram, $\mathcal{G}$, in Figure 34.4(b), and then computing the MEU for the same influence diagram where we add information arcs from S to the action node, and then computing the difference. In other words,

$$\text{VPI} = \text{MEU}(\mathcal{G} + S \to D) - \text{MEU}(\mathcal{G}) \tag{34.30}$$

where D is the decision node and S is the variable we are measuring. This will tell us whether it is worth adding obtaining measurement S.

34.2.4 Computing the optimal policy

In general, given an influence diagram, we can compute the optimal policy automatically by modifiying the variable elimination algorithm (Section 9.5), as explained in [LN01; KM08]. The basic idea is to work backwards from the final action, computing the optimal decision at each step, assuming all following actions are chosen optimally. When the influence diagram has a simple chain structure, as in a Markov decision process (Section 34.5), the result is equivalent to Bellman's equation (Section 34.5.5).

34.3 A/B testing

Suppose you are trying to decide which version of a product is likely to sell more, or which version of a drug is likely to work better. Let us call the versions you are choosing between A and B; sometimes version A is called the **control**, and version B is called the **treatment**. (Sometimes the different actions are called **"arms"**.)

A very common approach to such problems is to use an **A/B test**, in which you try both actions out for a while, by randomly assigning a different action to different subsets of the population, and then you measure the resulting accumulated **reward** from each action, and you pick the winner. (This is sometimes called a "**test and roll**" approach, since you test which method is best, and then roll it out for the rest of the population.)

A key problem in A/B testing is to come up with a decision rule, or policy, for deciding which action is best, after obtaining potentially noisy results during the test phase. Another problem is to choose how many people to assign to the treatment, n_1, and how many to the control, n_0. The fundamental tradeoff is that using larger values of n_1 and n_0 will help you collect more data and hence be more confident in picking the best action, but this incurs an **opportunity cost**, because the testing phase involves performing actions that may not result in the highest reward. (This is an example of the exploration-exploitation tradeoff, which we discuss more in Section 34.4.3.) In this section, we give a simple Bayesian decision theoretic analysis of this problem, following the presentation of [FB19].[2] More details on A/B testing can be found in [KTX20].

34.3.1 A Bayesian approach

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 $j = 0$ corresponds to the control (action A), $j = 1$ corresponds to the treatment (action B), and n_j is the number of samples you collect from group j. The parameters μ_j are the expected reward for action j; our goal is to estimate these parameters. (For simplicity, we assume the σ_j^2 are known.)

We will adopt a Bayesian approach, which is well suited to sequential decision problems. For 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 mean reward for action j, and τ_j is our confidence in this prior. We assume the prior parameters are known. (In practice we can use an empirical Bayes approach, as we discuss in Section 34.3.2.)

34.3.1.1 Optimal policy

Initially we assume the sample size of the experiment (i.e., the values n_1 for the treatment and n_0 for the control) are known. Our goal is to compute the optimal policy or decision rule $\pi(\boldsymbol{y}_1, \boldsymbol{y}_0)$, which specifies which action to deploy, where $\boldsymbol{y}_j = (y_{1j}, \ldots, y_{n_j,j})$ is the data from action j.

The optimal policy is simple: choose the action with the greater expected posterior expected reward:

$$\pi^*(\boldsymbol{y}_1, \boldsymbol{y}_0) = \begin{cases} 1 & \text{if } \mathbb{E}[\mu_1|\boldsymbol{y}_1] \geq \mathbb{E}[\mu_0|\boldsymbol{y}_0] \\ 0 & \text{if } \mathbb{E}[\mu_1|\boldsymbol{y}_1] < \mathbb{E}[\mu_0|\boldsymbol{y}_0] \end{cases} \tag{34.31}$$

2. 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`.

All that remains is to compute the posterior. over the unknown parameters, μ_j. Applying Bayes' rule for Gaussians (Equation (2.121)), we find that the corresponding posterior is given by

$$p(\mu_j|\boldsymbol{y}_j, n_j) = \mathcal{N}(\mu_j|\, \widehat{m}_j, \widehat{\tau}_j^2) \tag{34.32}$$

$$1/\, \widehat{\tau}_j^2 = n_j/\sigma_j^2 + 1/\tau_j^2 \tag{34.33}$$

$$\widehat{m}_j \,/\, \widehat{\tau}_j^2 = n_j \overline{y}_j/\sigma_j^2 + m_j/\tau_j^2 \tag{34.34}$$

We see that the posterior precision (inverse variance) is a weighted sum of the prior precision plus n_j units of measurement precision. We also see that the posterior precision weighted mean is a sum of the prior precision weighted mean and the measurement precision weighted mean.

Given the posterior, we can plug $\widehat{m}_j$ into Equation (34.31). In the fully symmetric case, where $n_1 = n_0$, $m_1 = m_0 = m$, $\tau_1 = \tau_0 = \tau$, and $\sigma_1 = \sigma_0 = \sigma$, we find that the optimal policy is to simply "pick the winner", which is the arm with higher empirical performance:

$$\pi^*(\boldsymbol{y}_1, \boldsymbol{y}_0) = \mathbb{I}\left(\frac{m}{\tau^2} + \frac{\overline{y}_1}{\sigma^2} > \frac{m}{\tau^2} + \frac{\overline{y}_0}{\sigma^2}\right) = \mathbb{I}\left(\overline{y}_1 > \overline{y}_0\right) \tag{34.35}$$

However, when the problem is asymmetric, we need to take into account the different sample sizes and/or different prior beliefs.

34.3.1.2 Optimal sample size

We now discuss how to compute the optimal sample size for each arm of the experiment, i.e, the values n_0 and n_1. We assume the total population size is N, and we cannot reuse people from the testing phase,

The prior expected reward in the testing phase is given by

$$\mathbb{E}\left[R_{\text{test}}\right] = n_0 m_0 + n_1 m_1 \tag{34.36}$$

The expected reward in the roll phase depends on the decision rule $\pi(\boldsymbol{y}_1, \boldsymbol{y}_0)$ that we use:

$$\mathbb{E}_\pi\left[R_{\text{roll}}\right] = \int_{\mu_1}\int_{\mu_0}\int_{\boldsymbol{y}_1}\int_{\boldsymbol{y}_0} (N - n_1 - n_0)\left(\pi(\boldsymbol{y}_1, \boldsymbol{y}_0)\mu_1 + (1 - \pi(\boldsymbol{y}_1, \boldsymbol{y}_0))\mu_0\right) \tag{34.37}$$

$$\times\, p(\boldsymbol{y}_0|\mu_0) p(\boldsymbol{y}_1|\mu_1) p(\mu_0) p(\mu_1) d\boldsymbol{y}_0 d\boldsymbol{y}_1 d\mu_0 d\mu_1 \tag{34.38}$$

For $\pi = \pi^*$ one can show that this equals

$$\mathbb{E}\left[R_{\text{roll}}\right] \triangleq \mathbb{E}_{\pi*}\left[R_{\text{roll}}\right] = (N - n_1 - n_0)\left(m_1 + e\Phi(\frac{e}{v}) + v\phi(\frac{e}{v})\right) \tag{34.39}$$

where ϕ is the Gaussian pdf, Φ is the Gaussian cdf, $e = m_0 - m_1$ and

$$v = \sqrt{\frac{\tau_1^4}{\tau_1^2 + \sigma_1^2/n_1} + \frac{\tau_0^4}{\tau_0^2 + \sigma_0^2/n_0}} \tag{34.40}$$

In the fully symmetric case, Equation (34.39) simplifies to

$$\mathbb{E}\left[R_{\text{roll}}\right] = \underbrace{(N - 2n)m}_{R_a} + \underbrace{(N - 2n)\frac{\sqrt{2}\tau^2}{\sqrt{\pi}\sqrt{2\tau^2 + \frac{2}{n}\sigma^2}}}_{R_b} \tag{34.41}$$

This has an intuitive interpretation. The first term, R_a, is the prior reward we expect to get before we learn anything about the arms. The second term, R_b, is the reward we expect to see by virtue of picking the optimal action to deploy.

Let us we write $R_b = (N - 2n)R_i$, where R_i is the incremental gain. We see that the incremental gain increases with n, because we are more likely to pick the correct action with a larger sample size; however, this gain can only be accrued for a smaller number of people, as shown by the $N - 2n$ prefactor. (This is a consequence of the explore-exploit tradeoff.)

The total expected reward is given by adding Equation (34.36) and Equation (34.41):

$$\mathbb{E}[R] = \mathbb{E}[R_{\text{test}}] + \mathbb{E}[R_{\text{roll}}] = Nm + (N - 2n)\left(\frac{\sqrt{2}\tau^2}{\sqrt{\pi}\sqrt{2\tau^2 + \frac{2}{n}\sigma^2}}\right) \tag{34.42}$$

(The equation for the nonsymmetric case is given in [FB19].)

We can maximize the expected reward in Equation (34.42) 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

$$n^* = \sqrt{\frac{N}{4}u^2 + \left(\frac{3}{4}u^2\right)^2} - \frac{3}{4}u^2 \leq \sqrt{N}\frac{\sigma}{2\tau} \tag{34.43}$$

where $u^2 = \frac{\sigma^2}{\tau^2}$. Thus we see that the optimal sample size n^* increases as the observation noise σ increases, since we need to collect more data to be confident of the right decision. However, the optimal sample size decreases with τ, 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.

34.3.1.3 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 μ_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

$$\mathbb{E}[R|PI] = N\left(m + \frac{\tau}{\sqrt{\pi}}\right) \tag{34.44}$$

Therefore the regret from the optimal policy is given by

$$\mathbb{E}[R|PI] - (\mathbb{E}[R_{\text{test}}|\pi^*] + \mathbb{E}[R_{\text{roll}}|\pi^*]) = N\frac{\tau}{\sqrt{\pi}}\left(1 - \frac{\tau}{\sqrt{\tau^2 + \frac{\sigma^2}{n^*}}}\right) + \frac{2n^*\tau^2}{\sqrt{\pi}\sqrt{\tau^2 + \frac{\sigma^2}{n^*}}} \tag{34.45}$$

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].

34.3.1.4 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 δ is small. Instead, it makes more sense to optimize total expected reward, using the method in Section 34.3.1.1.

Nevertheless, we may want to know the probability that we have picked the wrong arm if we use the policy from Section 34.3.1.1. In the symmetric case, this is given by the following:

$$\Pr(\pi(\boldsymbol{y}_1, \boldsymbol{y}_0) = 1|\mu_1 < \mu_0) = \Pr(Y_1 - Y_0 > 0|\mu_1 < \mu_0) = 1 - \Phi\left(\frac{\mu_1 - \mu_0}{\sigma\sqrt{\frac{1}{n_1} + \frac{1}{n_0}}}\right) \tag{34.46}$$

The above expression assumed that μ_j are known. Since they are not known, we can compute the expected error rate using $\mathbb{E}\left[\Pr(\pi(\boldsymbol{y}_1, \boldsymbol{y}_0) = 1|\mu_1 < \mu_0)\right]$. By symmetry, the quantity $\mathbb{E}\left[\Pr(\pi(\boldsymbol{y}_1, \boldsymbol{y}_0) = 0|\mu_1 > \mu_0)\right]$ is the same. One can show that both quantities are given by

$$\text{Prob. error} = \frac{1}{4} - \frac{1}{2\pi}\arctan\left(\frac{\sqrt{2}\tau}{\sigma}\sqrt{\frac{n_1 n_0}{n_1 + n_0}}\right) \tag{34.47}$$

As expected, the error rate decreases with the sample size n_1 and n_0, increases with observation noise σ, and decreases with variance of the effect size τ. Thus a policy that minimizes the classification error will also maximize expected reward, but it may pick an overly large sample size, since it does not take into account the magnitude of δ.

34.3.2 Example

In this section, we give a simple example of the above framework. Suppose our goal is to do **website testing**, where have two different versions of a webpage that we want to compare in terms of their **click through rate**. The observed data is now binary, $y_{ij} \sim \text{Ber}(\mu_j)$, so it is natural to use a beta prior, $\mu_j \sim \text{Beta}(\alpha, \beta)$ (see Section 3.4.1). However, in this case the optimal sample size and decision rule is harder to compute (see [FB19; Sta+17] for details). As a simple approximation, we can assume $\overline{y}_{ij} \sim \mathcal{N}(\mu_j, \sigma^2)$, 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 $\sigma^2 = m(1-m)$.

To set the Gaussian prior, [FB19] used empirical data from about 2000 prior A/B tests. For each test, they observed the number of times the page was served with each of the two variations, as well as the total number of times a user clicked on each version. Given this data, they used a hierarchical Bayesian model to infer $\mu_j \sim \mathcal{N}(m = 0.68, \tau = 0.03)$. This prior implies that the expected effect size is quite small, $\mathbb{E}\left[|\mu_1 - \mu_0|\right] = 0.023$. (This is consistent with the results in [Aze+20], who found that most changes made to the Microsoft Bing EXP platform had negligible effect, although there were occasionally some "big hits".)

With this prior, and assuming a population of $N = 100,000$, Equation (34.43) says that the optimal number of trials to run is $n_1^* = n_0^* = 2284$. The expected reward (number of clicks or **conversions**) in the testing phase is $\mathbb{E}\left[R_{\text{test}}\right] = 3106$, and in the deployment phase $\mathbb{E}\left[R_{\text{roll}}\right] = 66,430$, for a total reward of $69,536$. The expected error rate is 10%.

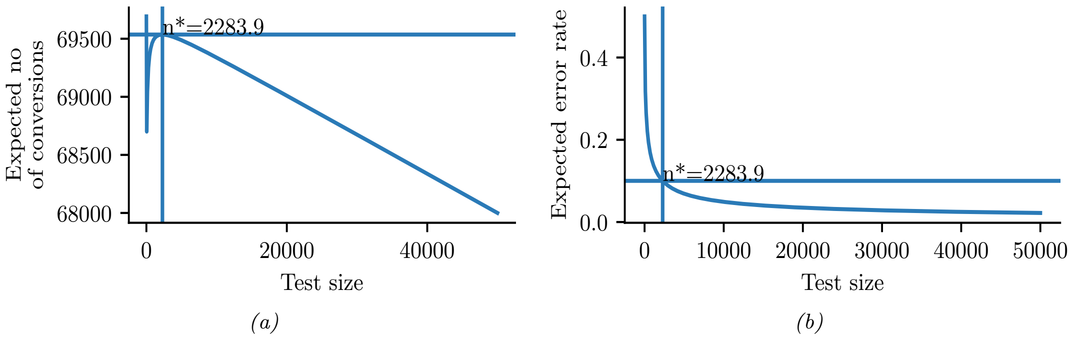


Figure 34.6: Total expected profit (a) and error rate (b) as a function of the sample size used for website testing. Generated by ab_test_demo.ipynb.

In Figure 34.6a, we plot the expected reward vs the size of the test phase n. We see that the reward increases sharply with n to the global maximum at $n^* = 2284$, and then drops off more slowly. This indicates that it is better to have a slightly larger test than one that is too small by the same amount. (However, when using a heavy tailed model, [Aze+20] finds that it is better to do lots of smaller tests.)

In Figure 34.6b, we plot the probability of picking the wrong action vs n. We see that tests that are larger than optimal only reduce this error rate marginally. Consequently, if you want to make the misclassification rate low, you may need a large sample size, particularly if $\mu_1 - \mu_0$ is small, since then it will be hard to detect the true best action. However, it is also less important to identify the best action in this case, since both actions have very similar expected reward. This explains why classical methods for A/B testing based on frequentist statistics, which use hypothesis testing methods to determine if A is better than B, may often recommend sample sizes that are much larger than necessary. (See [FB19] and references therein for further discussion.)

34.4 Contextual bandits

This section is co-authored with Lihong Li.

In Section 34.3, 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, $\boldsymbol{y}_1$ and $\boldsymbol{y}_0$, 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]).

34.4.1 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 $R(s, a) = \mathbb{E}[R|a]$.[3]

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, $a_t \in \{1, \ldots, K\}$. The goal is to quickly figure out which machine pays out the most money, and then to keep playing that one until you become as rich as possible.

We can extend this model by defining a **contextual bandit**, in which the input to the policy at each step is a randomly chosen state or context $s_t \in \mathcal{S}$. The states evolve over time according to some arbitrary process, $s_t \sim p(s_t|\boldsymbol{s}_{1:t-1})$, independent of the actions of the agent. The policy 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 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 $\mathcal{D}_t = (s_t, a_t, r_t)$, to update its policy, to maximize expected reward.

In the **finite horizon** formulation of (contextual) bandits, the goal is to maximize the expected **cumulative reward**:

$$J \triangleq \sum_{t=1}^{T} \mathbb{E}_{p_R(r_t|s_t,a_t)\pi_t(a_t|s_t)p(s_t|\boldsymbol{s}_{1:t-1})}[r_t] = \sum_{t=1}^{T} \mathbb{E}[r_t] \tag{34.48}$$

(Note that the reward is accrued at each step, even while the agent updates its policy; this is sometimes called "**earning while learning**".) In the **infinite horizon** formulation, where $T = \infty$, the cumulative reward may be infinite. To prevent J from being unbounded, we introduce a **discount factor** $0 < \gamma < 1$, so that

$$J \triangleq \sum_{t=1}^{\infty} \gamma^{t-1} \mathbb{E}[r_t] \tag{34.49}$$

The quantity γ can be interpreted as the probability that the agent is terminated at any moment in time (in which case it will cease to accumulate reward).

Another way to write this is as follows:

$$J = \sum_{t=1}^{\infty} \gamma^{t-1} \mathbb{E}[r_t] = \sum_{t=1}^{\infty} \gamma^{t-1} \mathbb{E}\left[\sum_{a=1}^{K} R_a(s_t, a_t)\right] \tag{34.50}$$

where we define

$$R_a(s_t, a_t) = \begin{cases} R(s_t, a) & \text{if } a_t = a \\ 0 & \text{otherwise} \end{cases} \tag{34.51}$$

3. 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**.

Thus we conceptually evaluate the reward for all arms, but only the one that was actually chosen (namely a_t) gives a non-zero value to the agent, namely r_t.

There are many extensions of the basic bandit problem. A natural one is to allow the agent to perform **multiple plays**, choosing $M \leq K$ distinct arms at once. Let $\boldsymbol{a}_t$ be the corresponding action vector which specifies the identity of the chosen arms. Then we define the reward to be

$$r_t = \sum_{a=1}^{K} R_a(s_t, \boldsymbol{a}_t) \tag{34.52}$$

where

$$R_a(s_t, \boldsymbol{a}_t) = \begin{cases} R(s_t, a) & \text{if } a \in \boldsymbol{a}_t \\ 0 & \text{otherwise} \end{cases} \tag{34.53}$$

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

$$r_t = \sum_{a=1}^{K} R_a(s_t^a, \boldsymbol{a}_t) \tag{34.54}$$

where $s_t^a \sim p(s_t^a | \boldsymbol{s}_{1:t-1}^a)$ 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.

34.4.2 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; Aga+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].)

34.4.3 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

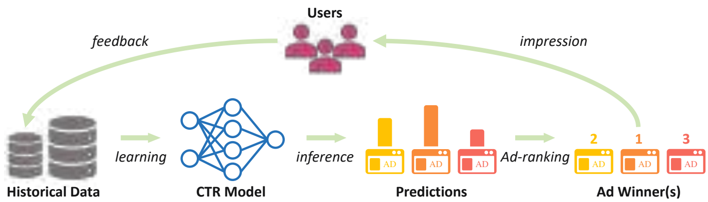


Figure 34.7: Illustration of the feedback problem in online advertising and recommendation systems. The click through rate (CTR) model is used to decide what ads to show, which affects what data is collected, which affects how the model learns. From Figure 1–2 of [Du+21]. Used with kind permission of Chao Du.

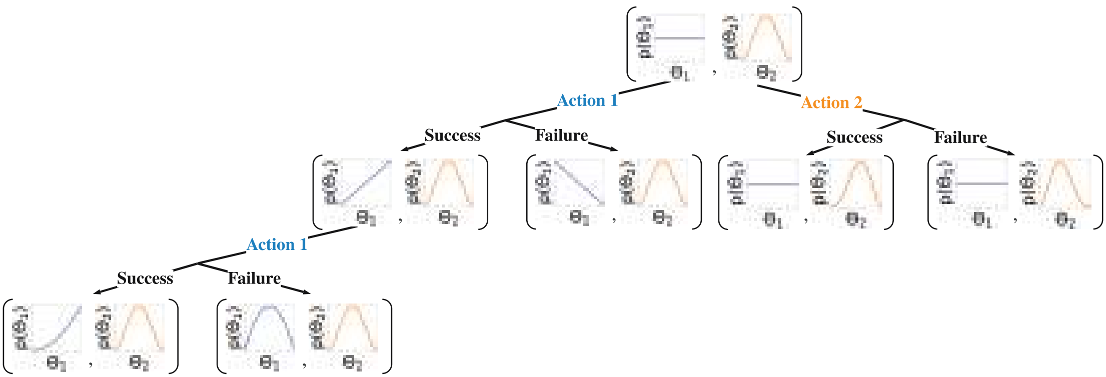


Figure 34.8: Illustration of sequential belief updating for a two-armed beta-Bernoulli bandit. The prior for the reward for action 1 is the (blue) uniform distribution Beta$(1, 1)$*; the prior for the reward for action 2 is the (orange) unimodal distribution* Beta$(2, 2)$*. We update the parameters of the belief state based on the chosen action, and based on whether the observed reward is success (1) or failure (0).*

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 34.7. This is different from supervised learning, where the data is drawn iid from a fixed distribution (see e.g., [Jeu+19] for details).

We discuss some solutions to the exploration-exploitation problem below.

34.4.4 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 $\boldsymbol{b}_t = p(\boldsymbol{\theta}|\boldsymbol{h}_t)$, where $\boldsymbol{h}_t = \{s_{1:t-1}, a_{1:t-1}, r_{1:t-1}\}$ is the history of observations; this is known as the **belief state** or **information state**. It is a finite sufficient statistic for the history $\boldsymbol{h}_t$. The belief state can be updated deterministically using Bayes' rule:

$$\boldsymbol{b}_t = \text{BayesRule}(\boldsymbol{b}_{t-1}, a_t, r_t) \tag{34.55}$$

For example, consider a context-free **Bernoulli bandit**, where $p_R(r|a) = \text{Ber}(r|\mu_a)$, and $\mu_a = p_R(r = 1|a) = R(a)$ is the expected reward for taking action a. Suppose we use a factored beta prior

$$p_0(\boldsymbol{\theta}) = \prod_a \text{Beta}(\mu_a|\alpha_0^a, \beta_0^a) \tag{34.56}$$

where $\boldsymbol{\theta} = (\mu_1, \ldots, \mu_K)$. We can compute the posterior in closed form, as we discuss in Section 3.4.1. In particular, we find

$$p(\boldsymbol{\theta}|\mathcal{D}_t) = \prod_a \text{Beta}(\mu_a|\underbrace{\alpha_0^a + N_t^0(a)}_{\alpha_t^a}, \underbrace{\beta_0^a + N_t^1(a)}_{\beta_t^a}) \tag{34.57}$$

where

$$N_t^r(a) = \sum_{s=1}^{t-1} \mathbb{I}\left(a_t = a, r_t = r\right) \tag{34.58}$$

This is illustrated in Figure 34.8 for a two-armed Bernoulli bandit.

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 from Section 3.4.3. In the case of contextual bandits, the problem becomes more complicated. If we assume a **linear regression bandit**, $p_R(r|s, a; \boldsymbol{\theta}) = \mathcal{N}(r|\boldsymbol{\phi}(s, a)^\mathsf{T}\boldsymbol{\theta}, \sigma^2)$, we can use Bayesian linear regression to compute $p(\boldsymbol{\theta}|\mathcal{D}_t)$ in closed form, as we discuss in Section 15.2. If we assume a **logistic regression bandit**, $p_R(r|s, a; \boldsymbol{\theta}) = \text{Ber}(r|\sigma(\boldsymbol{\phi}(s, a)^\mathsf{T}\boldsymbol{\theta}))$, we can use Bayesian logistic regression to compute $p(\boldsymbol{\theta}|\mathcal{D}_t)$, as we discuss in Section 15.3.5. If we have a **neural bandit** of the form $p_R(r|s, a; \boldsymbol{\theta}) = \text{GLM}(r|f(s, a; \boldsymbol{\theta}))$ for some nonlinear function f, then posterior inference becomes more challenging, as we discuss in Chapter 17. However, standard techniques, such as the extended Kalman filter (Section 17.5.2) can be applied. (For a way to scale this approach to large DNNs, see the "**subspace neural bandit**" approach of [DMKM22].)

Regardless of the algorithmic details, we can represent the belief state update as follows:

$$p(\boldsymbol{b}_t|\boldsymbol{b}_{t-1}, a_t, r_t) = \mathbb{I}\left(\boldsymbol{b}_t = \text{BayesRule}(\boldsymbol{b}_{t-1}, a_t, r_t)\right) \tag{34.59}$$

The observed reward at each step is then predicted to be

$$p(r_t|\boldsymbol{b}_t) = \int p_R(r_t|s_t, a_t; \boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{b}_t) d\boldsymbol{\theta} \tag{34.60}$$

We see that this is a special form of a (controlled) Markov decision process (Section 34.5) known as a **belief-state MDP**.

In the special case of context-free bandits with a finite number of arms, the optimal policy of this belief state MDP can be computed using dynamic programming (see Section 34.6); the result can be represented as a table of action probabilities, $\pi_t(a_1, \ldots, a_K)$, for each step; this is known as the **Gittins index** [Git89]. However, computing the optimal policy for general contextual bandits is intractable [PT87], so we have to resort to approximations, as we discuss below.

34.4.5 Upper confidence bounds (UCBs)

The optimal solution to explore-exploit is intractable. However, an intuitively sensible approach is based on the principle known as "**optimism in the face of uncertainty**". The principle selects actions greedily, but based on optimistic estimates of their rewards. The most important class of strategies with this principle are collectively called **upper confidence bound** or **UCB** methods.

To use a UCB strategy, the agent maintains an optimistic reward function estimate $\tilde{R}_t$, so that $\tilde{R}_t(s_t, a) \geq R(s_t, a)$ for all a with high probability, and then chooses the greedy action accordingly:

$$a_t = \underset{a}{\operatorname{argmax}}\, \tilde{R}_t(s_t, a) \tag{34.61}$$

UCB can be viewed a form of **exploration bonus**, where the optimistic estimate encourages exploration. Typically, the amount of optimism, $\tilde{R}_t - R$, decreases over time so that the agent gradually reduces exploration. With properly constructed optimistic reward estimates, the UCB strategy has been shown to achieve near-optimal regret in many variants of bandits [LS19]. (We discuss regret in Section 34.4.7.)

The optimistic function $\tilde{R}$ can be obtained in different ways, sometimes in closed forms, as we discuss below.

34.4.5.1 Frequentist approach

One approach is to use a **concentration inequality** [BLM16] to derive a high-probability upper 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 (often the MLE), and δ_t is a properly selected function. An optimistic reward is then obtained by setting $\tilde{R}_t(s, a) = \hat{R}_t(s, a) + \delta_t(s, a)$.

As an example, consider again the context-free Bernoulli bandit, $R(a) \sim \mathrm{Ber}(\mu(a))$. The MLE $\hat{R}_t(a) = \hat{\mu}_t(a)$ is given by the empirical average of observed rewards whenever action a was taken:

$$\hat{\mu}_t(a) = \frac{N_t^1(a)}{N_t(a)} = \frac{N_t^1(a)}{N_t^0(a) + N_t^1(a)} \tag{34.62}$$

where $N_t^r(a)$ is the number of times (up to step $t-1$) that action a has been tried and the observed reward was r, and $N_t(a)$ is the total number of times action a has been tried:

$$N_t(a) = \sum_{s=1}^{t-1} \mathbb{I}\left(a_t = a\right) \tag{34.63}$$

Then the **Chernoff-Hoeffding inequality** [BLM16] leads to $\delta_t(a) = c/\sqrt{N_t(a)}$ for some proper constant c, so

$$\tilde{R}_t(a) = \hat{\mu}_t(a) + \frac{c}{\sqrt{N_t(a)}} \tag{34.64}$$

34.4.5.2 Bayesian approach

We may also derive $\tilde{R}$ from Bayesian inference. If we use a beta prior, we can compute the posterior in closed form, as shown in Equation (34.57). The posterior mean is $\hat{\mu}_t(a) = \mathbb{E}\left[\mu(a)|\boldsymbol{h}_t\right] = \frac{\alpha_t^a}{\alpha_t^a + \beta_t^a}$. From Equation (3.17), the posterior standard deviation is approximately

$$\hat{\sigma}_t(a) = \sqrt{\mathbb{V}\left[\mu(a)|\boldsymbol{h}_t\right]} \approx \sqrt{\frac{\hat{\mu}_t(a)(1 - \hat{\mu}_t(a))}{N_t(a)}} \tag{34.65}$$

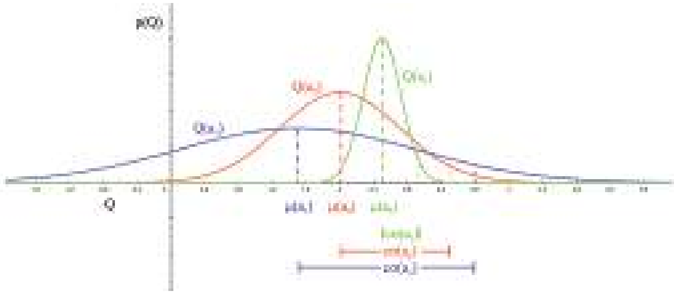


Figure 34.9: Illustration of the reward distribution $Q(a)$ for 3 different actions, and the corresponding lower and upper confidence bounds. From [Sil18]. Used with kind permission of David Silver.

We can use similar techniques for a Gaussian bandit, where $p_R(R|a, \boldsymbol{\theta}) = \mathcal{N}(R|\mu_a, \sigma_a^2)$, μ_a is the expected reward, and σ_a^2 the variance. If we use a conjugate prior, we can compute $p(\mu_a, \sigma_a|\mathcal{D}_t)$ in closed form (see Section 3.4.3). Using an uninformative version of the conjugate prior, we find $\mathbb{E}[\mu_a|\boldsymbol{h}_t] = \hat{\mu}_t(a)$, which is just the empirical mean of rewards for action a. The uncertainty in this estimate is the standard error of the mean, given by Equation (3.133), i.e., $\sqrt{\mathbb{V}[\mu_a|\boldsymbol{h}_t]} = \hat{\sigma}_t(a)/\sqrt{N_t(a)}$, where $\hat{\sigma}_t(a)$ is the empirical standard deviation of the rewards for action a.

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

$$\tilde{R}_t(a) = \hat{\mu}_t(a) + c\hat{\sigma}_t(a) \tag{34.66}$$

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.

34.4.5.3 Example

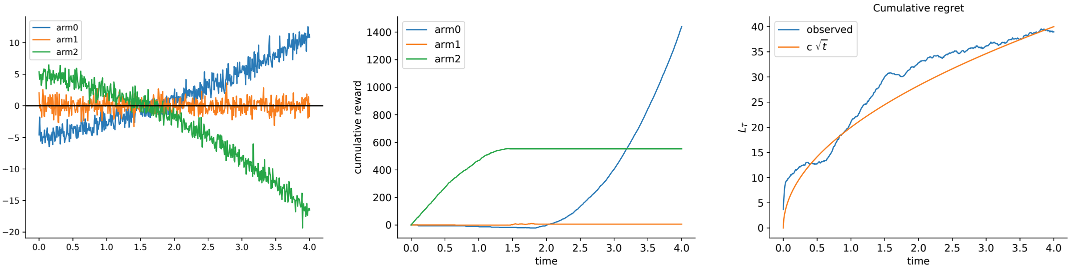
Figure 34.9 illustrates the UCB principle for a Gaussian bandit. We assume there are 3 actions, and we represent $p(R(a)|\mathcal{D}_t)$ using a Gaussian. We show the posterior means $Q(a) = \mu(a)$ with a vertical dotted line, and the scaled posterior standard deviations $c\sigma(a)$ as a horizontal solid line.

34.4.6 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 $\pi_t(a|s_t, \boldsymbol{h}_t) = p_a$, where p_a is the probability that a is the optimal action. This can be computed using

$$p_a = \Pr(a = a_*|s_t, \boldsymbol{h}_t) = \int \mathbb{I}\left(a = \operatorname*{argmax}_{a'} R(s_t, a'; \boldsymbol{\theta})\right) p(\boldsymbol{\theta}|\boldsymbol{h}_t) d\boldsymbol{\theta} \tag{34.67}$$

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.



(a) (b) (c)

Figure 34.10: Illustration of Thompson sampling applied to a linear-Gaussian contextual bandit. The context has the form $\boldsymbol{s}_t = (1, t, t^2)$. (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.ipynb.

To see how we can implement this method, note that we can compute the expression in Equation (34.67) by using a single Monte Carlo sample $\tilde{\boldsymbol{\theta}}_t \sim p(\boldsymbol{\theta}|\boldsymbol{h}_t)$. We then plug in this parameter into our reward model, and greedily pick the best action:

$$a_t = \underset{a'}{\operatorname{argmax}}\, R(s_t, a'; \tilde{\boldsymbol{\theta}}_t) \tag{34.68}$$

This sample-then-exploit approach will choose actions with exactly the desired probability, since

$$p_a = \int \mathbb{I}\left(a = \underset{a'}{\operatorname{argmax}}\, R(s_t, a'; \tilde{\boldsymbol{\theta}}_t)\right) p(\tilde{\boldsymbol{\theta}}_t|\boldsymbol{h}_t) = \Pr_{\tilde{\boldsymbol{\theta}}_t \sim p(\boldsymbol{\theta}|\boldsymbol{h}_t)}(a = \underset{a'}{\operatorname{argmax}}\, R(s_t, a'; \tilde{\boldsymbol{\theta}}_t)) \tag{34.69}$$

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 34.10, we give a simple example of Thompson sampling applied to a linear regression bandit. The context has the form $\boldsymbol{s}_t = (1, t, t^2)$. The true reward function has the form $R(\boldsymbol{s}_t, a) = \boldsymbol{w}_a^\mathsf{T} \boldsymbol{s}_t$. The weights per arm are chosen as follows: $\boldsymbol{w}_0 = (-5, 2, 0.5)$, $\boldsymbol{w}_1 = (0, 0, 0)$, $\boldsymbol{w}_2 = (5, -1.5, -1)$. 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 34.10(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 34.10(b).

34.4.7 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 π_*, which knows the true reward function. (Note that the oracle policy will in general be better than the Bayes optimal policy, which we disucssed in Section 34.4.4.)

Specifically, let π_t be the agent's policy at time t. Then the **per-step regret** at t is defined as

$$l_t \triangleq \mathbb{E}_{p(s_t)}\left[R(s_t, \pi_*(s_t))\right] - \mathbb{E}_{\pi_t(a_t|s_t)p(s_t)}\left[R(s_t, a_t)\right] \tag{34.70}$$

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 **cumulative regret**, also called the **total regret** or just the **regret**, which is defined as

$$L_T \triangleq \mathbb{E}\left[\sum_{t=1}^{T} l_t\right] \tag{34.71}$$

Here the expectation is with respect to randomness in determining π_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

$$L_T = \mathbb{E}\left[\sum_{t=1}^{T} R_* - R(a_t)\right] = \sum_{a\in\mathcal{A}} \mathbb{E}\left[N_{T+1}(a)\right](R_* - R(a)) = \sum_{a\in\mathcal{A}} \mathbb{E}\left[N_{T+1}(a)\right]\Delta_a \tag{34.72}$$

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 34.10(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 (3.133)). 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 34.9.)

The following fundamental result is proved by [LR85] for the asymptotic regret (under certain mild assumptions not given here):

$$\liminf_{T\to\infty} L_T \geq \log T \sum_{a:\Delta_a>0} \frac{\Delta_a}{D_{\mathbb{KL}}\left(p_R(a) \parallel p_R(a_*)\right)} \tag{34.73}$$

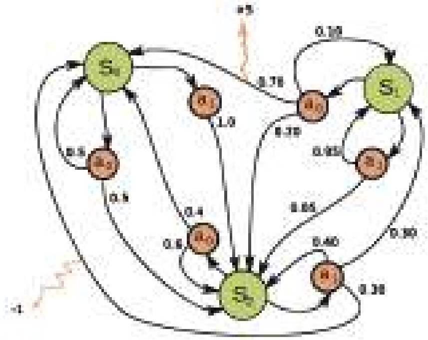


Figure 34.11: Illustration of an MDP as a finite state machine (FSM). The MDP has three discrete states (green cirlces), two discrete actions (orange circles), and two non-zero rewards (orange arrows). The numbers on the black edges represent state transition probabilities, e.g., $p(s' = s_0|a = a_0, s' = s_0) = 0.7$; most state transitions are impossible (probability 0), so the graph is sparse. The numbers on the yellow wiggly edges represent expected rewards, e.g., $R(s = s_1, a = a_0, s' = s_0) = +5$; state transitions with zero reward are not annotated. From https://en.wikipedia.org/wiki/Markov_decision_process. Used with kind permission of Wikipedia author waldoalvarez.

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.

34.5 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 decision process** or **MDP**, as we explain in detail below. This model forms the foundation of reinforcement learning, which we discuss in Chapter 35.

34.5.1 Basics

A **Markov decision process** [Put94] can be used to model the interaction of an **agent** and an **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 states, $\mathcal{A}$ a set of actions the agent can take, p_T a **transition model**, p_R a **reward model**, and p_0 the initial state distribution. The interaction starts at time $t = 0$, where the initial state $s_0 \sim p_0$. Then, at time $t \geq 0$, the agent observes the environment state $s_t \in \mathcal{S}$, and follows a **policy** π to take an action $a_t \in \mathcal{A}$. In response, the environment emits a real-valued reward signal $r_t \in \mathcal{R}$ and enters a new state $s_{t+1} \in \mathcal{S}$. The policy is in general stochastic, with $\pi(a|s)$ being the probability of choosing action a in state s. We use $\pi(s)$ to denote the conditional probability over $\mathcal{A}$ if the policy is stochastic, or the action it chooses if it is deterministic. The process at every step is called a **transition**; at time t, it consists of the tuple (s_t, a_t, r_t, s_{t+1}), where $a_t \sim \pi(s_t)$, $s_{t+1} \sim p_T(s_t, a_t)$, and $r_t \sim p_R(s_t, a_t, s_{t+1})$. Hence, under policy π, the probability of generating a trajectory $\boldsymbol{\tau}$ of length T can be written explicitly as

$$p(\boldsymbol{\tau}) = p_0(s_0) \prod_{t=0}^{T-1} \pi(a_t|s_t) p_T(s_{t+1}|s_t, a_t) p_R(r_t|s_t, a_t, s_{t+1}) \tag{34.74}$$

It is useful to define the **reward function** from the reward model p_R, as the average immediate reward of taking action a in state s, with the next state marginalized:

$$R(s, a) \triangleq \mathbb{E}_{p_T(s'|s,a)} \left[\mathbb{E}_{p(r|s,a,s')} [r] \right] \tag{34.75}$$

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 34.11 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 $\boldsymbol{x}_t \in \mathcal{X} \subseteq \mathbb{R}^D$, actions are denoted by $\boldsymbol{u}_t \in \mathcal{U} \subseteq \mathbb{R}^K$, 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.

34.5.2 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 34.12 for an illustration. MDPs are a special case where $x_t = s_t$.

In general, POMDPs are much harder to solve than MDPs (see e.g., [KLC98]). A common approximation is to use the last several observed inputs, say $\boldsymbol{x}_{t-h:t}$ for history of size h, as a proxy for the hidden state, and then to treat this as a fully observed MDP.

34.5.3 Episodes and returns

The Markov decision process describes how a trajectory $\boldsymbol{\tau} = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$ is stochastically generated. If the agent can potentially interact with the environment forever, we call it a **continuing task**. Alternatively, the agent is in an **episodic task**, if its interaction terminates once the system enters a **terminal state** or **absorbing state**; s is absorbing if the next state from s is always s with 0 reward. After entering a terminal state, we may start a new **epsiode** from a new initial state $s_0 \sim p_0$. The episode length is in general random. For example, the amount of time a robot takes to

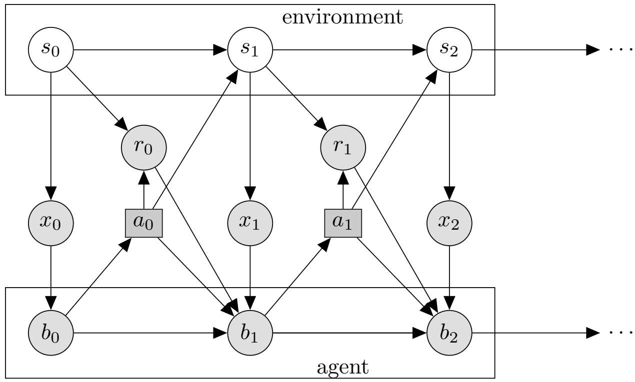


Figure 34.12: Illustration of a partially observable Markov decision process (POMDP) with hidden environment state s_t which generates the observation x_t, controlled by an agent with internal belief state b_t which generates the action a_t. The reward r_t depends on s_t and a_t. Nodes in this graph represent random variables (circles) and decision variables (squares).

reach its goal may be quite variable, depending on the decisions it makes, and the randomness in the environment. Note that we can convert an episodic MDP to a continuing MDP by redefining the transition model in absorbing states to be the initial-state distribution p_0. Finally, if the trajectory length T in an episodic task is fixed and known, it is called a **finite horizon problem**.

Let $\boldsymbol{\tau}$ be a trajectory of length T, where T may be ∞ if the task is continuing. We define the **return** for the state at time t to be the sum of expected rewards obtained going forwards, where each reward is multiplied by a **discount factor** $\gamma \in [0, 1]$:

$$G_t \triangleq r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{T-t-1} r_{T-1} \tag{34.76}$$

$$= \sum_{k=0}^{T-t-1} \gamma^k r_{t+k} = \sum_{j=t}^{T-1} \gamma^{j-t} r_j \tag{34.77}$$

G_t is sometimes called the **reward-to-go**. For episodic tasks that terminate at time T, we define $G_t = 0$ for $t \geq T$. Clearly, the return satisfies the following recursive relationship:

$$G_t = r_t + \gamma(r_{t+1} + \gamma r_{t+2} + \cdots) = r_t + \gamma G_{t+1} \tag{34.78}$$

The discount factor γ plays two roles. First, it ensures the return is finite even if $T = \infty$ (i.e., infinite horizon), provided we use $\gamma < 1$ and the rewards r_t are bounded. Second, it puts more weight on short-term rewards, which generally has the effect of encouraging the agent to achieve its goals more quickly (see Section 34.5.5.1 for an example). However, if γ is too small, the agent will become too greedy. In the extreme case where $\gamma = 0$, the agent is completely **myopic**, and only tries to maximize its immediate reward. In general, the discount factor reflects the assumption that there is a probability of $1 - \gamma$ that the interaction will end at the next step. For finite horizon problems, where T is known, we can set $\gamma = 1$, since we know the life time of the agent a priori.[4]

4. We may also use $\gamma = 1$ for continuing tasks, targeting the (undiscounted) average reward criterion [Put94].

34.5.4 Value functions

Let π be a given policy. We define the **state-value function**, or **value function** for short, as follows (with $\mathbb{E}_\pi[\cdot]$ indicating that actions are selected by π):

$$V_\pi(s) \triangleq \mathbb{E}_\pi[G_0|s_0 = s] = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s\right] \tag{34.79}$$

This is the expected return obtained if we start in state s and follow π 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:

$$Q_\pi(s, a) \triangleq \mathbb{E}_\pi[G_0|s_0 = s, a_0 = a] = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a\right] \tag{34.80}$$

This quantity represents the expected return obtained if we start by taking action a in state s, and then follow π to choose actions thereafter.

Finally, we define the **advantage function** as follows:

$$A_\pi(s, a) \triangleq Q_\pi(s, a) - V_\pi(s) \tag{34.81}$$

This tells us the benefit of picking action a in state s then switching to policy π, relative to the baseline return of always following π. 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: $V_\pi(s) = \mathbb{E}_{\pi(a|s)}[Q_\pi(s, a)]$.

34.5.5 Optimal value functions and policies

Suppose π_* is a policy such that $V_{\pi_*} \geq V_\pi$ for all $s \in \mathcal{S}$ and all policy π, 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**:

$$V_*(s) = \max_a R(s, a) + \gamma \mathbb{E}_{p_T(s'|s,a)}[V_*(s')] \tag{34.82}$$

$$Q_*(s, a) = R(s, a) + \gamma \mathbb{E}_{p_T(s'|s,a)}\left[\max_{a'} Q_*(s', a')\right] \tag{34.83}$$

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 (34.82) and (34.83) are called **Bellman error** or **Bellman residual**.

Furthermore, given the optimal value function, we can derive an optimal policy using

$$\pi_*(s) = \operatorname*{argmax}_a Q_*(s, a) \tag{34.84}$$

$$= \operatorname*{argmax}_a \left[R(s, a) + \gamma \mathbb{E}_{p_T(s'|s,a)}[V_*(s')]\right] \tag{34.85}$$

Following such an optimal policy ensures the agent achieves maximum expected return starting from any state. The problem of solving for V_*, Q_* or π_* is called **policy optimization**. In contrast, solving for V_π or Q_π for a given policy π 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 (34.82) and (34.83) with $\mathbb{E}_{\pi(a|s)}[\cdot]$.

In Equations (34.84) and (34.85), as in the Bellman optimality equations, we must take a maximum over all actions in $\mathcal{A}$, and the maximizing action is called the **greedy action** with respect to the value functions, Q_* or V_*. Finding greedy actions is computationally easy if $\mathcal{A}$ is a small finite set. For high dimensional continuous spaces, we can treat a as a sequence of actions, and optimize one dimension at a time [Met+17], or use gradient-free optimizers such as cross-entropy method (Section 6.7.5), as used in the **QT-Opt** method [Kal+18a]. Recently, **CAQL** (continuous action Q-learning, [Ryu+20]) proposed to use mixed integer programming to solve the argmax problem, leveraging the ReLU structure of the Q-network. We can also amortize the cost of this optimization by training a policy $a_* = \pi_*(s)$ after learning the optimal Q-function.

34.5.5.1 Example

In this section, we show a simple example, to make concepts like value functions more concrete. Consider the 1d **grid world** shown in Figure 34.13(a). There are 5 possible states, among them S_{T1} and S_{T2} are absorbing states, since the interaction ends once the agent enters them. There are 2 actions, $\uparrow$ and $\downarrow$. The reward function is zero everywhere except at the goal state, S_{T2}, which gives a reward of 1 upon entering. Thus the optimal action in every state is to move down.

Figure 34.13(b) shows the Q_* function for $\gamma = 0$. Note that we only show the function for non-absorbing states, as the optimal Q-values are 0 in absorbing states by definition. We see that $Q_*(s_3, \downarrow) = 1.0$, since the agent will get a reward of 1.0 on the next step if it moves down from s_3; however, $Q_*(s, a) = 0$ for all other state-action pairs, since they do not provide nonzero immediate reward. This optimal Q-function reflects the fact that using $\gamma = 0$ is completely myopic, and ignores the future.

Figure 34.13(c) shows Q_* when $\gamma = 1$. In this case, we care about all future rewards equally. Thus $Q_*(s, a) = 1$ for all state-action pairs, since the agent can always reach the goal eventually. This is infinitely far-sighted. However, it does not give the agent any short-term guidance on how to behave. For example, in s_2, it is not clear if it is should go up or down, since both actions will eventually reach the goal with identical Q_*-values.

Figure 34.13(d) shows Q_* when $\gamma = 0.9$. This reflects a preference for near-term rewards, while also taking future reward into account. This encourages the agent to seek the shortest path to the goal, which is usually what we desire. A proper choice of γ is up to the agent designer, just like the design of the reward function, and has to reflect the desired behavior of the agent.

34.6 Planning in an MDP

In this section, we discuss how to compute an optimal policy when the MDP model is known. This problem is called **planning**, in contrast to the learning problem where the models are unknown, which is tackled using reinforcement learning Chapter 35. The planning algorithms we discuss are based on **dynamic programming** (DP) and **linear programming** (LP).

For simplicity, in this section, we assume discrete state and action sets with $\gamma < 1$. However, exact

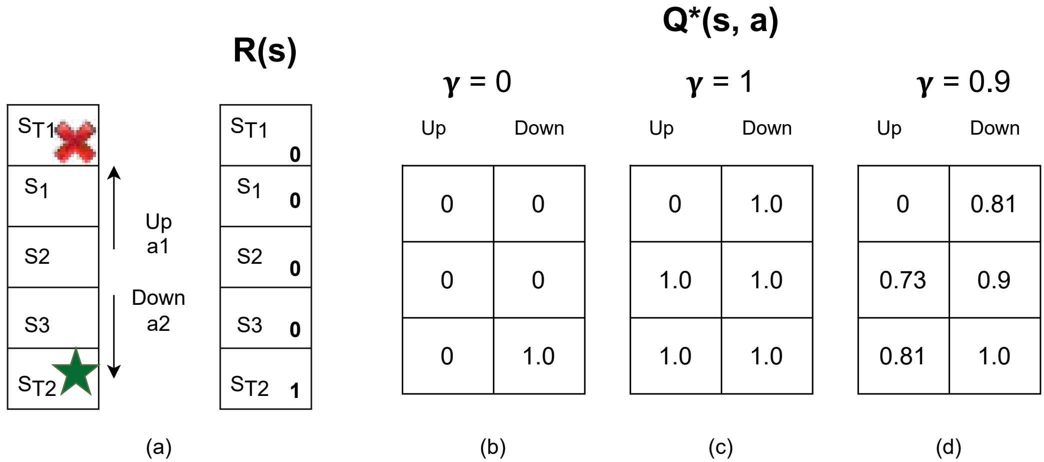


Figure 34.13: Left: illustration of a simple MDP corresponding to a 1d grid world of 3 non-absorbing states and 2 actions. Right: optimal Q-functions for different values of γ. Adapted from Figures 3.1, 3.2, 3.4 of [GK19].

calculation of optimal policies often depends polynomially on the sizes of $\mathcal{S}$ and $\mathcal{A}$, and is intractable, for example, when the state space is a Cartesian product of several finite sets. This challenge is known as the **curse of dimensionality**. Therefore, approximations are typically needed, such as using parametric or nonparametric representations of the value function or policy, both for computational tractability and for extending the methods to handle MDPs with general state and action sets. In this case, we have **approximate dynamic programming** (ADP) and **approximate linear programming** (ALP) algorithms (see e.g., [Ber19]).

34.6.1 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

$$V_{k+1}(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} p(s'|s,a) V_k(s') \right] \tag{34.86}$$

Note that the update rule, sometimes called a **Bellman backup**, is exactly the right-hand side of the Bellman optimality equation Equation (34.82), with the unknown V_* replaced by the current estimate V_k. A fundamental property of Equation (34.86) is that the update is a **contraction**: it can be verified that

$$\max_s |V_{k+1}(s) - V_*(s)| \leq \gamma \max_s |V_k(s) - V_*(s)| \tag{34.87}$$

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 (34.85). 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 34.14(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 transition matrix $p_T(s'|s,a)$ where taking edge a from node s leads to the neighboring node s' with probability 1. The reward function is defined as $R(s,a) = -1$ for all states s except the goal states, which are modeled as absorbing states.

In problems such as this, we can use a method known as **real-time dynamic programming** (RTDP) [BBS95], to efficiently compute an **optimal partial policy**, which only specifies what to do for the reachable states. RTDP maintains a value function estimate V. At each step, it performs a Bellman backup for the current state s by $V(s) \leftarrow \max_a \mathbb{E}_{p_T(s'|s,a)}\left[R(s,a) + \gamma V(s')\right]$. It can picks an action a (often with some exploration), reaches a next state s', and repeats the process. This can be seen as a form of the more general **asynchronous value iteration**, that focuses its computational effort on parts of the state space that are more likely to be reachable from the current state, rather than synchronously updating all states at each iteration.

34.6.2 Policy iteration

Another effective DP method for computing π_* is **policy iteration**. It is an iterative algorithm that searches in the space of deterministic policies until converging to an optimal policy. Each iteration consists of two steps, **policy evaluation** and **policy improvement**.

The policy evaluation step, as mentioned earlier, computes the value function for the current policy. Let π represent the current policy, $\boldsymbol{v}(s) = V_\pi(s)$ represent the value function encoded as a vector indexed by states, $\boldsymbol{r}(s) = \sum_a \pi(a|s)R(s,a)$ represent the reward vector, and $\mathbf{T}(s'|s) = \sum_a \pi(a|s)p(s'|s,a)$ represent the state transition matrix. Bellman's equation for policy evaluation can be written in the matrix-vector form as

$$\boldsymbol{v} = \boldsymbol{r} + \gamma \mathbf{T}\boldsymbol{v} \tag{34.88}$$

This is a linear system of equations in $|\mathcal{S}|$ unknowns. We can solve it using matrix inversion: $\boldsymbol{v} = (\mathbf{I} - \gamma\mathbf{T})^{-1}\boldsymbol{r}$. Alternatively, we can use value iteration by computing $\boldsymbol{v}_{t+1} = \boldsymbol{r} + \gamma\mathbf{T}\boldsymbol{v}_t$ until near convergence, or some form of asynchronous variant that is computationally more efficient.

Once we have evaluated V_π for the current policy π, we can use it to derive a better policy π', thus the name policy improvement. To do this, we simply compute a deterministic policy π' that acts greedily with respect to V_π in every state; that is, $\pi'(s) = \operatorname{argmax}_a\{R(s,a) + \gamma\mathbb{E}\left[V_\pi(s')\right]\}$. We can guarantee that $V_{\pi'} \geq V_\pi$. To see this, define $\boldsymbol{r}'$, $\mathbf{T}'$ and $\boldsymbol{v}'$ as before, but for the new policy π'. The definition of π' implies $\boldsymbol{r}' + \gamma\mathbf{T}'\boldsymbol{v} \geq \boldsymbol{r} + \gamma\mathbf{T}\boldsymbol{v} = \boldsymbol{v}$, where the equality is due to Bellman's equation. Repeating the same equality, we have

$$\boldsymbol{v} \leq \boldsymbol{r}' + \gamma\mathbf{T}'\boldsymbol{v} \leq \boldsymbol{r}' + \gamma\mathbf{T}'(\boldsymbol{r}' + \gamma\mathbf{T}'\boldsymbol{v}) \leq \boldsymbol{r}' + \gamma\mathbf{T}'(\boldsymbol{r}' + \gamma\mathbf{T}'(\boldsymbol{r}' + \gamma\mathbf{T}'\boldsymbol{v})) \leq \cdots \tag{34.89}$$

$$= (\mathbf{I} + \gamma\mathbf{T}' + \gamma^2\mathbf{T}'^2 + \cdots)\boldsymbol{r} = (\mathbf{I} - \gamma\mathbf{T}')^{-1}\boldsymbol{r} = \boldsymbol{v}' \tag{34.90}$$

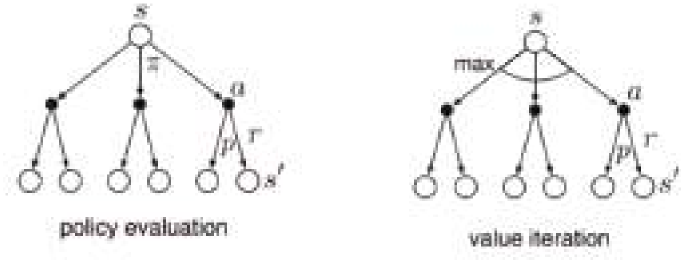


Figure 34.14: Policy iteration vs value iteration represented as backup diagrams. Empty circles represent states, solid (filled) circles represent states and actions. Adapted from Figure 8.6 of [SB18].

Starting from an initial policy π_0, policy iteration alternates between policy evaluation (E) and improvement (I) steps, as illustrated below:

$$\pi_0 \xrightarrow{E} V_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V_{\pi_1} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} V_* \tag{34.91}$$

The algorithm stops at iteration k, if the policy π_k is greedy with respect to its own value function V_{π_k}. In this case, the policy is optimal. Since there are at most $|\mathcal{A}|^{|\mathcal{S}|}$ deterministic policies, and every iteration strictly improves the policy, the algorithm must converge after finite iterations.

In PI, we alternate between policy evaluation (which involves multiple iterations, until convergence of V_π), 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_π whereas value iteration computes V_*. This difference is illustrated in Figure 34.14, 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.

34.6.3 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 35.5). The primal form of LP is given by

$$\min_V \sum_s p_0(s)V(s) \quad \text{s.t.} \quad V(s) \geq R(s,a) + \gamma \sum_{s'} p_T(s'|s,a)V(s), \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A} \tag{34.92}$$

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 (34.92) 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:

$$\max_{d \geq 0} \sum_{s,a} d(s,a) R(s,a) \quad \text{s.t.} \quad \sum_{a} d(s,a) = (1-\gamma) p_0(s) + \gamma \sum_{\bar{s},\bar{a}} p_T(s|\bar{s},\bar{a}) d(\bar{s},\bar{a}) \ \ \forall s \in \mathcal{S} \tag{34.93}$$

Any nonnegative d satisfying the constraint above is the **normalized occupancy distribution** of some corresponding policy $\pi_d(a|s) \triangleq d(s,a) / \sum_{a'} d(s,a')$: [5]

$$d(s,a) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t p\big(s_t = s, a_t = a | s_0 \sim p_0, a_t \sim \pi_d(s_t)\big) \tag{34.94}$$

The constant $(1-\gamma)$ normalizes d to be a valid distribution, so that it sums to unity. With this interpretation of d, the objective in Equation (34.93) is just the average per-step reward under the normalized occupancy distribution. Once an optimal solution d_* is found, an optimal policy can be immediately obtained by $\pi_*(a|s) = d_*(s,a) / \sum_{a'} d_*(s,a')$.

A challenge in solving the primal or dual LPs for MDPs is the large number of constraints and variables. Approximations are needed, where the variables are parameterized (either linearly or nonlinearly), and the constraints are sampled or approximated (see e.g., [dV04; LBS17; CLW18]).

34.7 Active learning

This section is coauthored with Zeel B Patel.

In this section, we discuss **active learning** (AL), in which the agent gets to choose which data it wants to use so as to learn the underlying predictive function as quickly as possible, i.e., using the smallest amount of labeled data. This can be much more efficient than using randomly collected data, as illustrated in Figure 34.15. This is useful when labels are expensive to collect, e.g., for medical image classification [GIG17; Wal+20].

There are many approaches to AL, as reviewed in [Set12; Ren+21; ZSH22]. In this section, we just consider a few methods.

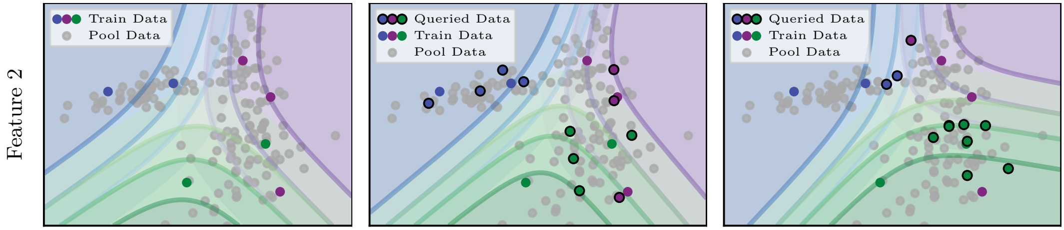
34.7.1 Active learning scenarios

One of the earliest AL methods is known as **membership query synthesis** [Ang88]. In this scenario the agent can generate an arbitrary query $\boldsymbol{x} \sim p(\boldsymbol{x})$ and then ask the oracle for its label, $y = f(\boldsymbol{x})$. (An "oracle" is the term given to a system that knows the true answer to every possible question.) This scenario is mostly of theoretical interest, since it is hard to learn good generative models, and it is rarely possible to have access to an oracle on demand (although human-power **crowd computing** platforms can be considered as oracles with high latency).

Another scenario is **stream-based selective sampling** [ACL89], where the agent receives a stream of inputs, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$, and at each step must decide whether to request the label or not. Again, this scenario is mostly of theoretical interest.

The last and widely used setting for machine learning is **pool-based-sampling** [LG94], where the pool of unlabeled samples $\mathcal{X}$ is available from the beginning. At each step we apply an **acquisition**

5. 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.



(a) (b) (c)

Figure 34.15: Decision boundaries for a logistic regression model applied to a 2-dimensional, 3-class dataset. (a) Results after fitting the model on the initial training data; the test accuracy is 0.818. (b) results after further training on 11 randomly sampled points; accuracy is 0.848. (c) Results after further training on 11 points chosen with margin sampling (see Section 34.7.3); accuracy is 0.969. Generated by active_learning_visualization_class.ipynb.

Problem	Goal	Action space
Active learning	$\text{argmin}_f \, \mathbb{E}_{p(\boldsymbol{x})} [\ell(f^*(\boldsymbol{x}), f(\boldsymbol{x}))]$	choose $\boldsymbol{x}$ at which to get $\boldsymbol{y} = f^*(\boldsymbol{x})$
Bayesian optimization	$\text{argmax}_{\boldsymbol{x} \in \mathcal{X}} \, f^*(\boldsymbol{x})$	choose $\boldsymbol{x}$ at which to evaluate $f^*(\boldsymbol{x})$
Contextual bandits	$\text{argmax}_\pi \, \mathbb{E}_{p(\boldsymbol{x})\pi(a\|\boldsymbol{x})} [R^*(\boldsymbol{x}, a)]$	choose a at which to evaluate $R^*(\boldsymbol{x}, a)$

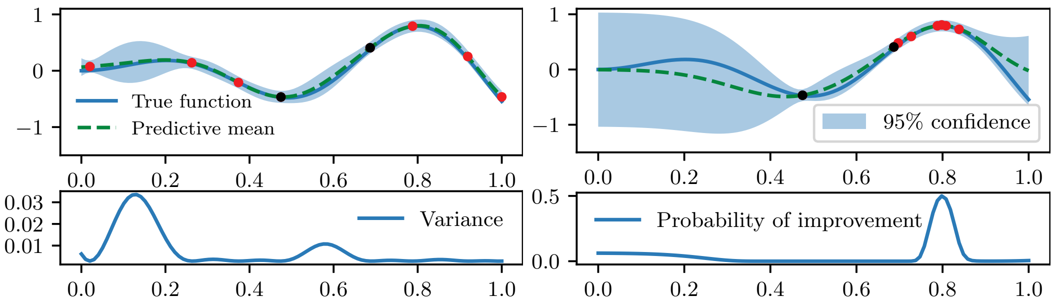
Table 34.1: Comparison among active learning, Bayesian optimization, and contextual bandits in terms of goal and action space.

function to each candidate in the batch, to decide which one to collect the label for. We then collect the label, update the model with the new data, and repeat the process until we exhaust the pool, run out of time, or reach some desired performance. In the subsequent sections, we will focus only on pool-based sampling.

34.7.2 Relationship to other forms of sequential decision making

(Pool-based) active learning is closely related to Bayesian optimization (BO, Section 6.6) and contexual bandit problems (Section 34.4). The connections are discussed at length [Tou14], but in brief, the methods differ because they solve slightly different objective functions, as summarized in Table 34.1. In particular, in active learning, our goal is to identify a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that will incur minimum expected loss when applied to random inputs $\boldsymbol{x}$; in BO, our goal is to identify an input point $\boldsymbol{x}$ where the function output $f(\boldsymbol{x})$ is maximal; and in bandits, our goal is to identify a policy $\pi : \mathcal{X} \rightarrow \mathcal{A}$ that will give maximum expected reward when applied to random inputs (contexts) $\boldsymbol{x}$. (We see that the goal in AL and bandits is similar, but in bandits the agent only gets to choose the action, not the state, so only has partial control over where the (reward) function is evaluated.)

In all three problems, we want to find the optimum with as few actions as possible, so we have to solve the exploration-exploitation problem (Section 34.4.3). One approach is to represent our uncertainty about the function using a method such as a Gaussian process (Chapter 18), which lets



(a) Active learning (b) Bayesian optimization

Figure 34.16: Active learning vs Bayesian optimization. Active learning tries to approximate the true function well. Bayesian optimization tries to find maximum value of the true function. Initial and queried points are denoted as black and red dots respectively. Generated by bayes_opt_vs_active_learning.ipynb.

us compute $p(f|\mathcal{D}_{1:t})$. We then define some acquisition function $\alpha(\boldsymbol{x})$ that evaluates how useful it would be to query the function at input location $\boldsymbol{x}$, given the belief state $p(f|\mathcal{D}_{1:t})$ and we pick as our next query $\boldsymbol{x}_{t+1} = \operatorname{argmax}_{\boldsymbol{x}} \alpha(\boldsymbol{x})$. (In the bandit setting, the agent does not get to choose the state $\boldsymbol{x}$, but does get to choose action a.) For example, in BO, it is common to use probability of improvement (Section 6.6.3.1), and for AL of a regression task, we can use the posterior predictive variance. The objective for AL will cause the agent to query "all over the place", whereas for BO, the agent will "zoom in" on the most promising regions, as shown in Figure 34.16. We discuss other acquisition functions for AL in Section 34.7.3.

34.7.3 Acquisition strategies

In this section, we discuss some common AL heuristics for choosing which points to query.

34.7.3.1 Uncertainty sampling

An intuitive heuristic for choosing which example to label next is to pick the one for which the model is currently most uncertain. This is called **uncertainty sampling**. We already illustrated this in the case of regression in Figure 34.16, where we represented uncertainty in terms of the posterior variance.

For classification problems, we can measure uncertainty in various ways. Let $\boldsymbol{p}_n = [p(y = c|\boldsymbol{x}_n)]_{c=1}^C$ be the vector of class probabilities for each unlabeled input $\boldsymbol{x}_n$. Let $U_n = \alpha(\boldsymbol{p}_n)$ be the uncertainty for example n, where α is an acquisition function. Some common choices for α are: **entropy sampling** [SW87a], which uses $\alpha(\boldsymbol{p}) = -\sum_{c=1}^C p_c \log p_c$; **margin sampling**, which uses $\alpha(\boldsymbol{p}) = p_2 - p_1$, where p_1 is the probability of the most probable class, and p_2 is the probability of the second most probable class; and **least confident sampling**, which uses $\alpha(\boldsymbol{p}) = 1 - p_{c^*}$, where $c^* = \operatorname{argmax}_c p_c$. The difference between these strategies is shown in Figure 34.17. In practice it is often found that margin sampling works the best [Chu+19].

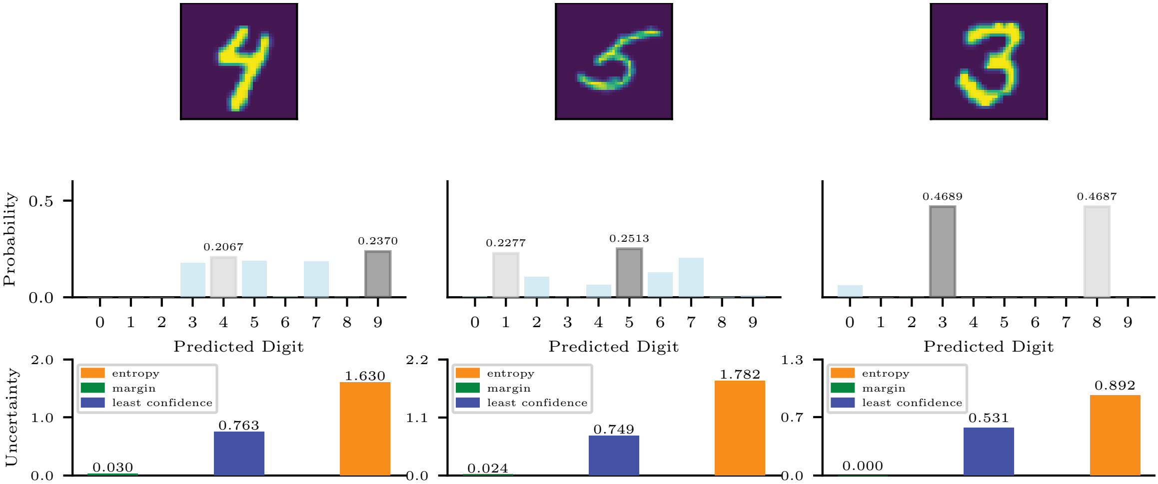


Figure 34.17: Outputs of a logistic regression model fit on some training points, and then applied to 3 candidate query inputs. We show the predicted probabilites for each class label. The highlighted dark gray is the max probability, the light gray bar is the 2nd highest probability. The least confident scores for the 3 inputs are: $1 - 0.23 = 0.76$, $1 - 0.25 = 0.75$, *and* $1 - 0.47 = 0.53$, *so we pick the first query. The entropy scores are: 1.63, 1.78 and 0.89, so we pick the second query. The margin scores are:* $0.237 - 0.2067 = 0.0303$, $0.2513 - 0.2277 = 0.0236$, *and* $0.4689 - 0.4687 = 0.0002$, *so we pick the third query. Generated by active_learning_comparison_mnist.ipynb.*

34.7.3.2 Query by committee

In this section, we discuss how to apply uncertainty sampling to models, such as support vector machines (SVMs), that only return a point prediction rather than a probability distribution. The basic approach is to create an ensemble of diverse models, and to use disagreement between the model predictions as a form of uncertainty. (This can be useful even for probabilistic models, such as DNNs, since model uncertainty can often be larger than parametric uncertainty, as we discuss in the section on deep ensembles, Section 17.3.9.)

In more detail, suppose we have K ensemble members, and let c_n^k be the predicted class from member k on input $\boldsymbol{x}_n$. Let $v_{nc} = \sum_{k=1}^{K} \mathbb{I}\left(c_n^k = c\right)$ be the number of votes cast for class c, and $q_{nc} = v_{nc}/C$ be the induced distribution. (A similar method can be used for regression models, where we use the standard deviation of the prediction across the members.) We can then use margin sampling or entropy sampling with distribution $\boldsymbol{q}_n$. This approach is called **query by committee (QBC)** [SOS92], and can often out-perform vanilla uncertainty sampling with a single model, as we show in Figure 34.18.

34.7.3.3 Information theoretic methods

A natural acquisition strategy is to pick points whose labels will maximimally reduce our uncertainty about the model parameters $\boldsymbol{w}$. This is known as the **information gain** criterion, and was first

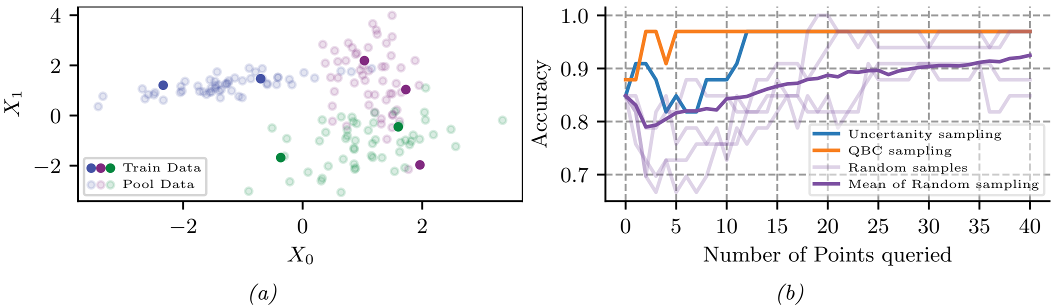


Figure 34.18: (a) Random forest (RF) classifier applied to a 2-dimensional, 3-class dataset. (b) Accuracy vs number of query points for margin sampling vs random sampling. We represent uncertainty using either a single RF (based on the predicted distribution over labels induced by the trees in the forest), or a committee containing an RF and a logistic regression model. Generated by active_learning_compare_class.ipynb.

proposed in [Lin56]. It is defined as follows:

$$\alpha(\boldsymbol{x}) \triangleq \mathbb{H}\left(p(\boldsymbol{w}|\mathcal{D})\right) - \mathbb{E}_{p(y|\boldsymbol{x},\mathcal{D})}\left[\mathbb{H}\left(p(\boldsymbol{w}|\mathcal{D},\boldsymbol{x},y)\right)\right] \tag{34.95}$$

(Note that the first term is a constant wrt $\boldsymbol{x}$, but we include it for later convenience.) This is equivalent to the expected change in the posterior over the parameters which is given by

$$\alpha'(\boldsymbol{x}) \triangleq \mathbb{E}_{p(y|\boldsymbol{x},\mathcal{D})}\left[D_{\mathbb{KL}}\left(p(\boldsymbol{w}|\mathcal{D},\boldsymbol{x},y) \parallel p(\boldsymbol{w}|\mathcal{D})\right)\right] \tag{34.96}$$

Using symmetry of the mutual information, we can rewrite Equation (34.95) as follows:

$$\begin{aligned}
\alpha(\boldsymbol{x}) &= \mathbb{H}\left(\boldsymbol{w}|\mathcal{D}\right) - \mathbb{E}_{p(y|\boldsymbol{x},\mathcal{D})}\left[\mathbb{H}\left(\boldsymbol{w}|\mathcal{D},\boldsymbol{x},y\right)\right] && (34.97)\\
&= \mathbb{I}(\boldsymbol{w},y|\mathcal{D},\boldsymbol{x}) && (34.98)\\
&= \mathbb{H}\left(y|\boldsymbol{x},\mathcal{D}\right) - \mathbb{E}_{p(\boldsymbol{w}|\mathcal{D})}\left[\mathbb{H}\left(y|\boldsymbol{x},\boldsymbol{w},\mathcal{D}\right)\right] && (34.99)
\end{aligned}$$

The advantage of this approach is that we now only have to reason about the uncertainty of the predictive distribution over outputs y, not over the parameters $\boldsymbol{w}$. This approach is called **Bayesian active learning by disagreement** or **BALD** [Hou+12].

Equation (34.99) has an interesting interpretation. The first term prefers examples $\boldsymbol{x}$ for which there is uncertainty in the predicted label. Just using this as a selection criterion is equivalent to uncertainty sampling, which we discussed above. However, this can have problems with examples which are inherently ambiguous or mislabeled. By adding the second term, we penalize such behavior, since we add a large negative weight to points whose predictive distribution is entropic even when we know the parameters. Thus we ignore aleatoric (intrinsic) uncertainty and focus on epistemic uncertainty.

34.7.4 Batch active learning

In many applications, we need to select a batch of unlabeled examples at once, since training a model on single examples is too slow. This is called **batch active learning**. The key challenge is that we need to ensure the different queries that we request are diverse, so we maximize the information gain. Various methods for this problem have been devised; here we focus on the **BatchBALD** method of [KAG19], which extends the BALD method of Section 34.7.3.3.

34.7.4.1 BatchBALD

The naive way to extend the BALD score to a batch of b candidate query points is to define

$$\alpha_{\text{BALD}}(\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_B\}, p(\boldsymbol{w}|\mathcal{D})) = \alpha_{\text{BALD}}(\boldsymbol{x}_{1:B}, p(\boldsymbol{w}|\mathcal{D})) = \sum_{i=1}^{B} \mathbb{I}(y_i; \boldsymbol{w}|\boldsymbol{x}_i, \mathcal{D}) \tag{34.100}$$

However this may pick points that are quite similar in terms of their information content. In BatchBALD, we use joint conditional mutual information between the set of labels and the parameters:

$$\alpha_{\text{BBALD}}(\boldsymbol{x}_{1:B}, p(\boldsymbol{w}|\mathcal{D})) = \mathbb{I}(\boldsymbol{y}_{1:B}; \boldsymbol{w}|\boldsymbol{x}_{1:B}, \mathcal{D}) = \mathbb{H}(\boldsymbol{y}_{1:b}|\boldsymbol{x}_{1:B}, \mathcal{D}) - \mathbb{E}_{p(\boldsymbol{w}|\mathcal{D})}\left[\mathbb{H}(\boldsymbol{y}_{1:B}|\boldsymbol{x}_{1:B}, \boldsymbol{w}, \mathcal{D})\right] \tag{34.101}$$

To understand how this differs from BALD, we will use information diagrams for representing MI in terms of Venn diagrams, as explained in Section 5.3.2. In particular, [Yeu91a] showed that we can define a signed measure, μ^*, for discrete random variables x and y such that $\mathbb{I}(x; y) = \mu^*(x \cap y)$, $\mathbb{H}(x, y) = \mu^*(x \cup y)$, $\mathbb{E}_{p(y)}[\mathbb{H}(x|y)] = \mu^*(x \setminus y)$, etc. Using this, we can interpret standard BALD as the sum of the individual intersections, $\sum_i \mu^*(y_i \cap \boldsymbol{w})$, which double counts overlaps between the y_i, as shown in Figure 34.19(a). By contrast, BatchBALD takes overlap into account by computing

$$\mathbb{I}(\boldsymbol{y}_{1:B}; \boldsymbol{w}|\boldsymbol{x}_{1:B}, \mathcal{D}) = \mu^*(\cup_i y_i \cap \boldsymbol{w}) = \mu^*(\cup_i y_i) - \mu^*(\cup_i y_i \setminus \boldsymbol{w}) \tag{34.102}$$

This is illustrated in Figure 34.19(b). From this, we can see that $\alpha_{\text{BBALD}} \leq \alpha_{\text{BALD}}$. Indeed, one can show[6]

$$\mathbb{I}(\boldsymbol{y}_{1:B}, \boldsymbol{w}|\boldsymbol{x}_{1:B}, \mathcal{D}) = \sum_{i=1}^{B} \mathbb{I}(y_i, \boldsymbol{w}|\boldsymbol{x}_{1:B}, \mathcal{D}) - \mathbb{TC}(\boldsymbol{y}_{1:B}|\boldsymbol{x}_{1:B}, \mathcal{D}) \tag{34.103}$$

where TC is the total correlation (see Section 5.3.5.1).

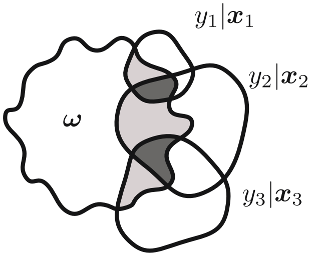
34.7.4.2 Optimizing BatchBALD

To avoid the combinatorial explosion that arises from jointly scoring subsets of points, we can use a a greedy approximation for computing BatchBALD one point at a time. In particular, suppose at step $n-1$ we already have a partial batch $\mathcal{A}_{n-1}$. The next point is chosen using

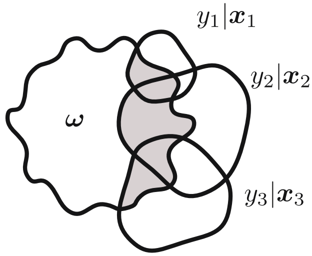
$$\boldsymbol{x}_n = \underset{\boldsymbol{x} \in \mathcal{D}_{\text{pool}} \setminus \mathcal{A}_{n-1}}{\operatorname{argmax}} \alpha_{\text{BBALD}}(\mathcal{A}_{n-1} \cup \{\boldsymbol{x}\}, p(\boldsymbol{w}|\mathcal{D})) \tag{34.104}$$

We then add $\boldsymbol{x}_n$ to $\mathcal{A}_{n-1}$ to get $\mathcal{A}_n$. Fortunately the BatchBALD acquisition function is submodular, as shown in [KAG19]. Hence this greedy algorithm is within $1 - 1/e \approx 0.63$ of optimal (see Section 6.9.4.1).

6. See http://blog.blackhc.net/2022/07/kbald/



(a) BALD



(b) BatchBALD

Figure 34.19: Intuition behind BALD and BatchBALD. D_{pool} is an unlabelled dataset (from which $\boldsymbol{x}_{1:b}$ are taken) , D_{train} is the current training set, $\boldsymbol{w}$ is set of model parameters, $p(y|\boldsymbol{x}, \boldsymbol{w}, D_{train})$ are output predictions for datapoint $\boldsymbol{x}$. BALD overestimates the joint mutual information whereas BatchBALD takes the overlap between variables into account. Areas contributing to the respective score are shown in grey, and areas that are double-counted in dark grey. From Figure 3 of [KAG19]. Used with kind permission of Andreas Kirsch.

34.7.4.3 Computing BatchBALD

Computing the joint (conditional) mutual information is intractable, so in this section, we discuss how to approximate it. For brevity we drop the conditioning on $\boldsymbol{x}$ and $\mathcal{D}$. With this new notation, the objective becomes

$$\alpha_{\text{BBALD}}(\boldsymbol{x}_{1:B}, p(\boldsymbol{w}|\mathcal{D})) = \mathbb{H}(y_1, \ldots, y_B) - \mathbb{E}_{p(\boldsymbol{w})}\left[\mathbb{H}(y_1, \ldots, y_B|\boldsymbol{w})\right] \tag{34.105}$$

Note that the y_i are conditionally independent given $\boldsymbol{w}$, so $\mathbb{H}(y_1, \ldots, y_B|\boldsymbol{w}) = \sum_{i=1}^{B} \mathbb{H}(y_i|\boldsymbol{w})$. Hence we can approximate the second term with Monte Carlo:

$$\mathbb{E}_{p(\boldsymbol{w})}\left[\mathbb{H}(y_1, \ldots, y_B|\boldsymbol{w})\right] \approx \frac{1}{S}\sum_{i=1}^{n}\sum_{s} \mathbb{H}(y_i|\hat{\boldsymbol{w}}_s) \tag{34.106}$$

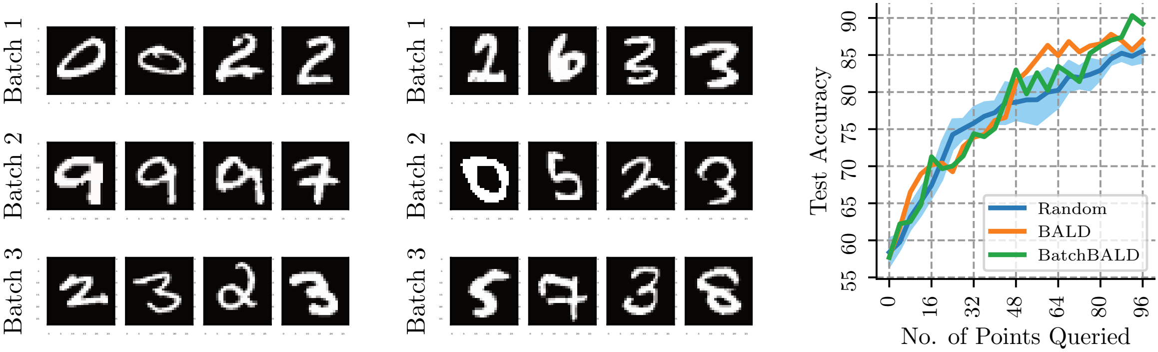
where $\hat{\boldsymbol{w}}_s \sim p(\boldsymbol{w}|\mathcal{D})$.

The first term, $\mathbb{H}(y_1, \ldots, y_B)$, is a joint entropy, so is harder to compute. [KAG19] propose the following approximation, summing over all possible label sequences in the batch, and leveraging the fact that $p(\boldsymbol{y}) = \mathbb{E}_{p(\boldsymbol{w})}\left[p(\boldsymbol{y}|\boldsymbol{w})\right]$:

$$\mathbb{H}(\boldsymbol{y}_{1:B}) = \mathbb{E}_{p(\boldsymbol{w})p(\boldsymbol{y}_{1:B}|vw)}\left[-\log p(\boldsymbol{y}_{1:B}|\boldsymbol{w})\right] \tag{34.107}$$

$$\approx \sum_{\hat{\boldsymbol{y}}_{1:B}} \left(\frac{1}{S}\sum_{s=1}^{S} p(\hat{\boldsymbol{y}}_{1:B}|\hat{\boldsymbol{w}}_s)\right) \log\left(\frac{1}{S}\sum_{s=1}^{S} p(\hat{\boldsymbol{y}}_{1:B}|\hat{\boldsymbol{w}}_s)\right) \tag{34.108}$$

The sum over all possible labels sequences can be made more efficient by noting that $p(\boldsymbol{y}_{1:n}|\boldsymbol{w}) = p(y_n|\boldsymbol{w})p(\boldsymbol{y}_{1:n-1}|\boldsymbol{w})$, so when we implement the greedy algorithm, we can incrementally update the probabilities, reusing previous computations. See [KAG19] for the details.



(a) *(b)* *(c)*

Figure 34.20: Three batches (each of size 4) queried from the MNIST pool by (a) BALD and (b) BatchBALD. (c) Plot of accuracy vs number of points queried. BALD may select replicas of single informative datapoint while BatchBALD selects diverse points, thus increasing data efficiency. Generated by batch_ bald_ mnist.ipynb.

34.7.4.4 Experimental comparison of BALD vs BatchBALD on MNIST

In this section, we show some experimental results applying BALD and BatchBALD to train a CNN on the standard MNIST dataset. We use a batch size of 4, and approximate the posterior over parameters $p(\boldsymbol{w}|\mathcal{D})$ using MC dropout (Section 17.3.1). In Figure 34.20(a), we see that BALD selects examples that are very similar to each other, whereas in Figure 34.20(b), we see that BatchBALD selects a greater diversity of points. In Figure 34.20(c), we see that BatchBALD results in more efficient learning than BALD, which in turn is more efficient than randomly sampling data.

35 Reinforcement learning

This chapter is co-authored with Lihong Li.

35.1 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 $\boldsymbol{\tau} = (s_0, a_0, r_0, s_1, a_1, r_1, s_2, \ldots, s_T)$ be a trajectory of length T, consisting of a sequence of states s_t, actions a_t, and rewards r_t.[1] 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 34.1.3), but note other possibilities such as **conditional value at risk**[2] that can be more appropriate in risk-sensitive applications.

We will focus on the Markov decision process, where the generative model for the trajectory $\boldsymbol{\tau}$ can be factored into single-step models. When these model parameters are known, solving for an optimal policy is called **planning** (see Section 34.6); 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. See Figure 35.1 for an overview. 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; Aga+22], and reviews such as [WO12; Aru+17; FL+18; Li18].

35.1.1 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 34.4). Our choices of notation are 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.

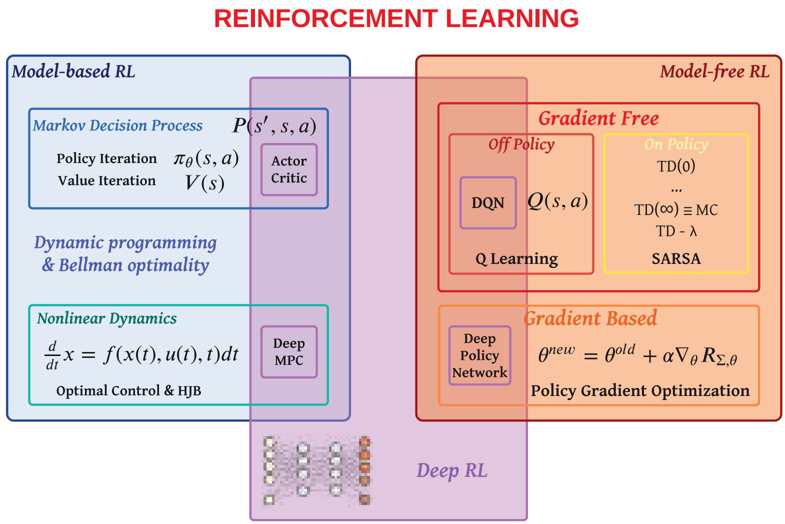


Figure 35.1: Overview of RL methods. Abbreviations: DQN = Deep Q network (Section 35.2.6); MPC = Model Predictive Control (Section 35.4); HJB = Hamilton-Jacobi-Bellman equation; TD = temporal difference learning (Section 35.2.2). Adapted from a slide by Steve Brunton.

Method	Functions learned	On/Off	Section
SARSA	$Q(s, a)$	On	Section 35.2.4
Q-learning	$Q(s, a)$	Off	Section 35.2.5
REINFORCE	$\pi(a\|s)$	On	Section 35.3.2
A2C	$\pi(a\|s)$, $V(s)$	On	Section 35.3.3.1
TRPO/PPO	$\pi(a\|s)$, $A(s, a)$	On	Section 35.3.4
DDPG	$a = \pi(s)$, $Q(s, a)$	Off	Section 35.3.5
Soft actor-critic	$\pi(a\|s)$, $Q(s, a)$	Off	Section 35.6.1
Model-based RL	$p(s'\|s, a)$	Off	Section 35.4

Table 35.1: Summary of some popular methods for RL. On/off refers to on-policy vs off-policy methods.

trajectories. This is the core focus of RL. We will go into more details into later sections, but first provide this roadmap.

We may categorize RL methods by the quantity the agent represents and learns: value function, policy, and model; or by how actions are selected: on-policy (actions must be selected by the agent's current policy), and off-policy. Table 35.1 lists a few representative examples. More details are given in the subsequent sections. We will also discuss at greater depth two important topics of off-policy learning and inference-based control in Sections 35.5 and 35.6.

35.1.2 Value-based methods

In a value-based method, we often try to learn the optimal Q-function from experience, and then derive a policy from it using Equation (34.84). Typically, a function approximator (e.g., a neural network), $Q_{\boldsymbol{w}}$, is used to represent the Q-function, which is trained iteratively. Given a transition (s, a, r, s'), we define the **temporal difference** (also called the **TD error**) as

$$r + \gamma \max_{a'} Q_{\boldsymbol{w}}(s', a') - Q_{\boldsymbol{w}}(s, a)$$

Clearly, the expected TD error is the Bellman error evaluated at (s, a). Therefore, if $Q_{\boldsymbol{w}} = Q_*$, the TD error is 0 on average by Bellman's optimality equation. Otherwise, the error provides a signal for the agent to change $\boldsymbol{w}$ to make $Q_{\boldsymbol{w}}(s, a)$ closer to $R(s, a) + \gamma \max_{a'} Q_{\boldsymbol{w}}(s', a')$. The update on $Q_{\boldsymbol{w}}$ is based on a target that is computed using $Q_{\boldsymbol{w}}$. This kind of update is known as **bootstrapping** in RL, and should not be confused with the statistical bootstrap (Section 3.3.2). Value based methods such as **Q-learning** and **SARSA** are discussed in Section 35.2.

35.1.3 Policy search methods

In **policy search**, we try to directly maximize $J(\pi_{\boldsymbol{\theta}})$ wrt the policy parameter $\boldsymbol{\theta}$. If $J(\pi_{\boldsymbol{\theta}})$ is differentiable wrt $\boldsymbol{\theta}$, we can use stochastic gradient ascent to optimize $\boldsymbol{\theta}$, which is known as **policy gradient**, as described in Section 35.3.1. The basic idea is to perform **Monte Carlo rollouts**, in which we sample trajectories by interacting with the environment, and then use the score function estimator (Section 6.3.4) to estimate $\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}})$. Here, $J(\pi_{\boldsymbol{\theta}})$ is defined as an expectation whose distribution depends on $\boldsymbol{\theta}$, so it is invalid to swap ∇ and $\mathbb{E}$ in computing the gradient, and the score function estimator can be used instead. An example of policy gradient is **REINFORCE**.

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 35.5.3). 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 $\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}})$ 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, $V_{\boldsymbol{w}}(s)$. and to use it as a baseline in the score function estimator. We can learn $V_{\boldsymbol{w}}(s)$ using one of the value function methods similar to Q-learning. Alternatively, we can learn an advantage function, $A_{\boldsymbol{w}}(s, a)$, and use it to estimate the gradient. These policy gradient variants are called **actor critic** methods, where the actor refers to the policy $\pi_{\boldsymbol{\theta}}$ and the critic refers to $V_{\boldsymbol{w}}$ or $A_{\boldsymbol{w}}$. See Section 35.3.3 for details.

35.1.4 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 34.6). 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 35.4.

35.1.5 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 34.4. The literature on efficient exploration is huge. In this section, we briefly describe several representative techniques.

35.1.5.1 ϵ-greedy

A common heuristic is to use an **ϵ-greedy** policy π_ϵ, 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 ϵ. 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 $\epsilon/|\mathcal{A}|$.

35.1.5.2 Boltzmann exploration

A source of inefficiency in the ϵ-greedy rule is that exploration occurs uniformly over all actions. The **Boltzmann policy** can be more efficient, by assigning higher probabilities to explore more promising actions:

$$\pi_\tau(a|s) = \frac{\exp(\hat{R}_t(s_t, a)/\tau)}{\sum_{a'} \exp(\hat{R}_t(s_t, a')/\tau)} \tag{35.1}$$

where $\tau > 0$ is a temperature parameter that controls how entropic the distribution is. As τ gets close to 0, π_τ becomes close to a greedy policy. On the other hand, higher values of τ will make $\pi(a|s)$ more uniform, and encourage more exploration. Its action selection probabilities can be much "smoother" with respect to changes in the reward estimates than ϵ-greedy, as illustrated in Table 35.2.

35.1.5.3 Upper confidence bounds and Thompson sampling

The upper confidence bound (UCB) (Section 34.4.5) and Thompson sampling (Section 34.4.6) approaches may also be extended to MDPs. In contrast to the contextual bandit case, where the only uncertainty is in the reward function, here we must also take into account uncertainty in the transition probabilities.

As in the bandit case, the UCB approach requires to estimate an upper confidence bound for all actions' Q-values in the current state, and then take the action with the highest UCB score. One way to obtain UCBs of the Q-values is to use **count-based exploration**, where we learn the optimal

$\hat{R}(s, a_1)$	$\hat{R}(s, a_2)$	$\pi_\epsilon(a\|s_1)$	$\pi_\epsilon(a\|s_2)$	$\pi_\tau(a\|s_1)$	$\pi_\tau(a\|s_2)$
1.00	9.00	0.05	0.95	0.00	1.00
4.00	6.00	0.05	0.95	0.12	0.88
4.90	5.10	0.05	0.95	0.45	0.55
5.05	4.95	0.95	0.05	0.53	0.48
7.00	3.00	0.95	0.05	0.98	0.02
8.00	2.00	0.95	0.05	1.00	0.00

Table 35.2: Comparison of ϵ-greedy policy (with $\epsilon = 0.1$) and Boltzmann policy (with $\tau = 1$) for a simple MDP with 6 states and 2 actions. Adapted from Table 4.1 of [GK19].

Q-function with an **exploration bonus** added to the reward in a transition (s, a, r, s'):

$$\tilde{r} = r + \alpha/\sqrt{N_{s,a}} \tag{35.2}$$

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 that controls the degree of exploration. This is the approach taken by the **MBIE-EB** method [SL08] for finite-state MDPs, and in the generalization to continuous-state MDPs through the use of hashing [Bel+16]. Other approaches also explicitly maintain uncertainty in state transition probabilities, and use that information to obtain UCBs. Examples are **MBIE** [SL08], **UCRL2** [JOA10], and **UCBVI** [AOM17], among many others.

Thompson sampling can be similarly adapted, by maintaining the posterior distribution of the reward and transition model parameters. In finite-state MDPs, for example, the transition model is a categorical distribution conditioned on the state. We may use the conjugate prior of Dirichlet distributions (Section 3.4) 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 34.4.7) [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.

35.1.5.4 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 34.4.4 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:

$$T^+(s_{t+1}, b_{t+1}|s_t, b_t, a_t, r_t) = T^+(s_{t+1}|s_t, a_t, b_t)T^+(b_{t+1}|s_t, a_t, r_t, s_{t+1}) \tag{35.3}$$

$$= \mathbb{E}_{b_t}\left[T(s_{t+1}|s_t, a_t)\right] \times \mathbb{I}\left(b_{t+1} = p(R, T|\boldsymbol{h}_{t+1})\right) \tag{35.4}$$

where $\mathbb{E}_{b_t}\left[T(s_{t+1}|s_t, a_t)\right]$ is the posterior predictive distribution over next states, and $p(R, T|\boldsymbol{h}_{t+1})$ is the new belief state given $\boldsymbol{h}_{t+1} = (s_{1:t+1}, a_{1:t+1}, r_{1:t+1})$, which can be computed using Bayes' rule.

Similarly, the reward function for the augmented MDP is given by

$$R^+(r|s_t, b_t, a_t, s_{t+1}, b_{t+1}) = \mathbb{E}_{b_{t+1}} \left[R(s_t, a_t, s_{t+1}) \right] \tag{35.5}$$

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 19.6.4), 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 $\boldsymbol{h}_t$ using a VAE (Section 21.2). The posterior $p(m|\boldsymbol{h}_t)$ 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.

35.2 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.

35.2.1 Monte Carlo RL

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 a, and then sample the rest of the trajectory according to π, and then compute the average sum of discounted rewards. The trajectory ends when we reach a terminal state, if the task is episodic, or when the discount factor γ^t becomes negligibly small, whichever occurs first. This is the **Monte Carlo estimation** of the value function.

We can use this technique together with policy iteration (Section 34.6.2) to learn an optimal policy. Specifically, at iteration k, we compute a new, improved policy using $\pi_{k+1}(s) = \text{argmax}_a Q_k(s, a)$, where Q_k is approximated using MC estimation. This update can be applied to all the states visited on the sampled trajectory. This overall technique is called **Monte Carlo control**.

To ensure this method converges to the optimal policy, we need to collect data for every (state, action) pair, at least in the tabular case, since there is no generalization across different values of $Q(s, a)$. One way to achieve this is to use an ϵ-greedy policy. Since this is an on-policy algorithm, the resulting method will converge to the optimal ϵ-soft policy, as opposed to the optimal policy. It is possible to use importance sampling to estimate the value function for the optimal policy, even if actions are chosen according to the ϵ-greedy policy. However, it is simpler to just gradually reduce ϵ.

35.2.2 Temporal difference (TD) learning

The Monte Carlo (MC) method in Section 35.2.1 results in an estimator for $Q_\pi(s, a)$ with very high variance, since it has to unroll many trajectories, whose returns are a sum of many random rewards generated by stochastic state transitions. In addition, it is limited to episodic tasks (or finite horizon

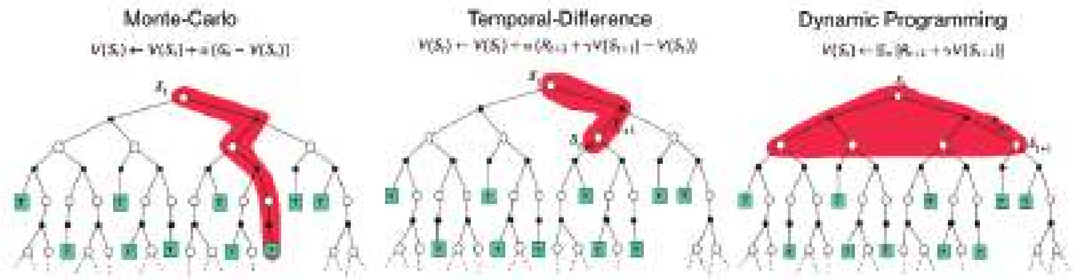


Figure 35.2: Backup diagrams of $V(s_t)$ for Monte Carlo, temporal difference, and dynamic programming updates of the state-value function. Used with kind permission of Andy Barto.

truncation of continuing tasks), since it must unroll to the end of the episode before each update step, to ensure it reliably estimates the long term return.

In this section, we discuss a more efficient technique called **temporal difference** or **TD** learning [Sut88]. The basic idea is to incrementally reduce the Bellman error for sampled states or state-actions, based on transitions instead of a long trajectory. More precisely, suppose we are to learn the value function V_π for a fixed policy π. Given a state transition (s, a, r, s') where $a \sim \pi(s)$, we change the estimate $V(s)$ so that it moves towards the bootstrapping target (Section 35.1.2)

$$V(s_t) \leftarrow V(s_t) + \eta \left[r_t + \gamma V(s_{t+1}) - V(s_t)\right] \tag{35.6}$$

where η is the learning rate. The term multiplied by η above is known as the **TD error**. A more general form of TD update for parametric value function representations is

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \left[r_t + \gamma V_{\boldsymbol{w}}(s_{t+1}) - V_{\boldsymbol{w}}(s_t)\right] \nabla_{\boldsymbol{w}} V_{\boldsymbol{w}}(s_t) \tag{35.7}$$

of which Equation (35.6) is a special case. The TD update rule for learning Q_π is similar.

It can be shown that TD learning in the tabular case, Equation (35.6), converges to the correct value function, under proper conditions [Ber19]. However, it may diverge when approximation is used (Equation (35.7)), an issue we will discuss further in Section 35.5.3.

The potential divergence of TD is also consistent with the fact that Equation (35.7) is not SGD (Section 6.3.1) on any objective function, despite a very similar form. Instead, it is an example of **bootstrapping**, in which the estimate, $V_{\boldsymbol{w}}(s_t)$, is updated to approach a target, $r_t + \gamma V_{\boldsymbol{w}}(s_{t+1})$, which is defined by the value function estimate itself. This idea is shared by DP methods like value iteration, although they rely on the complete MDP model to compute an exact Bellman backup. In 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 35.2 illustrates the difference between MC, TD, and DP.

35.2.3 TD learning with eligibility traces

A key difference between TD and MC is the way they estimate returns. Given a trajectory $\boldsymbol{\tau} = (s_0, a_0, r_0, s_1, \ldots, s_T)$, TD estimates the return from state s_t by one-step lookahead, $G_{t:t+1} = r_t +$

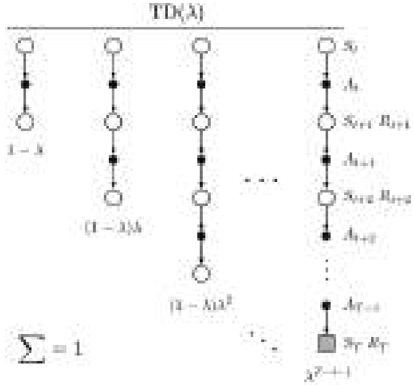


Figure 35.3: The backup diagram for TD(λ). Standard TD learning corresponds to $\lambda = 0$, and standard MC learning corresponds to $\lambda = 1$. From Figure 12.1 of [SB18]. Used with kind permission of Richard Sutton.

$\gamma V(s_{t+1})$, 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 $G_{t:T} = r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t-1} r_{T-1}$. 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 35.4.1.1). That is, we can use the n-step estimate

$$G_{t:t+n} = r_t + \gamma r_{t+1} + \cdots + \gamma^{n-1} r_{t+n-1} + \gamma^n V(s_{t+n}) \tag{35.8}$$

The corresponding n-step version of the TD update becomes

$$V(s_t) \leftarrow V(s_t) + \eta \left[G_{t:t+n} - V(s_t)\right] \tag{35.9}$$

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

$$G_t^\lambda \triangleq (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n} \tag{35.10}$$

This is called the **λ-return**. The coefficient of $1 - \lambda = (1 + \lambda + \lambda^2 + \cdots)^{-1}$ in the front ensures this is a convex combination of n-step returns. See Figure 35.3 for an illustration.

An important benefit of using the geometric weighting in Equation (35.10) is that the corresponding TD learning update can be efficiently implemented, through the use of **eligibility traces**, even though G_t^λ is a sum of infinitely many terms. The method is called **TD(λ)**, and can be combined with many algorithms to be studied in the rest of the chapter. See [SB18] for a detailed discussion.

35.2.4 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 34.6.2). In this case, it is more convenient to work with the action-value function, Q, and a policy π that is greedy with respect to Q. The agent follows π in every step to choose actions, and upon a transition (s, a, r, s') the TD update rule is

$$Q(s,a) \leftarrow Q(s,a) + \eta\left[r + \gamma Q(s',a') - Q(s,a)\right] \tag{35.11}$$

where $a' \sim \pi(s')$ is the action the agent will take in state s'. After Q is updated (for policy evaluation), π 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 ϵ-greedy policy, with ϵ vanishing to 0 gradually. It can be shown that SARSA with a GLIE policy will converge to Q_* and π_* [Sin+00].

35.2.5 Q-learning: off-policy TD control

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, with a simple modification, we can convert this to an **off-policy** algorithm that learns Q_*, even if a suboptimal policy is used to choose actions.

The idea is to replace the sampled next action $a' \sim \pi(s')$ in Equation (35.11) with a greedy action in s': $a' = \operatorname{argmax}_b Q(s', b)$. This results in the following update when a transition (s, a, r, s') happens

$$Q(s,a) \leftarrow Q(s,a) + \eta\left[r + \gamma \max_b Q(s',b) - Q(s,a)\right] \tag{35.12}$$

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 (35.7). 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 35.1 gives a vanilla implementation of Q-learning with ϵ-greedy exploration.

35.2.5.1 Example

Figure 35.4 gives an example of Q-learning applied to the simple 1d grid world from Figure 34.13, using $\gamma = 0.9$. We show the Q-functon at the start and end of each episode, after performing actions chosen by an ϵ-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.

35.2.5.2 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, $\mathbb{E}\left[\max_a X_a\right] \geq \max_a \mathbb{E}\left[X_a\right]$,

Algorithm 35.1: Q-learning with ϵ-greedy exploration

Initialize value function parameters $\boldsymbol{w}$
repeat
  Sample starting state s of new episode
  repeat
    Sample action $a = \begin{cases} \operatorname{argmax}_b Q_{\boldsymbol{w}}(s,b), & \text{with probability } 1-\epsilon \\ \text{random action}, & \text{with probability } \epsilon \end{cases}$
    Observe state s', reward r
    Compute the TD error: $\delta = r + \gamma \max_{a'} Q_{\boldsymbol{w}}(s',a') - Q_{\boldsymbol{w}}(s,a)$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta\delta\nabla_{\boldsymbol{w}} Q_{\boldsymbol{w}}(s,a)$
    $s \leftarrow s'$
  until *state s is terminal*
until *converged*

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 35.5 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 -0.1, making it suboptimal. Nevertheless, the RL algorithm may pick the left action due to the maximization bias making B appear to have a positive value.

One solution to avoid the maximization bias is to use two separate Q-functions, Q_1 and Q_2, one for selecting the greedy action, and the other for estimating the corresponding Q-value. In particular, upon seeing a transition (s,a,r,s'), we perform the following update

$$Q_1(s,a) \leftarrow Q_1(s,a) + \eta \left[r + \gamma Q_2\big(s', \operatorname*{argmax}_{a'} Q_1(s',a')\big) - Q_1(s,a) \right] \tag{35.13}$$

and may repeat the same update but with the roles of Q_1 and Q_2 swapped. This technique is called **double Q-learning** [Has10]. Figure 35.5 shows the benefits of the algorithm over standard Q-learning in a toy problem.

35.2.6 Deep Q-network (DQN)

When function approximation is used, Q-learning may be hard to use in practice due to instability problems. Here, we will describe two important heuristics, popularized by the **deep Q-network** or **DQN** work [Mni+15], which was able to train agents to outperform humans at playing Atari games, using CNN-structured Q-networks.

The first technique, originally proposed in [Lin92], is to leverage an **experience replace** buffer, which stores the most recent (s,a,r,s') transition tuples. In contrast to standard Q-learning which updates the Q-function when a new transition occurs, the DQN agent also performs additional updates using transitions sampled from the buffer. This modification has two advantages. First, it

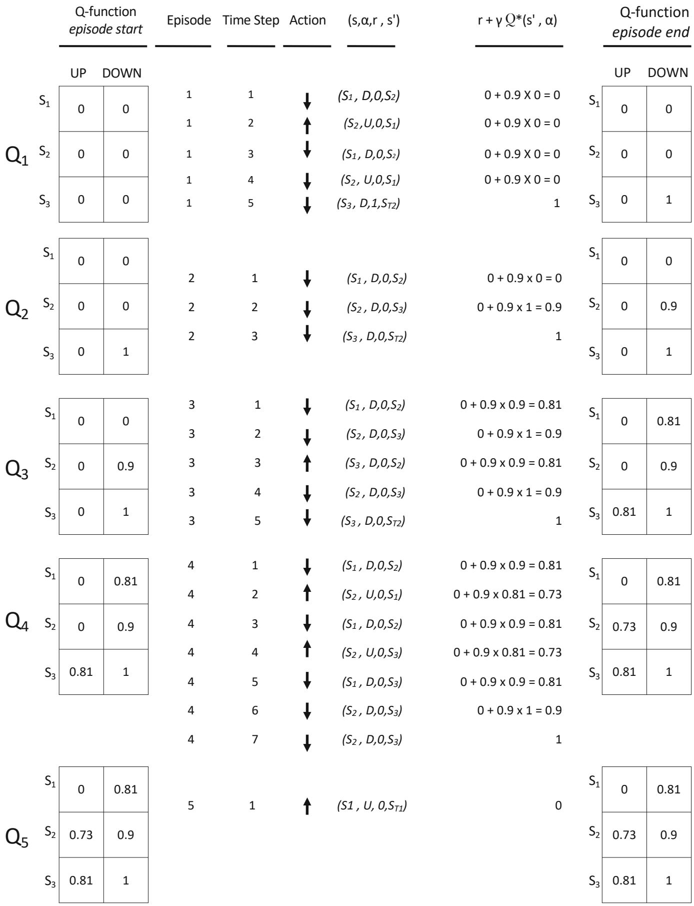


Figure 35.4: Illustration of Q learning for the 1d grid world in Figure 34.13 using ϵ-greedy exploration. At the end of episode 1, we make a transition from S_3 to S_{T2} and get a reward of $r = 1$, so we estimate $Q(S_3, \downarrow) = 1$. In episode 2, we make a transition from S_2 to S_3, so S_2 gets incremented by $\gamma Q(S_3, \downarrow) = 0.9$. Adapted from Figure 3.3 of [GK19].

improves data efficiency as every transition can be used multiple times. Second, it improves stability in training, by reducing the correlation of the data samples that the network is trained on.

The second idea to improve stability is to regress the Q-network to a "frozen" **target network** computed at an earlier iteration, rather than trying to chase a constantly moving target. Specifically, we maintain an extra, frozen copy of the Q-network, $Q_{\boldsymbol{w}^-}$, of the same structure as $Q_{\boldsymbol{w}}$. This new Q-network is to compute bootstrapping targets for training $Q_{\boldsymbol{w}}$, in which the loss function is

$$\mathcal{L}^{\text{DQN}}(\boldsymbol{w}) = \mathbb{E}_{(s,a,r,s') \sim U(\mathcal{D})} \left[\left(r + \gamma \max_{a'} Q_{\boldsymbol{w}^-}(s', a') - Q_{\boldsymbol{w}}(s, a) \right)^2 \right] \tag{35.14}$$

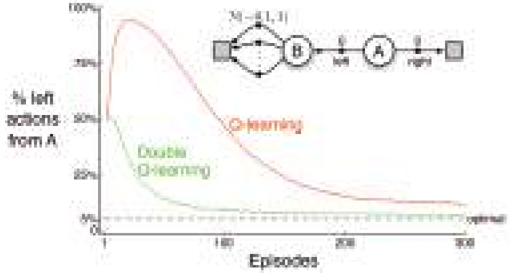


Figure 35.5: Comparison of Q-learning and double Q-learning on a simple episodic MDP using ϵ-greedy action selection with $\epsilon = 0.1$. The initial state is A, and squares denote absorbing states. The data are averaged over 10,000 runs. From Figure 6.5 of [SB18]. Used with kind permission of Richard Sutton.

where $U(\mathcal{D})$ is a uniform distribution over the replay buffer $\mathcal{D}$. We then periodically set $\boldsymbol{w}^- \leftarrow \boldsymbol{w}$, usually after a few episodes. This approach is an instance of **fitted value iteration** [SB18].

Various improvements to DQN have been proposed. One is **double DQN** [HGS16], which uses the double learning technique (Section 35.2.5.2) to remove the maximization bias. The second is to replace the uniform distribution in Equation (35.14) with one that favors more important transition tuples, resulting in the use of **prioritized experience replay** [Sch+16a]. For example, we can sample transitions from $\mathcal{D}$ with probability $p(s, a, r, s') \propto (|\delta| + \varepsilon)^\eta$, where δ is the corresponding TD error (under the current Q-function), $\varepsilon > 0$ a hyperparameter to ensure every experience is chosen with nonzero probability, and $\eta \geq 0$ controls the "inverse temperature" of the distribution (so $\eta = 0$ corresponds to uniform sampling). The third is to learn a value function $V_{\boldsymbol{w}}$ and an advantage function $A_{\boldsymbol{w}}$, with shared parameter $\boldsymbol{w}$, instead of learning $Q_{\boldsymbol{w}}$. The resulting **dueling DQN** [Wan+16] is shown to be more sample efficient, especially when there are many actions with similar Q-values.

The **rainbow** method [Hes+18] combines all three improvements, as well as others, including multi-step returns (Section 35.2.3), distributional RL [BDM17] (which predicts the distribution of returns, not just the expected return), and noisy nets [For+18b] (which adds random noise to the network weights to encourage exploration). It produces state-of-the-art results on the Atari benchmark.

35.3 Policy-based RL

In the previous section, we considered methods that estimate the action-value function, $Q(s, a)$, from which we derive a policy, which may be greedy or softmax. However, these methods have three main disadvantages: (1) they can be difficult to apply to continuous action spaces; (2) they may diverge if function approximation is used; and (3) the training of Q, often based on TD-style updates, is not directly related to the expected return garnered by the learned policy.

In this section, we discuss **policy search** methods, which directly optimize the parameters of the policy so as to maximize its expected return. However, we will see that these methods often benefit from estimating a value or advantage function to reduce the variance in the policy search process.

35.3.1 The policy gradient theorem

We start by defining the objective function for policy learning, and then derive its gradient. We consider the episodic case. A similar result can be derived for the continuing case with the average reward criterion [SB18, Sec 13.6].

We define the objective to be the expected return of a policy, which we aim to maximize:

$$J(\pi) \triangleq \mathbb{E}_{p_0,\pi}\left[G_0\right] = \mathbb{E}_{p_0(s_0)}\left[V_\pi(s_0)\right] = \mathbb{E}_{p_0(s_0)\pi(a_0|s_0)}\left[Q_\pi(s_0,a_0)\right] \tag{35.15}$$

We consider policies $\pi_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$, and compute the gradient of Equation (35.15) wrt $\boldsymbol{\theta}$:

$$\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) = \mathbb{E}_{p_0(s_0)}\left[\nabla_{\boldsymbol{\theta}}\left(\sum_{a_0}\pi_{\boldsymbol{\theta}}(a_0|s_0)Q_{\pi_{\boldsymbol{\theta}}}(s_0,a_0)\right)\right] \tag{35.16}$$

$$= \mathbb{E}_{p_0(s_0)}\left[\sum_{a_0}\nabla\pi_{\boldsymbol{\theta}}(a_0|s_0)Q_{\pi_{\boldsymbol{\theta}}}(s_0,a_0)\right] + \mathbb{E}_{p_0(s_0)\pi_{\boldsymbol{\theta}}(a_0|s_0)}\left[\nabla_{\boldsymbol{\theta}} Q_{\pi_{\boldsymbol{\theta}}}(s_0,a_0)\right] \tag{35.17}$$

Now we calculate the term $\nabla_{\boldsymbol{\theta}} Q_{\pi_{\boldsymbol{\theta}}}(s_0,a_0)$:

$$\nabla_{\boldsymbol{\theta}} Q_{\pi_{\boldsymbol{\theta}}}(s_0,a_0) = \nabla_{\boldsymbol{\theta}}\left[R(s_0,a_0) + \gamma\mathbb{E}_{p_T(s_1|s_0,a_0)}\left[V_{\pi_{\boldsymbol{\theta}}}(s_1)\right]\right] = \gamma\nabla_{\boldsymbol{\theta}}\mathbb{E}_{p_T(s_1|s_0,a_0)}\left[V_{\pi_{\boldsymbol{\theta}}}(s_1)\right] \tag{35.18}$$

The right-hand side above is in a form similar to $\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}})$. Repeating the same steps as before gives

$$\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) = \sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{p_t(s)}\left[\sum_a \nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(a|s)Q_{\pi_{\boldsymbol{\theta}}}(s,a)\right] \tag{35.19}$$

$$= \frac{1}{1-\gamma}\mathbb{E}_{p^\infty_{\pi_{\boldsymbol{\theta}}}(s)}\left[\sum_a \nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(a|s)Q_{\pi_{\boldsymbol{\theta}}}(s,a)\right] \tag{35.20}$$

$$= \frac{1}{1-\gamma}\mathbb{E}_{p^\infty_{\pi_{\boldsymbol{\theta}}}(s)\pi_{\boldsymbol{\theta}}(a|s)}\left[\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a|s)\; Q_{\pi_{\boldsymbol{\theta}}}(s,a)\right] \tag{35.21}$$

where $p_t(s)$ is the probability of visiting s in time t if we start with $s_0 \sim p_0$ and follow $\pi_{\boldsymbol{\theta}}$, and $p^\infty_{\pi_{\boldsymbol{\theta}}}(s) = (1-\gamma)\sum_{t=0}^\infty \gamma^t p_t(s)$ is the normalized discounted state visitation distribution. Equation (35.21) is known as the **policy gradient theorem** [Sut+99].

In practice, estimating the policy gradient using Equation (35.21) can have a high variance. A baseline $b(s)$ can be used for variance reduction (Section 6.3.4.1):

$$\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) = \frac{1}{1-\gamma}\mathbb{E}_{p^\infty_{\pi_{\boldsymbol{\theta}}}(s)\pi_{\boldsymbol{\theta}}(a|s)}\left[\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a|s)(Q_{\pi_{\boldsymbol{\theta}}}(s,a) - b(s))\right] \tag{35.22}$$

A common choice for the baseline is $b(s) = V_{\pi_{\boldsymbol{\theta}}}(s)$. We will discuss how to estimate it below.

35.3.2 REINFORCE

One way to apply the policy gradient theorem to optimize a policy is to use stochastic gradient ascent. Suppose $\boldsymbol{\tau} = (s_0, a_0, r_0, s_1, \ldots, s_T)$ is a trajectory with $s_0 \sim p_0$ and $\pi_{\boldsymbol{\theta}}$. Then,

$$\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) = \frac{1}{1-\gamma} \mathbb{E}_{p^{\infty}_{\pi_{\boldsymbol{\theta}}}(s)\pi_{\boldsymbol{\theta}}(a|s)} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) Q_{\pi_{\boldsymbol{\theta}}}(s,a) \right] \tag{35.23}$$

$$\approx \sum_{t=0}^{T-1} \gamma^t G_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \tag{35.24}$$

where the return G_t is defined in Equation (34.76), and the factor γ^t is due to the definition of $p^{\infty}_{\pi_{\boldsymbol{\theta}}}$ where the state at time t is discounted.

We can use a baseline in the gradient estimate to get the following update rule:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \sum_{t=0}^{T-1} \gamma^t (G_t - b(s_t)) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \tag{35.25}$$

This is called the **REINFORCE** algorithm [Wil92].[3] The update equation can be interepreted 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 $\boldsymbol{\theta}$ so as to make this trajectory more likely, otherwise we decrease $\boldsymbol{\theta}$. 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, $V_{\boldsymbol{w}}(s)$, which can be learned, e.g., with MC. Algorithm 35.2 gives the pseudocode where stochastic gradient updates are used with separate learning rates.

Algorithm 35.2: REINFORCE with value function baseline

Initialize policy parameters $\boldsymbol{\theta}$, baseline parameters $\boldsymbol{w}$
repeat
  Sample an episode $\boldsymbol{\tau} = (s_0, a_0, r_0, s_1, \ldots, s_T)$ using $\pi_{\boldsymbol{\theta}}$
  Compute G_t for all $t \in \{0, 1, \ldots, T-1\}$ using Equation (34.76)
  for $t = 0, 1, \ldots, T-1$ **do**
    $\delta = G_t - V_{\boldsymbol{w}}(s_t)$ // scalar error
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta_{\boldsymbol{w}} \delta \nabla_{\boldsymbol{w}} V_{\boldsymbol{w}}(s_t)$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_{\boldsymbol{\theta}} \gamma^t \delta \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)$
until *converged*

35.3.3 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

3. The term "REINFORCE" is an acronym for "REward Increment = nonnegative Factor x Offset Reinforcement x Characteristic Eligibility". The phrase "characteristic eligibility" refers to the $\nabla \log \pi_{\boldsymbol{\theta}}(a_t|s_t)$ term; the phrase "offset reinforcement" refers to the $G_t - b(s_t)$ term; and the phrase "nonnegative factor" refers to the learning rate η of SGD.

"actor" refers to the policy, and the term "critic" refers to the value function. The use of bootstrapping in TD updates allows more efficient learning of the value function compared to MC. In addition, it allows us to develop a fully online, incremental algorithm, that does not need to wait until the end of the trajectory before updating the parameters (as in Algorithm 35.2).

Concretely, consider the use of the one-step TD(0) method to estimate the return in the episodic csae, i.e., we replace G_t with $G_{t:t+1} = r_t + \gamma V_{\boldsymbol{w}}(s_{t+1})$. If we use $V_{\boldsymbol{w}}(s_t)$ as a baseline, the REINFORCE update in Equation (35.25) becomes

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \sum_{t=0}^{T-1} \gamma^t \left(G_{t:t+1} - V_{\boldsymbol{w}}(s_t)\right) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \tag{35.26}$$

$$= \boldsymbol{\theta} + \eta \sum_{t=0}^{T-1} \gamma^t \big(r_t + \gamma V_{\boldsymbol{w}}(s_{t+1}) - V_{\boldsymbol{w}}(s_t)\big) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \tag{35.27}$$

35.3.3.1 A2C and A3C

Note that $r_{t+1} + \gamma V_{\boldsymbol{w}}(s_{t+1}) - V_{\boldsymbol{w}}(s_t)$ is a single sample approximation to the advantage function $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$. This method is therefore called **advantage actor critic** or **A2C** (Algorithm 35.3). If we run the actors in parallel and asynchronously update their shared parameters, the method is called **asynchrononous advantage actor critic** or **A3C** [Mni+16].

Algorithm 35.3: Advantage actor critic (A2C) algorithm

Initialize actor parameters $\boldsymbol{\theta}$, critic parameters $\boldsymbol{w}$
repeat
    Sample starting state s_0 of a new episode
    for $t = 0, 1, 2, \ldots$ **do**
        Sample action $a_t \sim \pi_{\boldsymbol{\theta}}(\cdot|s_t)$
        Observe next state s_{t+1} and reward r_t
        $\delta = r_t + \gamma V_{\boldsymbol{w}}(s_{t+1}) - V_{\boldsymbol{w}}(s_t)$
        $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta_{\boldsymbol{w}} \delta \nabla_{\boldsymbol{w}} V_{\boldsymbol{w}}(s_t)$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_{\boldsymbol{\theta}} \gamma^t \delta \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)$
until *converged*

35.3.3.2 Eligibility traces

In A2C, we use a single step rollout, and then use the value function in order to approximate the expected return for the trajectory. More generally, we can use the n-step estimate

$$G_{t:t+n} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{n-1} r_{t+n-1} + \gamma^n V_{\boldsymbol{w}}(s_{t+n}) \tag{35.28}$$

and obtain an n-step advantage estimate as follows:

$$A_{\pi_{\boldsymbol{\theta}}}^{(n)}(s_t, a_t) = G_{t:t+n} - V_{\boldsymbol{w}}(s_t) \tag{35.29}$$

The n steps of actual rewards are an unbiased sample, but have high variance. By contrast, $V_{\boldsymbol{w}}(s_{t+n+1})$ has lower variance, but is biased. By changing n, we can control the bias-variance tradeoff. Instead of using a single value of n, we can take an weighted average, with weight proportional to λ^n for $A_{\pi_{\boldsymbol{\theta}}}^{(n)}(s_t, a_t)$, as in TD($\lambda$). The average can be shown to be equivalent to

$$A_{\pi_{\boldsymbol{\theta}}}^{(\lambda)}(s_t, a_t) \triangleq \sum_{\ell=0}^{\infty} (\gamma\lambda)^{\ell} \delta_{t+l} \tag{35.30}$$

where $\delta_t = r_t + \gamma V_{\boldsymbol{w}}(s_{t+1}) - V_{\boldsymbol{w}}(s_t)$ is the TD error at time t. Here, $\lambda \in [0, 1]$ is a parameter that controls the bias-variance tradeoff: larger values decrease the bias but increase the variance, as in TD(λ). We can implement Equation (35.30) efficiently using eligibility traces, as shown in Algorithm 35.4, as an example of **generalized advantage estimation** (**GAE**) [Sch+16b]. See [SB18, Ch.12] for further details.

Algorithm 35.4: Actor critic with eligibility traces

Initialize actor parameters $\boldsymbol{\theta}$, critic parameters $\boldsymbol{w}$
repeat
    Initialize eligibility trace vectors: $\boldsymbol{z}_{\boldsymbol{\theta}} \leftarrow \boldsymbol{0}$, $\boldsymbol{z}_{\boldsymbol{w}} \leftarrow \boldsymbol{0}$
    Sample starting state s_0 of a new episode
    for $t = 0, 1, 2, \ldots$ **do**
        Sample action $a_t \sim \pi_{\boldsymbol{\theta}}(\cdot|s_t)$
        Observe state s_{t+1} and reward r_t
        Compute the TD error: $\delta = r_t + \gamma V_{\boldsymbol{w}}(s_{t+1}) - V_{\boldsymbol{w}}(s_t)$
        $\boldsymbol{z}_{\boldsymbol{w}} \leftarrow \gamma \lambda_{\boldsymbol{w}} \boldsymbol{z}_{\boldsymbol{w}} + \nabla_{\boldsymbol{w}} V_{\boldsymbol{w}}(s)$
        $\boldsymbol{z}_{\boldsymbol{\theta}} \leftarrow \gamma \lambda_{\boldsymbol{\theta}} \boldsymbol{z}_{\boldsymbol{\theta}} + \gamma^t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)$
        $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta_{\boldsymbol{w}} \delta \boldsymbol{z}_{\boldsymbol{w}}$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_{\boldsymbol{\theta}} \delta \boldsymbol{z}_{\boldsymbol{\theta}}$
until *converged*

35.3.4 Bound optimization methods

In policy gradient methods, the objective $J(\boldsymbol{\theta})$ does not necessarily increase monotonically, but rather can collapse especially if the learning rate is not small enough. We now describe methods that guarantee monotonic improvement, similar to bound optimization algorithms (Section 6.5).

We start with a useful fact that relate the policy values of two arbitrary policies [KL02]:

$$J(\pi') - J(\pi) = \frac{1}{1-\gamma} \mathbb{E}_{p_{\pi'}^{\infty}(s)} \left[\mathbb{E}_{\pi'(a|s)} \left[A_{\pi}(s, a) \right] \right] \tag{35.31}$$

where π can be interpreted as the current policy during policy optimization, and π' a candidate new policy (such as the greedy policy wrt Q_π). As in the policy improvement theorem (Section 34.6.2), 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 hold when function approximation is used, as such a uniformly improving policy π' may not be representable by our parametric family, $\{\pi_{\boldsymbol{\theta}}\}_{\boldsymbol{\theta}\in\Theta}$. Therefore, nonnegativity of Equation (35.31) is not easy to ensure, when we do not have a direct way to sample states from $p^{\infty}_{\pi'}$.

One way to ensure monotonic improvement of J is to improve the policy conservatively. Define $\pi_\theta = \theta\pi' + (1-\theta)\pi$ for $\theta \in [0,1]$. It follows from the policy gradient theorem (Equation (35.21), with $\boldsymbol{\theta} = [\theta]$) that $J(\pi_\theta) - J(\pi) = \theta L(\pi') + O(\theta^2)$, where

$$L(\pi') \triangleq \frac{1}{1-\gamma}\mathbb{E}_{p^{\infty}_{\pi}(s)}\left[\mathbb{E}_{\pi'(a|s)}\left[A_\pi(s,a)\right]\right] = \frac{1}{1-\gamma}\mathbb{E}_{p^{\infty}_{\pi}(s)\pi(a|s)}\left[\frac{\pi'(a|s)}{\pi(a|s)}A_\pi(s,a)\right] \tag{35.32}$$

In the above, we have switched the state distribution from $p^{\infty}_{\pi'}$ in Equation (35.31) to p^{∞}_{π}, while at the same time introducing a higher order residual term of $O(\theta^2)$. The linear term, $\theta L(\pi')$, can be estimated and optimized based on episodes sampled by π. The higher order term can be bounded in various ways, resulting in different lower bounds of $J(\pi_\theta) - J(\pi)$. We can then optimize θ to make sure this lower bound is positive, which would imply $J(\pi_\theta) - J(\pi) > 0$. In **conservative policy iteration** [KL02], the following (slightly simplified) lower bound is used

$$J^{\text{CPI}}(\pi_\theta) \triangleq J(\pi) + \theta L(\pi') - \frac{2\varepsilon\gamma}{(1-\gamma)^2}\theta^2 \tag{35.33}$$

where $\varepsilon = \max_s \left|\mathbb{E}_{\pi'(a|s)}\left[A_\pi(s,a)\right]\right|$.

This idea can be generalized to policies beyond those in the form of π_θ, where the condition of a small enough θ is replaced by a small enough divergence between π' and π. In **safe policy iteration** [Pir+13], the divergence is the maximum total variation, while in **trust region policy optimization** (**TRPO**) [Sch+15b], the divergence is the maximum KL-divergence. In the latter case, π' may be found by optimizing the following lower bound

$$J^{\text{TRPO}}(\pi') \triangleq J(\pi) + L(\pi') - \frac{\varepsilon\gamma}{(1-\gamma)^2}\max_s D_{\mathbb{KL}}\left(\pi(s) \parallel \pi'(s)\right) \tag{35.34}$$

where $\varepsilon = \max_{s,a}|A_\pi(s,a)|$.

In practice, the above update rule can be overly conservative, and approximations are used. [Sch+15b] propose a version that implements two ideas: one is to replace the point-wise maximum KL-divergence by some average KL-divergence (usually averaged over $p^{\infty}_{\pi_{\boldsymbol{\theta}}}$); the second is to maximize the first two terms in Equation (35.34), with π' lying in a KL-ball centered at π. That is, we solve

$$\operatorname*{argmax}_{\pi'} L(\pi') \;\; \text{s.t.} \;\; \mathbb{E}_{p^{\infty}_{\pi}(s)}\left[D_{\mathbb{KL}}\left(\pi(s) \parallel \pi'(s)\right)\right] \leq \delta \tag{35.35}$$

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 η in parameter space does not always correspond to a step of size η in the policy space:

$$d_{\boldsymbol{\theta}}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = d_{\boldsymbol{\theta}}(\boldsymbol{\theta}_2, \boldsymbol{\theta}_3) \not\Rightarrow d_\pi(\pi_{\boldsymbol{\theta}_1}, \pi_{\boldsymbol{\theta}_2}) = d_\pi(\pi_{\boldsymbol{\theta}_2}, \pi_{\boldsymbol{\theta}_3}) \tag{35.36}$$

where $d_{\boldsymbol{\theta}}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|$ is the Euclidean distance, and $d_\pi(\pi_1, \pi_2) = D_{\mathbb{KL}}(\pi_1 \parallel \pi_2)$ 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 (35.34) 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]:

$$J^{\text{PPO}}(\pi') \triangleq \frac{1}{1-\gamma}\mathbb{E}_{p_\pi^\infty(s)\pi(a|s)}\left[\kappa_\epsilon\left(\frac{\pi'(a|s)}{\pi(a|s)}\right)A_\pi(s,a)\right] \tag{35.37}$$

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.

35.3.5 Deterministic policy gradient methods

In this section, we consider the case of a deterministic policy, that predicts a unique action for each state, so $a_t = \mu_{\boldsymbol{\theta}}(s_t)$, rather than $a_t \sim \pi_{\boldsymbol{\theta}}(s_t)$. We assume the states and actions are continuous, and define the objective as

$$J(\mu_{\boldsymbol{\theta}}) \triangleq \frac{1}{1-\gamma}\mathbb{E}_{p_{\mu_{\boldsymbol{\theta}}}^\infty(s)}\left[R(s,\mu_{\boldsymbol{\theta}}(s))\right] \tag{35.38}$$

The **deterministic policy gradient theorem** [Sil+14] provides a way to compute the gradient:

$$\nabla_{\boldsymbol{\theta}} J(\mu_{\boldsymbol{\theta}}) = \frac{1}{1-\gamma}\mathbb{E}_{p_{\mu_{\boldsymbol{\theta}}}^\infty(s)}\left[\nabla_{\boldsymbol{\theta}} Q_{\mu_{\boldsymbol{\theta}}}(s,\mu_{\boldsymbol{\theta}}(s))\right] \tag{35.39}$$

$$= \frac{1}{1-\gamma}\mathbb{E}_{p_{\mu_{\boldsymbol{\theta}}}^\infty(s)}\left[\nabla_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}}(s)\nabla_a Q_{\mu_{\boldsymbol{\theta}}}(s,a)|_{a=\mu_{\boldsymbol{\theta}}(s)}\right] \tag{35.40}$$

where $\nabla_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}}(s)$ is the $M \times N$ Jacobian matrix, and M and N are the dimensions of $\mathcal{A}$ and $\boldsymbol{\theta}$, respectively. For stochastic policies of the form $\pi_{\boldsymbol{\theta}}(a|s) = \mu_{\boldsymbol{\theta}}(s) + \text{noise}$, the standard policy gradient theorem reduces to the above form as the noise level goes to zero.

Note that the gradient estimate in Equation (35.40) 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 β, whose stationary state distribution is p_β^∞. The original objective, $J(\mu_{\boldsymbol{\theta}})$, is approximated by the following:

$$J_b(\mu_{\boldsymbol{\theta}}) \triangleq \mathbb{E}_{p_\beta^\infty(s)}\left[V_{\mu_{\boldsymbol{\theta}}}(s)\right] = \mathbb{E}_{p_\beta^\infty(s)}\left[Q_{\mu_{\boldsymbol{\theta}}}(s,\mu_{\boldsymbol{\theta}}(s))\right] \tag{35.41}$$

with the off-policy deterministic policy gradient approximated by (see also Section 35.5.1.2)

$$\nabla_{\boldsymbol{\theta}} J_b(\mu_{\boldsymbol{\theta}}) \approx \mathbb{E}_{p_\beta^\infty(s)}\left[\nabla_{\boldsymbol{\theta}}\left[Q_{\mu_{\boldsymbol{\theta}}}(s,\mu_{\boldsymbol{\theta}}(s))\right]\right] = \mathbb{E}_{p_\beta^\infty(s)}\left[\nabla_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}}(s)\nabla_a Q_{\mu_{\boldsymbol{\theta}}}(s,a)|_{a=\mu_{\boldsymbol{\theta}}(s)}ds\right] \tag{35.42}$$

where we have a dropped a term that depends on $\nabla_{\boldsymbol{\theta}} Q_{\mu_{\boldsymbol{\theta}}}(s,a)$ and is hard to estimate [Sil+14].

To apply Equation (35.42), we may learn $Q_{\boldsymbol{w}} \approx Q_{\mu_{\boldsymbol{\theta}}}$ with TD, giving rise to the following updates:

$$\delta = r_t + \gamma Q_{\boldsymbol{w}}(s_{t+1},\mu_{\boldsymbol{\theta}}(s_{t+1})) - Q_{\boldsymbol{w}}(s_t,a_t) \tag{35.43}$$

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \eta_{\boldsymbol{w}}\delta\nabla_{\boldsymbol{w}} Q_{\boldsymbol{w}}(s_t,a_t) \tag{35.44}$$

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \eta_{\boldsymbol{\theta}}\nabla_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}}(s_t)\nabla_a Q_{\boldsymbol{w}}(s_t,a)|_{a=\mu_{\boldsymbol{\theta}}(s_t)} \tag{35.45}$$

This avoids importance sampling in the actor update because of the deterministic policy gradient, and avoids importance sampling in the critic update because of the use of Q-learning.

If $Q_{\boldsymbol{w}}$ is linear in $\boldsymbol{w}$, and uses features of the form $\boldsymbol{\phi}(s,a) = \boldsymbol{a}^\mathsf{T} \nabla_{\boldsymbol{\theta}} \mu_{\boldsymbol{\theta}}(s)$, where $\boldsymbol{a}$ is the vector representation of a, then we say the function approximator for the critic is **compatible** with the actor; in this case, one can show that the above approximation does not bias the overall gradient. The method has been extended in various ways. The **DDPG** algorithm of [Lil+16], which stands for "deep deterministic policy gradient", uses the DQN method (Section 35.2.6) to update Q that is represented by deep neural networks. The **TD3** algorithm [FHM18], which stands for "twin delayed DDPG", extends DDPG by using double DQN (Section 35.2.5.2) and other heuristics to further improve performance. Finally, the **D4PG** algorithm [BM+18], which stands for "distributed distributional DDPG", extends DDPG to handle distributed training, and to handle **distributional RL** (i.e., working with distributions of rewards instead of expected rewards [BDM17]).

35.3.6 Gradient-free methods

The policy gradient estimator computes a "zeroth order" gradient, which essentially evaluates the function with randomly sampled trajectories. Sometimes it can be more efficient to use a derivative-free optimizer (Section 6.7), that does not even attempt to estimate the gradient. For example, [MGR18] obtain good results by training linear policies with random search, and [Sal+17b] show how to use evolutionary strategies to optimize the policy of a robotic controller.

35.4 Model-based RL

Model-free approaches to RL typically need a lot of interactions with the environment to achieve good performance. For example, state of the art methods for the Atari benchmark, such as rainbow (Section 35.2.6), use millions of frames, equivalent to many days of playing at the standard frame rate. By contrast, humans can achieve the same performance in minutes [Tsi+17]. Similarly, OpenAI's robot hand controller [And+20] learns to manipulate a cube using 100 years of simulated data.

One promising approach to greater sample efficiency is **model-based RL** (**MBRL**). In this approach, we first learn the transition model and reward function, $p_T(s'|s,a)$ and $R(s,a)$, then use them to compute a near-optimal policy. This approach can significantly reduce the amount of real-world data that the agent needs to collect, since it can "try things out" in its imagination (i.e., the models), rather than having to try them out empirically.

There are several ways we can use a model, and many different kinds of model we can create. Some of the algorithms mentioned earlier, such as MBIE and UCLR2 for provably efficient exploration (Section 35.1.5.3), are examples of model-based methods. MBRL also provides a natural connection between RL and planning (Section 34.6) [Sut90]. We discuss some examples in the sections below, and refer to [MBJ20; PKP21; MH20] for more detailed reviews.

35.4.1 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

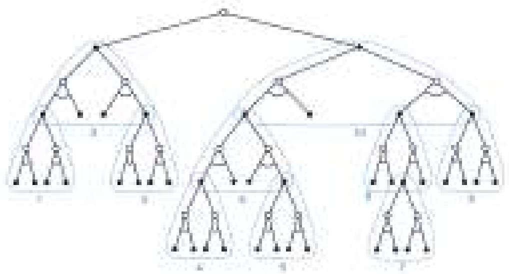


Figure 35.6: Illustration of heuristic search. In this figure, the subtrees are ordered according to a depth-first search procedure. From Figure 8.9 of [SB18]. Used with kind permission of Richard Sutton.

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

$$\boldsymbol{a}^*_{t:t+H-1} = \operatorname*{argmax}_{\boldsymbol{a}_{t:t+H-1}} \mathbb{E}\left[\sum_{h=0}^{H-1} R(s_{t+h}, a_{t+h}) + \hat{V}(s_{t+H})\right] \tag{35.46}$$

where the expectation is over state sequences that might result from executing $\boldsymbol{a}_{t:t+H-1}$ 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 **receeding horizon control** or **model predictive control** (**MPC**) [MM90; CA13]. We discuss some special cases of this below.

35.4.1.1 Heuristic search

If the state and action spaces are finite, we can solve Equation (35.46) 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 35.6.

If the heuristic function is an optimistic lower bound on the true distance to the goal, it is called **admissible**; If we aim to maximize total rewards, admissibility means the heuristic function is an upper bound of the true value function. Admissibility ensures we will never incorrectly prune off parts of the search space. In this case, the resulting algorithm is known as A^* **search**, and is optimal. For more details on classical AI **heuristic search** methods, see [Pea84; RN19].

35.4.1.2 Monte Carlo tree search (MCTS)

Monte Carlo tree search or **MCTS** is similar to heuristic search, but learns a value function for each encountered state, rather than relying on a manually designed heuristic (see e.g., [Mun14] for details). MCTS is inspired by UCB for bandits (Section 34.4.5), but applies to general sequential decision making problems including MDPs [KS06].

The MCTS method forms the basis of the famous **AlphaGo** and **AlphaZero** programs [Sil+16; Sil+18], which can play expert-level Go, chess, and shogi (Japanese chess), using a known model of the environment. The **MuZero** method of [Sch+20] and the **Stochastic MuZero** method of [Ant+22] extend this to the case where the world model is also learned. The action-value functions for the intermediate nodes in the search tree are represented by deep neural networks, and updated using temporal difference methods that we discuss in Section 35.2. MCTS can also be applied to many other kinds of seqential decision problems, such as experiment design for sequentially creating molecules [SPW18].

35.4.1.3 Trajectory optimization for continuous actions

For continuous actions, we cannot enumerate all possible branches in the search tree. Instead, Equation (35.46) can be viewed as a nonlinear program, where $\boldsymbol{a}_{t:t+H-1}$ are the real-valued variables to be optimized. If the system dynamics are linear and the reward function corresponds to negative quadratic cost, the optimal action sequence can be solved mathematically, as in the **linear-quadratic-Gaussian** (**LQG**) controller (see e.g., [AM89; HR17]). However, this problem is hard in general and often solved by numerical methods such as **shooting** and **collocation** [Die+07; Rao10; Kal+11]. Many of them work in an iterative fashion, starting with an initial action sequence followed by a step to improve it. This process repeats until convergence of the cost.

An example is **differential dynamic programming** (**DDP**) [JM70; TL05]. In each iteration, DDP starts with a reference trajectory, and linearizes the system dynamics around states on the trajectory to form a locally quadratic approximation of the reward function. This system can be solved using LQG, whose optimal solution results in a new trajectory. The algorithm then moves to the next iteration, with the new trajectory as the reference trajectory.

Other alternatives are possible, including black-box (gradient-free) optimization methods like the cross-entropy method. (see Section 6.7.5).

35.4.2 Combining model-based and model-free

In Section 35.4.1, we discussed MPC, which uses the model to decide which action to take at each step. However, this can be slow, and can suffer from problems when the model is inaccurate. An alternative is to use the learned model to help reduce the sample complexity of policy learning.

There are many ways to do this. One approach is to generate rollouts from the model, and then train a policy or Q-function on the "hallucinated" data. This is the basis of the famous **dyna** method [Sut90]. In [Jan+19], they propose a similar method, but generate short rollouts from previously visited real states; this ensures the model only has to extrapolate locally.

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.

(a) (b)

Figure 35.7: (a) A cart-pole system being controlled by a policy learned by PILCO using just 17.5 seconds of real-world interaction. The goal state is marked by the red cross. The initial state is where the cart is stationary on the right edge of the workspace, and the pendulum is horizontal. For a video of the system learning, see https://bit.ly/35fpLmR. (b) A low-quality robot arm being controlled by a block-stacking policy learned by PILCO using just 230 seconds of real-world interaction. From Figures 11, 12 from [DFR15]. Used with kind permission of Marc Deisenroth.

35.4.3 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 $\boldsymbol{s}$ and $\boldsymbol{a}$ for states and actions to emphasize they are vectors.

35.4.3.1 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 $\boldsymbol{s}_{t+1} = f(\boldsymbol{s}_t, \boldsymbol{a}_t) + \boldsymbol{\epsilon}_t$, where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, and f is an unknown, continuous function.[4] 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, $J(\pi_{\boldsymbol{\theta}}) = \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{a}_t \sim \pi_{\boldsymbol{\theta}}} [c(\boldsymbol{s}_t)]$, where $\boldsymbol{s}_0 \sim p_0$. This function and its gradients wrt $\boldsymbol{\theta}$ 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 $\boldsymbol{\theta}$, instead of applying SGD to sampled trajectories. (See https://github.com/mathDR/jax-pilco for some JAX code.)

Due to its data efficiency, it is possible to apply PILCO to real robots. Figure 35.7a 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 $\boldsymbol{s} \in \mathbb{R}^4$ consists of the position x of the cart (with $x = 0$ being the center of the track), the

4. An alternative, which often works better, is to use f to model the residual, so that $\boldsymbol{s}_{t+1} = \boldsymbol{s}_t + f(\boldsymbol{s}_t, \boldsymbol{a}_t) + \boldsymbol{\epsilon}_t$.

velocity $\dot{x}$, the angle θ 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 $\boldsymbol{s}_* = (0, \star, \pi, \star)$, 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.[5]

Figure 35.7b 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 $\boldsymbol{s} \in \mathbb{R}^3$ is the 3d location of the center of the block in the arm's gripper (derived from an RGBD sensor), and the control $\boldsymbol{a} \in \mathbb{R}^4$ corresponds to the pulse widths of four servo motors. A linear policy was successfully trained using as few as 10 real world trials.

35.4.3.2 GP-MPC

[KD18a] have proposed an extension to PILCO that they call **GP-MPC**, since it combines a GP dynamics model with model predictive control (Section 35.4.1). In particular, they use an open-loop control policy to propose a sequence of actions, $\boldsymbol{a}_{t:t+H-1}$, as opposed to sampling them from a policy. They compute a Gaussian approximation to the future state trajectory, $p(\boldsymbol{s}_{t+1:t+H}|\boldsymbol{a}_{t:t+H-1}, \boldsymbol{s}_t)$, by moment matching, and use this to deterministically compute the expected reward and its gradient wrt $\boldsymbol{a}_{t:t+H-1}$ (as opposed to the policy parameters $\boldsymbol{\theta}$). Using this, they can solve Equation (35.46) to find $\boldsymbol{a}^*_{t:t+H-1}$; finally, they execute the first step of this plan, a^*_t, and repeat the whole process.

The advantage of GP-MPC over policy-based PILCO is that it can handle constraints more easily, and it can be more data efficient, since it continually updates the GP model after every step (instead of at the end of an trajectory).

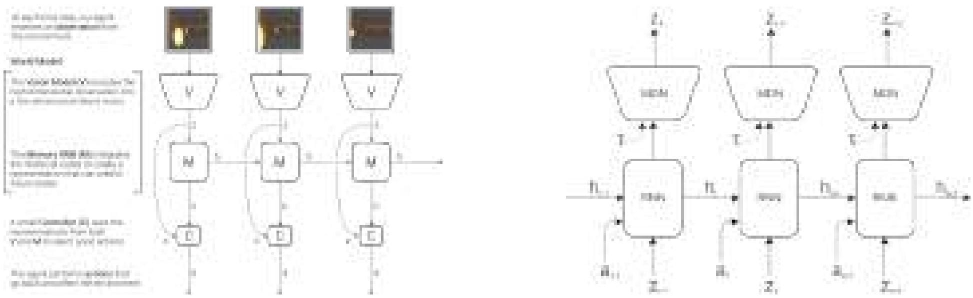
35.4.4 MBRL using DNNs

Gaussian processes do not scale well to large sample sizes and high dimensional data. Deep neural networks (DNNs) work much better in this regime. However, they do not naturally model uncertainty, which can cause MPC methods to fail. We discuss various methods for representing uncertainty with DNNs in Section 17.1. Here, we mention a few approaches that have been used for MBRL.

The **deep PILCO** method uses DNNs together with Monte Carlo dropout (Section 17.3.1) to represent uncertainty [GMAR16]. [Chu+18] proposed **probabilistic ensembles with trajectory sampling** or **PETS**, which represents uncertainty using an ensemble of DNNs (Section 17.3.9). Many other approaches are possible, depending on the details of the problem being tackled.

Since these are all stochastic methods (as opposed to the GP methods above), they can suffer from a high variance in the predicted returns, which can make it difficult for the MPC controller to pick the best action. We can reduce variance with the **common random number** trick [KSN99], where all rollouts share the same random seed, so differences in $J(\pi_{\boldsymbol{\theta}})$ can be attributed to changes in $\boldsymbol{\theta}$ but not other factors. This technique was used in **PEGASUS** [NJ00][6] and in [HMD18].

5. 2 random initial trials, each 5 seconds, and then 5 policy-generated trials, each 2.5 seconds, totalling 17.5 seeconds.
6. PEGASUS stands for "Policy Evaluation-of-Goodness And Search Using Scenarios", where the term "scenario" refers to one of the shared random samples.



(a) (b)

Figure 35.8: (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 τ 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.

35.4.5 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.

35.4.5.1 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 35.8. First, we collect some random experience, and use this to fit a VAE model (Section 21.2) to reduce the dimensionality of the images, $\boldsymbol{x}_t \in \mathbb{R}^{64\times 64\times 3}$, to a latent $\boldsymbol{z}_t \in \mathbb{R}^{64}$. Next, we train an RNN to predict $p(\boldsymbol{z}_{t+1}|\boldsymbol{z}_t, \boldsymbol{a}_t, \boldsymbol{h}_t)$, where $\boldsymbol{h}_t$ is the deterministic RNN state, and $\boldsymbol{a}_t$ 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 $\boldsymbol{z}_t$ and $\boldsymbol{h}_t$ as inputs; here $\boldsymbol{z}_t$ is a compact representation of the current frame, and $\boldsymbol{h}_t$ is a compact representation of the predicted distribution over $\boldsymbol{z}_{t+1}$.

The authors of [HS18] trained the controller using a derivative free optimizer called **CMA-ES** (covariance matrix adaptation evolutionary strategy, see Section 6.7.6.2). It can work better than policy gradient methods, as discussed in Section 35.3.6. However, it does not scale to high dimensions. To tackle this, the authors use a linear controller, which has only 867 parameters.[7] By contrast, VAE has 4.3M parameters and MDN-RNN 422k. Fortunately, these two models can be trained in an unsupervised way from random rollouts, so sample efficiency is less critical than when training the policy.

7. The input is a 32-dimensional $\boldsymbol{z}_t$ plus a 256-dimensional $\boldsymbol{h}_t$, and there are 3 outputs. So the number of parameters is $(32 + 256) \times 3 + 3 = 867$, to account for the weights and biases.

So far, we have described how to use the representation learned by the generative model as informative features for the controller, but the controller is still learned by interacting with the real world. Surprisingly, we can also train the controller entirely in "dream mode", in which the generated images from the VAE decoder at time t are fed as input to the VAE encoder at time $t+1$, and the MDN-RNN is trained to predict the next reward r_{t+1} as well as $\boldsymbol{z}_{t+1}$. Unfortunately, this method does not always work, since the model (which is trained in an unsupervised way) may fail to capture task-relevant features (due to underfitting) and may memorize task-irrelevant features (due to overfitting). The controller can learn to exploit weaknesses in the model (similar to an adversarial attack) and achieve high simulated reward, but such a controller may not work well when transferred to the real world.

One approach to combat this is to artificially increase the variance of the MDN model (by using a temperature parameter τ), in order to make the generated samples more stochastic. This forces the controller to be robust to large variations; the controller will then treat the real world as just another kind of noise. This is similar to the technique of domain randomization, which is sometimes used for sim-to-real applications; see e.g., [MAZA18].

35.4.5.2 PlaNet and Dreamer

In [HS18], they first learn the world model on random rollouts, and then train a controller. On harder problems, it is necessary to iterate these two steps, so the model can be trained on data collected by the controller, in an iterative fashion.

In this section, we describe one method of this kind, known as **PlaNet** [Haf+19]. PlaNet uses a POMDP model, where $\boldsymbol{z}_t$ are the latent states, $\boldsymbol{s}_t$ are the observations, $\boldsymbol{a}_t$ are the actions, and r_t are the rewards. It fits a recurrent state space model (Section 29.13.2) of the form $p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{a}_{t-1})p(\boldsymbol{s}_t|\boldsymbol{z}_t)p(r_t|\boldsymbol{z}_t)$ using variational inference, where the posterior is approximated by $q(\boldsymbol{z}_t|\boldsymbol{s}_{1:t}, \boldsymbol{a}_{1:t-1})$. After fitting the model to some random trajectories, the system uses the inference model to compute the current belief state, and then uses the cross entropy method to find an action sequence for the next H steps to maximize expected reward, by optimizing in latent space. The system then executes $\boldsymbol{a}_t^*$, updates the model, and repeats the whole process. To encourage the dynamics model to capture long term trajectories, they use the "latent overshooting" training method described in Section 29.13.3. The PlaNet method outperforms model-free methods, such as A3C (Section 35.3.3.1) and D4PG (Section 35.3.5), on various image-based continuous control tasks, illustrated in Figure 35.9.

Although PlaNet is sample efficient, it is not computationally efficient. For example, they use CEM with 1000 samples and 10 iterations to optimize trajectories with a horizon of length 12, which requires $120,000$ evaluations of the transition dynamics to choose a single action. [AY19] improve this by replacing CEM with differentiable CEM, and then optimize in a latent space of action sequences. This is much faster, but the results are not quite as good. However, since the whole policy is now differentiable, it can be fine-tuned using PPO (Section 35.3.4), which closes the performance gap at negligible cost.

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, $\pi(\boldsymbol{a}_t|\boldsymbol{z}_t)$, 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

Figure 35.9: 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.

MSE between predicted future reward and the TD-λ estimate (Section 35.2.2). 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.

35.4.6 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 19.2). 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 35.5.) Failing that, the model should at least be able to quantify its uncertainty (Section 19.3). 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]).

35.5 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 following a new policy must be considered. This section covers this important topic.

A key challenge in off-policy learning is that the data distribution is typically different from the desired one, and this mismatch must be dealt with. For example, the probability of visiting a state s at time t in a trajectory depends not only on the MDP's transition model, but also on the policy that is being followed. If we are to estimate $J(\pi)$, as defined in Equation (35.15), but the trajectories are generated by a different policy π', simply averaging rewards in the data gives us $J(\pi')$, not $J(\pi)$. We have to somehow correct for the gap, or "bias". Another challenge is that off-policy data can also make an algorithm unstable and divergent, which we will discuss in Section 35.5.3.

Removing distribution mismatches is not unique in off-policy learning, and is also needed in supervised learning to handle covariate shift (Section 19.2.3.1), and in causal effect estimation (Chapter 36), among others. Off-policy learning is also closely related to **offline reinforcement learning** (also called **batch reinforcement learning**): the former emphasizes the distributional mismatch between data and the agent's policy, and the latter emphasizes that the data is static and no further online interaction with the environment is allowed [LP03; EGW05; Lev+20]. Clearly, in the offline scenario with fixed data, off-policy learning is typically a critical technical component. Recently, several datasets have been prepared to facilitate empirical comparisons of offline RL methods (see e.g., [Gul+20; Fu+20]).

Finally, while this section focuses on MDPs, most methods can be simplified and adapted to the special case of contextual bandits (Section 34.4). In fact, off-policy methods have been successfully used in numerous industrial bandit applications (see e.g., [Li+10; Bot+13; SJ15; HLR16]).

35.5.1 Basic techniques

We start with four basic techniques, and will consider more sophisticated ones in subsequent sections. The off-policy data is assumed to be a collection of trajectories: $\mathcal{D} = \{\boldsymbol{\tau}^{(i)}\}_{1\leq i\leq n}$, where each trajectory is a sequence as before: $\boldsymbol{\tau}^{(i)} = (s_0^{(i)}, a_0^{(i)}, r_0^{(i)}, s_1^{(i)} \ldots)$. Here, the reward and next states are sampled according to the reward and transition models; the actions are chosen by a **behavior policy**, denoted π_b, which is different from the **target policy**, π_e, that the agent is evaluating or improving. When π_b is unknown, we are in a **behavior-agnostic off-policy** setting.

35.5.1.1 Direct method

A natural approach to off-policy learning starts with estimating the unknown reward and transition models of the MDP from off-policy data. This can be done using regression and density estimation methods on the reward and transition models, respectively, to obtain $\hat{R}$ and $\hat{P}$; see Section 35.4 for further discussions. These estimated models then give us an inexpensive way to (approximately) simulate the original MDP, and we can apply on-policy methods on the simulated data. This method directly models the outcome of taking an action in a state, thus the name **direct method**, and is sometimes known as **regression estimator** and **plug-in estimator**.

While the direct method is natural and sometimes effective, it has a few limitations. First, a small estimation error in the simulator has a compounding effect in long-horizon problems (or equivalently, when the discount factor γ is close to 1). Therefore, an agent that is optimized against an MDP simulator may overfit the estimation errors. Unfortunately, learning the MDP model, especially the transition model, is generally difficult, making the method limited in domains where $\hat{R}$ and $\hat{P}$ can be learned to high fidelity. See Section 35.4.6 for a related discussion.

35.5.1.2 Importance sampling

The second approach relies on importance sampling (IS) (Section 11.5) to correct for distributional mismatches in the off-policy data. To demonstrate the idea, consider the problem of estimating the target policy value $J(\pi_e)$ with a fixed horizon T. Correspondingly, the trajectories in $\mathcal{D}$ are also of length T. Then, the IS off-policy estimator, first adopted by [PSS00], is given by

$$\hat{J}_{\text{IS}}(\pi_e) \triangleq \frac{1}{n} \sum_{i=1}^{n} \frac{p(\boldsymbol{\tau}^{(i)}|\pi_e)}{p(\boldsymbol{\tau}^{(i)}|\pi_b)} \sum_{t=0}^{T-1} \gamma^t r_t^{(i)} \tag{35.47}$$

It can be verified that $\mathbb{E}_{\pi_b}\left[\hat{J}_{\text{IS}}(\pi_e)\right] = J(\pi_e)$, that is, $\hat{J}_{\text{IS}}(\pi_e)$ is **unbiased**, provided that $p(\boldsymbol{\tau}|\pi_b) > 0$ 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 data is sampled from π_b and not π_e. Furthermore, this ratio does *not* depend on the MDP models, because for any trajectory $\boldsymbol{\tau} = (s_0, a_0, r_0, s_1, \ldots, s_T)$, we have from Equation (34.74) that

$$\frac{p(\boldsymbol{\tau}|\pi_e)}{p(\boldsymbol{\tau}|\pi_b)} = \frac{p(s_0)\prod_{t=0}^{T-1} \pi_e(a_t|s_t) p_T(s_{t+1}|s_t, a_t) p_R(r_t|s_t, a_t, s_{t+1})}{p(s_0)\prod_{t=0}^{T-1} \pi_b(a_t|s_t) p_T(s_{t+1}|s_t, a_t) p_R(r_t|s_t, a_t, s_{t+1})} = \prod_{t=0}^{T-1} \frac{\pi_e(a_t|s_t)}{\pi_b(a_t|s_t)} \tag{35.48}$$

This simplification makes it easy to apply IS, as long as the target and behavior policies are known. If the behavior policy is unknown, we can estimate it from $\mathcal{D}$ (using, e.g., logistic regression or DNNs), and replace π_b by its estimate $\hat{\pi}_b$ in Equation (35.48). For convenience, define the **per-step importance ratio** at time t by $\rho_t(\boldsymbol{\tau}) \triangleq \pi_e(a_t|s_t)/\pi_b(a_t|s_t)$, and similarly, $\hat{\rho}_t(\boldsymbol{\tau}) \triangleq \pi_e(a_t|s_t)/\hat{\pi}_b(a_t|s_t)$.

Although IS can in principle eliminate distributional mismatches, in practice its usability is often limited by its potentially high variance. Indeed, the importance ratio in Equation (35.47) can be 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. One improvement is based on the observation that the reward r_t is independent of the trajectory beyond time t. This leads to a **per-decision importance sampling** variant that often yields lower variance (see Section 11.6.2 for a statistical motivation, and [LBB20] for a further discussion):

$$\hat{J}_{\text{PDIS}}(\pi_e) \triangleq \frac{1}{n} \sum_{i=1}^{n} \sum_{t=0}^{T-1} \prod_{t' \le t} \rho_{t'}(\boldsymbol{\tau}^{(i)}) \gamma^t r_t^{(i)} \tag{35.49}$$

There are many other variants such as self-normalized IS and truncated IS, both of which aim to reduce variance possibly at the cost of a small bias; precise expressions of these alternatives are found, e.g., in [Liu+18b]. In the next subsection, we will discuss another systematic way to improve IS.

IS may also be applied to improve a policy against the policy value given in Equation (35.15). However, directly applying the calculation of Equation (35.48) runs into a fundamental issue with IS, which we will discuss in Section 35.5.2. For now, we may consider the following approximation of policy value, averaging over the state distribution of the behavior policy:

$$J_b(\pi_{\boldsymbol{\theta}}) \triangleq \mathbb{E}_{p_\beta^\infty(s)}\left[V_\pi(s)\right] = \mathbb{E}_{p_\beta^\infty(s)}\left[\sum_a \pi_{\boldsymbol{\theta}}(a|s) Q_\pi(s, a)\right] \tag{35.50}$$

Differentiating this and ignoring the term $\nabla_{\boldsymbol{\theta}} Q_\pi(s, a)$, as suggested by [DWS12], gives a way to (approximately) estimate the **off-policy policy-gradient** using a one-step IS correction ratio:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J_b(\pi_{\boldsymbol{\theta}}) &\approx \mathbb{E}_{p_\beta^\infty(s)} \left[\sum_a \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s) Q_\pi(s,a) \right] \\ &= \mathbb{E}_{p_\beta^\infty(s)\beta(a|s)} \left[\frac{\pi_{\boldsymbol{\theta}}(a|s)}{\beta(a|s)} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) Q_\pi(s,a) \right]\end{aligned}$$

Finally, we note that in the tabular MDP case, there exists a policy π_* that is optimal in all states (Section 34.5.5). This policy maximizes J and J_b simultaneously, so Equation (35.50) can be a good proxy for Equation (35.15) as long as all states are "covered" by the behavior policy π_b. The situation is similar when the set of value functions or policies under consideration is sufficiently expressive: an example is a Q-learning like algorithm called Retrace [Mun+16; ASN20]. Unfortunately, in general when we work with parametric families of value functions or policies, such a uniform optimality is lost, and the distribution of states has a direct impact on the solution found by the algorithm. We will revisit this problem in Section 35.5.2.

35.5.1.3 Doubly robust

It is possible to combine the direct and importance sampling methods discussed previously. To develop intuition, consider the problem of estimating $J(\pi_e)$ in a contextual bandit (Section 34.4), that is, when $T = 1$ in $\mathcal{D}$. The **doubly robust** (DR) estimator is given by

$$\hat{J}_{\text{DR}}(\pi_e) \triangleq \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\pi_e(a_0^{(i)}|s_0^{(i)})}{\hat{\pi}_b(a_0^{(i)}|s_0^{(i)})} \left(r_0^{(i)} - \hat{Q}(s_0^{(i)}, a_0^{(i)}) \right) + \hat{V}(s_0^{(i)}) \right) \tag{35.51}$$

where $\hat{Q}$ is an estimate of Q_{π_e}, which can be obtained using methods discussed in Section 35.2, and $\hat{V}(s) = \mathbb{E}_{\pi_e(a|s)} \left[\hat{Q}(s,a) \right]$. If $\hat{\pi}_b = \pi_b$, the term $\hat{Q}$ is canceled by $\hat{V}$ on average, and we get the IS estimate that is unbiased; if $\hat{Q} = Q_{\pi_e}$, the term $\hat{Q}$ is canceled by the reward on average, and we get the estimator as in the direct method that is also unbiased. In other words, the estimator Equation (35.51) is unbiased, as long as one of the estimates, $\hat{\pi}_b$ and $\hat{Q}$, is right. This observation justifies the name doubly robust, which has its origin in causal inference (see e.g., [BR05]).

The above DR estimator may be extended to MDPs recursively, starting from the last step. Given a length-T trajectory $\boldsymbol{\tau}$, define $\hat{J}_{\text{DR}}[T] \triangleq 0$, and for $t < T$,

$$\hat{J}_{\text{DR}}[t] \triangleq \hat{V}(s_t) + \hat{\rho}_t(\boldsymbol{\tau}) \left(r_t + \gamma \hat{J}_{\text{DR}}[t+1] - \hat{Q}(s_t, a_t) \right) \tag{35.52}$$

where $\hat{Q}(s_t, a_t)$ is the estimated cumulative reward for the remaining $T - t$ steps. The DR estimator 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 verified (as an exercise) that the recursive definition is equivalent to

$$\hat{J}_{\text{DR}}[0] = \hat{V}(s_0) + \sum_{t=0}^{T-1} \left(\prod_{t'=0}^{t} \hat{\rho}_{t'}(\boldsymbol{\tau}) \right) \gamma^t \left(r_t + \gamma \hat{V}(s_{t+1}) - \hat{Q}(s_t, a_t) \right) \tag{35.53}$$

This form can be easily generalized to the infinite-horizon setting by letting $T \to \infty$ [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].

35.5.1.4 Behavior regularized method

The three methods discussed previously do not impose any constraint on the target policy π_e. Typically, the more different π_e is from π_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 35.3.4).

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 (35.14)) as follows

$$\mathcal{L}_1^{\text{DQN}}(\boldsymbol{w}) \triangleq \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(r + \gamma \max_{\pi:D(\pi,\pi_b)\leq\varepsilon} \mathbb{E}_{\pi(a'|s')}\left[Q_{\boldsymbol{w}^-}(s',a')\right] - Q_{\boldsymbol{w}}(s,a)\right)^2\right] \tag{35.54}$$

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:

$$\mathcal{L}_2^{\text{DQN}}(\boldsymbol{w}) \triangleq \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(r + \gamma \max_{\pi} \mathbb{E}_{\pi(a'|s')}\left[Q_{\boldsymbol{w}^-}(s',a')\right] - \alpha\gamma D(\pi(s'),\pi_b(s')) - Q_{\boldsymbol{w}}(s,a)\right)^2\right] \tag{35.55}$$

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_β^∞ and p^∞, which we will discuss next.

35.5.2 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] describes an example, where the state-action distributions under the behavior and target policies are identical, but the importance ratio of a trajectory grows exponentially large. It is now more convenient to assume the off-policy data consists of a set of transitions: $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{1 \le i \le m}$, where $(s_i, a_i) \sim p_{\mathcal{D}}$ (some fixed but unknown sampling distribution, such as p_β^∞), and r_i and s'_i are sampled from the MDP's reward and transition models. Given a policy π, we aim to 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 (Equation (35.15)) as

$$J(\pi) = \frac{1}{1-\gamma} \mathbb{E}_{p_\pi^\infty(s,a)} [R(s, a)] = \frac{1}{1-\gamma} \mathbb{E}_{p_\beta^\infty(s,a)} [\zeta_*(s, a) R(s, a)] \tag{35.56}$$

For simplicity, we assume the initial state distribution p_0 is known, or can be easily sampled from. This assumption is often easy to satisfy in practice.

The starting point is the following linear program formulation for any given π:

$$\max_{d \ge 0} - \mathrm{D_f}(d \| p_{\mathcal{D}})) \quad \text{s.t.} \quad d(s, a) = (1-\gamma)\mu_0(s)\pi(a|s) + \gamma \sum_{\bar{s},\bar{a}} p(s|\bar{s}, \bar{a}) d(\bar{s}, \bar{a}) \pi(a|s) \;\; \forall (s, a) \tag{35.57}$$

where $\mathrm{D_f}$ is the f-divergence (Section 2.7.1). The constraint is a variant of Equation (34.93), giving similar flow conditions in the space of $\mathcal{S} \times \mathcal{A}$ under policy π. Under mild conditions, p_π^∞ is only solution that satisfies the flow constraints, so the objective does not affect the solution, but will facilitate the derivation below. We can now obtain the Lagrangian, with multipliers $\{\nu(s, a)\}$, and use the change-of-variables $\zeta(s, a) = d(s, a)/p_{\mathcal{D}}(s, a)$ to obtain the following optimization problem:

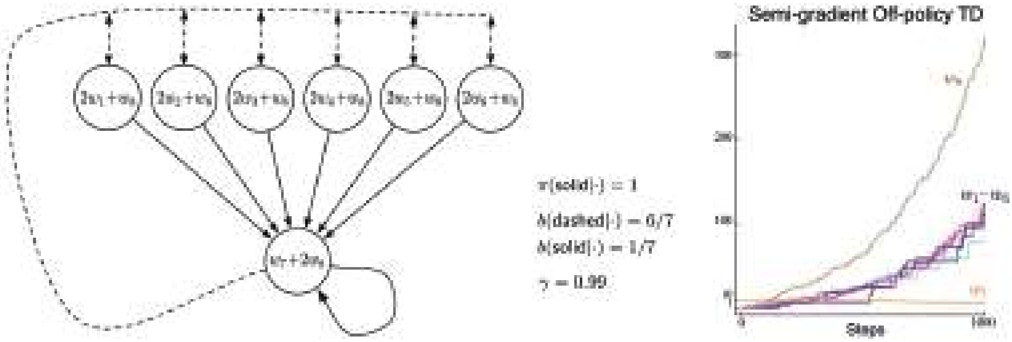
$$\begin{aligned} \max_{\zeta \ge 0} \min_\nu \mathcal{L}(\zeta, \nu) =& \mathbb{E}_{p_{\mathcal{D}}(s,a)} [-f(\zeta(s, a)] + (1-\gamma) \mathbb{E}_{p_0(s)\pi(a|s)} [\nu(s, a)] \\ &+ \mathbb{E}_{\pi(a'|s')p(s'|s,a)p_{\mathcal{D}}(s,a)} [\zeta(s, a) (\gamma \nu(s', a') - \nu(s, a))] \end{aligned} \tag{35.58}$$

It can be shown that the saddle point to Equation (35.58) must coincide with the desired correction ratio ζ_*. In practice, we may parameterize ζ and ν, and apply two-timescales stochastic gradient descent/ascent on the off-policy data $\mathcal{D}$ to solve for an approximate saddle-point. This is the **DualDICE** method [Nac+19a], which is extended to **GenDICE** [Zha+20c].

Compared to the IS or DR approaches, Equation (35.58) does not compute the importance ratio of a trajectory, thus generally has a lower variance. Furthermore, it is behavior-agnostic, without having to estimate the behavior policy, or even to assume data consists of a collection of trajectories. Finally, this approach can be extended to be doubly robust (e.g., [UHJ20]), and to optimize a policy [Nac+19b] against the true policy value $J(\pi)$ (as opposed to approximations like Equation (35.50)). For more examples along this line of approach, see [ND20] and the references therein.

35.5.3 The deadly triad

Other than introducing bias, off-policy data may also make a value-based RL method unstable and even divergent. Consider the simple MDP depicted in Figure 35.10a, due to [Bai95]. It has 7 states and 2 actions. Taking the dashed action takes the environment to the 6 upper states uniformly at random, while the solid action takes it to the bottom state. The reward is 0 in all transitions, and $\gamma = 0.99$. The value function $V_{\boldsymbol{w}}$ uses a linear parameterization indicated by the expressions shown inside the states, with $\boldsymbol{w} \in \mathbb{R}^8$. The target policies π always chooses the solid action in every state. Clearly, the true value function, $V_\pi(s) = 0$, can be exactly represented by setting $\boldsymbol{w} = \mathbf{0}$.



(a) (b)

Figure 35.10: (a) A simple MDP. (b) Parameters of the policy diverge over time. From Figures 11.1 and 11.2 of [SB18]. Used with kind permission of Richard Sutton.

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 TD(0) on this trajectory, the parameters diverge to ∞ (Figure 35.10b), even though the problem appears simple! In contrast, with on-policy data (that is, when b is the same as π), 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 TD, 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 (34.87)) may no longer hold when V is approximated by $V_{\boldsymbol{w}}$. 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 (34.82), 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]:

$$\min_{V,\pi} \mathcal{L}^{\mathrm{PCL}}(V,\pi) \triangleq \mathbb{E}\left[\frac{1}{2}\big(r + \gamma V(s') - \lambda \log \pi(a|s) - V(s)\big)^2\right] \tag{35.59}$$

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].

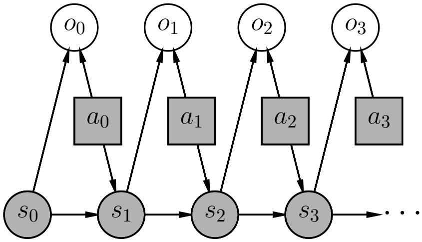


Figure 35.11: A graphical model for optimal control. States and actions are observed, while optimality variables are not. Adapted from Figure 1b of [Lev18].

This problem can be mitigated by introducing a dual function in the optimization [Dai+18]

$$\min_{V,\pi} \max_{\nu} \mathcal{L}^{\text{SBEED}}(V, \pi; \nu) \triangleq \mathbb{E}\left[\nu(s,a)\big(r + \gamma V(s') - \lambda \log \pi(a|s) - V(s)\big)^2 - \nu(s,a)^2/2\right] \tag{35.60}$$

where ν belongs to some function class (e.g., a DNN [Dai+18] or RKHS [FLL19]). It can be shown that optimizing Equation (35.60) forces ν 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.

35.6 Control as inference

In this section, we will discuss another approach to policy optimization, by reducing it to probabilistic inference. This is called **control as inference**, see e.g., [Att03; TS06; Tou09; BT12; KGO12; HR17; Lev18]. 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.

35.6.1 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 35.11 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:

$$p(o_t = 1|s_t, a_t) = \exp(\lambda^{-1} R(s_t, a_t)) \tag{35.61}$$

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 (35.61) 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 (35.61). Under these assumptions, the likelihood of observing a length-T trajectory $\boldsymbol{\tau}$, when optimality achieved in every step, is:

$$p(\boldsymbol{\tau}|\mathbf{o}_{0:T-1} = \mathbf{1}) \propto p(\boldsymbol{\tau}, \mathbf{o}_{0:T-1} = \mathbf{1}) \propto p(s_0) \prod_{t=0}^{T-1} p(o_t = 1|s_t, a_t) p_T(s_{t+1}|s_t, a_t)$$
$$= p(s_0) \prod_{t=0}^{T-1} p_T(s_{t+1}|s_t, a_t) \exp\left(\frac{1}{\lambda} \sum_{t=0}^{T-1} R(s_t, a_t)\right) \tag{35.62}$$

The intuition of Equation (35.62) 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, $p(\boldsymbol{\tau}|\mathbf{o}_{0:T-1} = \mathbf{1})$ is either proportional to $\exp(\lambda^{-1} \sum_{t=0}^{T-1} R(s_t, a_t))$ if $\boldsymbol{\tau}$ is feasible, or 0 otherwise. Maximizing reward is equivalent to inferring a trajectory with maximum $p(\boldsymbol{\tau}|\mathbf{o}_{0:T-1} = \mathbf{1})$.

The optimal policy in this probabilistic model is given by

$$p(a_t|s_t, \mathbf{o}_{t:T-1} = \mathbf{1}) = \frac{p(s_t, a_t|\mathbf{o}_{t:T-1} = \mathbf{1})}{p(s_t|\mathbf{o}_{t:T-1} = \mathbf{1})} = \frac{p(\mathbf{o}_{t:T-1} = \mathbf{1}|s_t, a_t) p(a_t|s_t) p(s_t)}{p(\mathbf{o}_{t:T-1} = \mathbf{1}|s_t) p(s_t)}$$
$$\propto \frac{p(\mathbf{o}_{t:T-1} = \mathbf{1}|s_t, a_t)}{p(\mathbf{o}_{t:T-1} = \mathbf{1}|s_t)} \tag{35.63}$$

The two probabilities in Equation (35.63) can be computed as follows, starting with $p(o_{T-1} = 1|s_{T-1}, a_{T-1}) = \exp(\lambda^{-1} R(s_{T-1}, a_{T-1}))$,

$$p(\mathbf{o}_{t:T-1} = \mathbf{1}|s_t, a_t) = \int_{\mathcal{S}} p(\mathbf{o}_{t+1:T-1} = \mathbf{1}|s_{t+1}) p_T(s_{t+1}|s_t, a_t) \exp(\lambda^{-1} R(s_t, a_t)) ds_{t+1} \tag{35.64}$$

$$p(\mathbf{o}_{t:T-1} = \mathbf{1}|s_t) = \int_{\mathcal{A}} p(\mathbf{o}_{t:T-1} = \mathbf{1}|s_t, a_t) p(a_t|s_t) da_t \tag{35.65}$$

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 $\boldsymbol{\tau}$ now becomes

$$p_\theta(\boldsymbol{\tau}) = p(s_1) \prod_{t=0}^{T-1} p_T(s_{t+1}|s_t, a_t) \pi_\theta(a_t|s_t) \tag{35.66}$$

If we optimize θ so that $D_{\mathbb{KL}}\left(p_\theta(\boldsymbol{\tau}) \parallel p(\boldsymbol{\tau}|\mathbf{o}_{0:T-1} = \mathbf{1})\right)$ is minimized, which can be simplified to

$$D_{\mathbb{KL}}\left(p_\theta(\boldsymbol{\tau}) \parallel p(\boldsymbol{\tau}|\mathbf{o}_{0:T-1} = \mathbf{1})\right) = -\mathbb{E}_{p_\theta}\left[\sum_{t=0}^{T-1} \lambda^{-1} R(s_t, a_t) + \mathbb{H}(\pi_{\boldsymbol{\theta}}(s_t))\right] + \text{const} \tag{35.67}$$

where the constant term only depends on the uniform action prior $p(a_t|s_t)$, but not $\boldsymbol{\theta}$. 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 $\pi_{\boldsymbol{\theta}}$ can represent all stochastic policies, a softmax version of the Bellman equation can be obtained for Equation (35.67):

$$Q_*(s_t, a_t) = \lambda^{-1} R(s_t, a_t) + \mathbb{E}_{p_T(s_{t+1}|s_t, a_t)}\left[\log \int_{\mathcal{A}} \exp(Q_*(s_{t+1}, a_{t+1})) da\right] \tag{35.68}$$

with the convention that $Q_*(s_T, a) = 0$ for all a, and the optimal policy has a softmax form: $\pi_*(a_t|s_t) \propto \exp(Q_*(s_t, a_t))$. Note that the Q_* above is different from the usual optimal Q-function (Equation (34.83)), due to the introduction of the entropy term. However, as $\lambda \to 0$, their difference vanishes, and the softmax policy becomes greedy, recovering the standard RL setting.

The **soft actor-critic** (**SAC**) algorithm [Haa+18b; Haa+18c] is an off-policy actor-critic method whose objective function is equivalent to Equation (35.67) (by taking T to ∞):

$$J^{\text{SAC}}(\boldsymbol{\theta}) \triangleq \mathbb{E}_{p^{\infty}_{\pi_{\boldsymbol{\theta}}}(s)\pi_{\boldsymbol{\theta}}(a|s)}\left[R(s,a) + \lambda\, \mathbb{H}(\pi_{\boldsymbol{\theta}}(s))\right] \tag{35.69}$$

Note that the entropy term has also the added benefit of encouraging exploration.

To compute the optimal policy, similar to other actor-critic algorithms, we will work with the "soft" state- and action-function approximations, parameterized by $\boldsymbol{w}$ and $\boldsymbol{u}$, respectively:

$$Q_{\boldsymbol{w}}(s,a) = R(s,a) + \gamma \mathbb{E}_{p_T(s'|s,a)}\left[V_{\boldsymbol{u}}(s',a') - \lambda \log \pi_{\boldsymbol{\theta}}(a'|s')\right] \tag{35.70}$$

$$V_{\boldsymbol{u}}(s,a) = \lambda \log \sum_a \exp(\lambda^{-1} Q_{\boldsymbol{w}}(s,a)) \tag{35.71}$$

This induces an improved policy (with entropy regularization): $\pi_{\boldsymbol{w}}(a|s) = \exp(\lambda^{-1}Q_{\boldsymbol{w}}(s,a))/Z_{\boldsymbol{w}}(s)$, where $Z_{\boldsymbol{w}}(s) = \sum_a \exp(\lambda^{-1}Q_{\boldsymbol{w}}(s,a))$ is the normalization constant. We then perform a soft policy improvement step to update $\boldsymbol{\theta}$ by minimizing $\mathbb{E}\left[D_{\mathbb{KL}}\left(\pi_{\boldsymbol{\theta}}(s) \parallel \pi_{\boldsymbol{w}}(s)\right)\right]$ where the expectation may be approximated by sampling s from a replay buffer D.

In [Haa+18c; Haa+18b], they show that the SAC method outperforms the off-policy DDPG algorithm (Section 35.3.5) and the on-policy PPO algorithm (Section 35.3.4) by a wide margin on various continuous control tasks. For more details, see [Haa+18c].

There is a variant of soft actor-critic, which only requires to model the action-value function. It is based on the observation that both the policy and soft value function can be induced by the soft action-value function as follows:

$$V_{\boldsymbol{w}}(s) = \lambda \log \sum_a \exp\left(\lambda^{-1} Q_{\boldsymbol{w}}(s,a)\right) \tag{35.72}$$

$$\pi_{\boldsymbol{w}}(a|s) = \exp\left(\lambda^{-1}(Q_{\boldsymbol{w}}(s,a) - V_{\boldsymbol{w}}(s))\right) \tag{35.73}$$

We then only need to learn $\boldsymbol{w}$, using approaches similar to DQN (Section 35.2.6). The resulting algorithm, **soft Q-learning** [SAC17], is convenient if the number of actions is small (when $\mathcal{A}$ is discrete), or if the integral in obtaining $V_{\boldsymbol{w}}$ from $Q_{\boldsymbol{w}}$ is easy to compute (when $\mathcal{A}$ is continuous).

It is interesting to see that algorithms derived in the maximum entropy RL framework bears a resemblance to PCL and SBEED in Section 35.5.3, both of which were to minimize an objective function resulting from the entropy-smoothed Bellman equation.

35.6.2 Other approaches

VIREL is an alternative model to maximum entropy RL [Fel+19]. Similar to soft actor-critic, it uses an approximate action-value function, $Q_{\boldsymbol{w}}$, a stochastic policy, $\pi_{\boldsymbol{\theta}}$, and a binary optimality random variable o_t at time t. A different probability model for o_t is used

$$p(o_t = 1|s_t, a_t) = \exp\left(\frac{Q_{\boldsymbol{w}}(s_t,a_t) - \max_a Q_{\boldsymbol{w}}(s_t,a)}{\lambda_{\boldsymbol{w}}}\right) \tag{35.74}$$

The temperature parameter $\lambda_{\boldsymbol{w}}$ is also part of the parameterization, and can be updated from data.

An EM method can be used to maximize the objective

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\theta}) = \mathbb{E}_{p(s)} \left[\mathbb{E}_{\pi_{\boldsymbol{\theta}}(a|s)} \left[\frac{Q_{\boldsymbol{w}}(s,a)}{\lambda_{\boldsymbol{w}}} \right] + \mathbb{H}(\pi_{\boldsymbol{\theta}}(s)) \right] \tag{35.75}$$

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 $\boldsymbol{w}$ is fixed, and the actor parameter $\boldsymbol{\theta}$ is updated using gradient accent with stepsize $\eta_{\boldsymbol{\theta}}$ (for policy improvement):

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\theta}) \tag{35.76}$$

In the M-step, the actor parameter is fixed, and the critic parameter is updated (for policy evaluation):

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta_{\boldsymbol{w}} \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\theta}) \tag{35.77}$$

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 $\boldsymbol{\theta}$:

$$J(\pi_{\boldsymbol{\theta}}) = \int G(\boldsymbol{\tau}) p(\boldsymbol{\tau}|\boldsymbol{\theta}) d\boldsymbol{\tau} \tag{35.78}$$

It can be interpreted as a pseudo-likelihood function, when the $G(\boldsymbol{\tau})$ 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 (35.67)), although the distribution involving $\boldsymbol{\theta}$ appears in the second argument of $D_{\mathbb{KL}}$. As discussed in Section 2.7.1, this forwards 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; SKM18; 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 $\boldsymbol{s}$, observations $\boldsymbol{y}$, and parameters $\boldsymbol{\theta}$. Following Section 10.1.1.1, we define the variational free energy to be $\mathcal{F}(\boldsymbol{o}) = D_{\mathbb{KL}}(q(\boldsymbol{s}, \boldsymbol{\theta}|\boldsymbol{y}) \parallel p(\boldsymbol{s}, \boldsymbol{y}, \boldsymbol{\theta}))$. State estimation corresponds to solving $\min_{q(\boldsymbol{s}|\boldsymbol{y})} \mathcal{F}(\boldsymbol{y})$, and parameter estimation corresponds to solving $\min_{q(\boldsymbol{\theta}|\boldsymbol{y})} \mathcal{F}(\boldsymbol{y})$, just as in variational Bayes EM (Section 10.3.5). (Minimizing the VFE for certain hierarchical Gaussian models also forms the foundation of predictive coding, which we discuss in Supplementary Section 8.1.4.)

To extend this to decision making problems we can define the **expected free energy** as $\overline{\mathcal{F}}(a) = \mathbb{E}_{q(\boldsymbol{y}|\boldsymbol{a})} [\mathcal{F}(\boldsymbol{y})]$, where $q(\boldsymbol{y}|\boldsymbol{a})$ is the posterior predictive distribution over observations given actions sequence $\boldsymbol{a}$. The connection to control as inference is explained in [Mil+20; WIP20; LÖW21].

35.6.3 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 rewards, but has access to a collection $\mathcal{D}_{\text{exp}}$ of trajectories generated by an expert policy π_{exp}; that is, $\boldsymbol{\tau} = (s_0, a_0, s_1, a_1, \ldots, s_T)$ and $a_t \sim \pi_{\text{exp}}(s_t)$ for $\boldsymbol{\tau} \in \mathcal{D}_{\text{exp}}$. The goal is to learn a good policy by imitating the expert, in the absence of reward signals. IL finds many applications in scenarios where we have demonstrations of experts (often humans) but designing a good reward function is not easy, such as car driving and conversational systems. See [Osa+18] for a survey up to 2018.

35.6.3.1 Imitation learning by behavior cloning

A natural method is **behavior cloning**, which reduces IL to supervised learning; see [Pom89] for an early application to autonomous driving. It interprets a policy as a classifier that maps states (inputs) to actions (labels), and finds a policy by minimizing the imitation error, such as

$$\min_{\pi} \mathbb{E}_{p^{\infty}_{\pi_{\text{exp}}}(s)} \left[D_{\mathbb{KL}} \left(\pi_{\text{exp}}(s) \parallel \pi(s) \right) \right] \tag{35.79}$$

where the expectation wrt $p^{\infty}_{\pi_{\text{exp}}}$ may be approximated by averaging over states in $\mathcal{D}_{\text{exp}}$. A challenge with this method is that the loss does not consider the sequential nature of IL: future state distribution is not fixed but instead depends on earlier actions. Therefore, if we learn a policy $\hat{\pi}$ that has a low imitation error under distribution $p^{\infty}_{\pi_{\text{exp}}}$, as defined in Equation (35.79), it may still incur a large error under distribution $p^{\infty}_{\hat{\pi}}$ (when the policy $\hat{\pi}$ is actually run). Further expert demonstrations or algorithmic augmentations are often needed to handle the distribution mismatch (see e.g., [DLM09; RGB11]).

35.6.3.2 Imitation learning by inverse reinforcement learning

An effective approach to IL is **inverse reinforcement learning** (IRL) or **inverse optimal control** (IOC). Here, we first infer a reward function that "explains" the observed expert trajectories, and then compute a (near-)optimal policy against this learned reward using any standard RL algorithms studied in earlier sections. The key step of reward learning (from expert trajectories) is the opposite of standard RL, thus called inverse RL [NR00a].

It is clear that there are infinitely many reward functions for which the expert policy is optimal, for example by several optimality-preserving transformations [NHR99]. To address this challenge, we can follow the maximum entropy principle (Section 2.4.7), and use an energy-based probability model to capture how expert trajectories are generated [Zie+08]:

$$p(\boldsymbol{\tau}) \propto \exp\Big(\sum_{t=0}^{T-1} R_{\boldsymbol{\theta}}(s_t, a_t) \Big) \tag{35.80}$$

where $R_{\boldsymbol{\theta}}$ is an unknown reward function with parameter $\boldsymbol{\theta}$. Abusing notation slightly, we denote by $R_{\boldsymbol{\theta}}(\boldsymbol{\tau}) = \sum_{t=0}^{T-1} R_{\boldsymbol{\theta}}(s_t, a_t))$ the cumulative reward along the trajectory $\boldsymbol{\tau}$. This model assigns exponentially small probabilities to trajectories with lower cumulative rewards. The partition function, $Z_{\boldsymbol{\theta}} \triangleq \int_{\boldsymbol{\tau}} \exp(R_{\boldsymbol{\theta}}(\boldsymbol{\tau}))$, is in general intractable to compute, and must be approximated. Here, we can take a sample-based approach. Let $\mathcal{D}_{\text{exp}}$ and $\mathcal{D}$ be the sets of trajectories generated by an expert, and by some known distribution q, respectively. We may infer $\boldsymbol{\theta}$ by maximizing the likelihood, $p(\mathcal{D}_{\text{exp}}|\boldsymbol{\theta})$, or equivalently, minimizing the negative log-likelihood loss

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{|\mathcal{D}_{\text{exp}}|} \sum_{\boldsymbol{\tau} \in \mathcal{D}_{\text{exp}}} R_{\boldsymbol{\theta}}(\boldsymbol{\tau}) + \log \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{\tau} \in \mathcal{D}} \frac{\exp(R_{\boldsymbol{\theta}}(\boldsymbol{\tau}))}{q(\boldsymbol{\tau})} \tag{35.81}$$

The term inside the log of the loss is an importance sampling estimate of Z that is unbiased as long as $q(\boldsymbol{\tau}) > 0$ for all $\boldsymbol{\tau}$. However, in order to reduce the variance, we can choose q adaptively as $\boldsymbol{\theta}$ is being updated. The optimal sampling distribution (Section 11.5), $q_*(\boldsymbol{\tau}) \propto \exp(R_{\boldsymbol{\theta}}(\boldsymbol{\tau}))$, 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 35.6.1. Interestingly, the process above produces the inferred reward $R_{\boldsymbol{\theta}}$ as well as an approximate optimal policy $\hat{\pi}$. This approach is used by **guided cost learning** [FLA16], and found effective in robotics applications.

35.6.3.3 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 π, so that $p_\pi^\infty(s,a)$ and $p_{\pi_{\text{exp}}}^\infty(s,a)$ are close, then π receives similar long-term reward as π_{exp}, and is a good imitation of π_{exp} in this regard. A number of IL algorithms find π by minimizing the divergence between p_π^∞ 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 $\mathrm{D_f}$ the f-divergence (Section 2.7.1). From the above intuition, we want to minimize $\mathrm{D_f}\left(p_{\pi_{\text{exp}}}^\infty \middle\| p_\pi^\infty\right)$. Then, using a variational approximation of $\mathrm{D_f}$ [NWJ10a], we can solve the following optimization problem for π:

$$\min_\pi \max_{\boldsymbol{w}} \mathbb{E}_{p_{\pi_{\text{exp}}}^\infty(s,a)}\left[T_{\boldsymbol{w}}(s,a)\right] - \mathbb{E}_{p_\pi^\infty(s,a)}\left[f^*(T_{\boldsymbol{w}}(s,a))\right] \tag{35.82}$$

where $T_{\boldsymbol{w}} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a function parameterizd by $\boldsymbol{w}$. 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 π. Furthermore, to implement this algorithm, we often use a parametric policy representation $\pi_{\boldsymbol{\theta}}$, and then perform stochastic gradient updates to find a saddle-point to Equation (35.82).

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 π to approximate the second expectation in Equation (35.82). 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 π. Hence, we need to correct the mismatch between p_π^∞ and the off-policy trajectory distribution, for which techniques from Section 35.5 can be used. An example is **ValueDICE** [KNT20], which uses a similar distribution correction method of DualDICE (Section 35.5.2).

36 Causality

This chapter is written by Victor Veitch and Alex D'Amour.

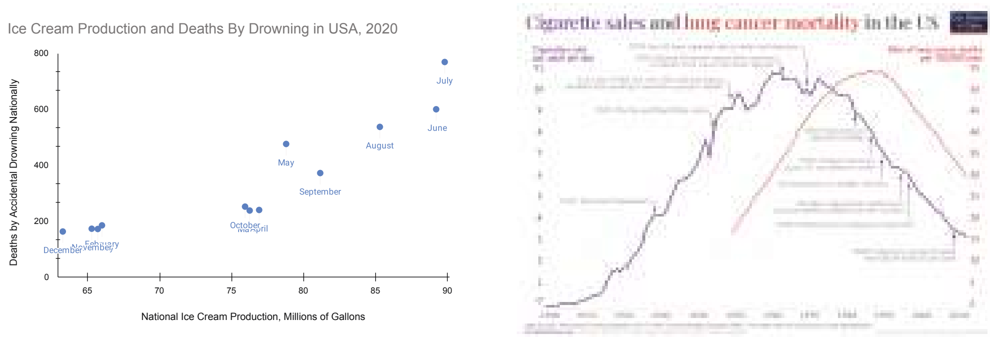
36.1 Introduction

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`. For more information on the connections between ML and causal inference, see e.g., [Kad+22; Xia+21a].

To make the gap between observed data modeling and causal inference concrete, consider the relationships depicted in Figure 36.1a and Figure 36.1b. Figure 36.1a shows the relationship between deaths by drowning and ice cream production in the United States in 1931 (the pattern holds across most years). Figure 36.1b 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?

Our goal in this chapter to introduce the essentials of estimating causal effects. The high-level



(a) (b)

Figure 36.1: Correlation is not causation. (a) Ice cream production is strongly associated with deaths by drowning. Ice cream production data from the US Department of Agriculture National Agricultural Statistics Service. Drowning data from the National Center for Health Statistics at the United States Centers for Disease Control. (b) Smoking is strongly associated with lung cancer. From `ourworldindata.org/smoking-big-problem-in-brief`. *Used with kind permission of Max Roser.*

approach has three steps.

- **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 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.

36.2 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.

36.2.1 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 towards health), and G_i, an indicator for whether the person has a genetic predisposition towards cancer. Suppose we observe a dataset of these variables drawn independently and identically from a population, $(A_i, Y_i, H_i) \overset{\text{iid}}{\sim} P^{\text{obs}}$, where "obs" stands for "observed".

In standard practice, we model data like these using probabilistic models. Notably, there are many different ways to specify a probabilistic model for the same observed distribution. For example, we could write a probabilistic model for P^{obs} as

$$A \sim P^{\text{obs}}(A) \tag{36.1}$$

$$H|A \sim P^{\text{obs}}(H|A) \tag{36.2}$$

$$Y|A, H \sim P^{\text{obs}}(Y|H, A) \tag{36.3}$$

$$G|A, H, Y \sim P^{\text{obs}}(G|A, H, Y) \tag{36.4}$$

This is a valid factorization, and sampling variables in this order would yield valid samples from the joint distribution P^{obs}. However, this factorization does not map well to a mechanistic understanding of how these variables are causally related in the world. In particular, it is perhaps more plausible that health-consciousness H causally precedes smoking status A, since a person's health-consciousness would influence their decision to smoke.

These intuitions about causal ordering are intimately tied to the notion of intervention. Here, we will focus on a notion of intervention that can be represented in terms of "structural" models that describe mechanistic relationships between variables. The fundamental objects that we will reason about are **structural causal models**, or SCM's. SCM's resemble probabilistic models, but they encode additional assumptions (see also Section 4.7). Specifically, SCM's serve two purposes: they describe a probabilistic model *and* they provide semantics for transforming the data-generating process through intervention.

Formally, SCM's describe a mechanistic data generating process with an ordered sequence of equations that resemble assignment operations in a program. Each variable in a system is determined by combining other modeled variables (the causes) with exogenous "noise" according to some (unknown) deterministic function. For instance, a plausible SCM for P^{obs} might be

$$G \leftarrow f_G(\xi_0) \tag{36.5}$$
$$H \leftarrow f_H(\xi_1) \tag{36.6}$$
$$A \leftarrow f_A(H, \xi_2) \tag{36.7}$$
$$Y \leftarrow f_Y(G, H, A, \xi_3) \tag{36.8}$$

where the (unknown) functions f are fixed, and the variables ξ are unmeasured causes, modeled as independent random "noise" variables. Conceptually, the functions f_G, f_H, f_A, f_Y describe deterministic physical relationships in the real world, while the variables ξ are hidden causes that are sufficient to distinguish each unit i in the population. Because we assume that each observed unit i is drawn at random from the population, we model ξ as random noise.

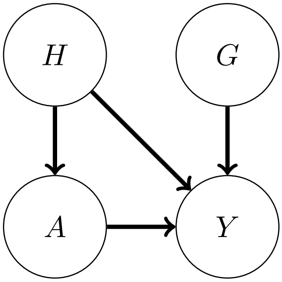
SCM's imply probabilistic models, but not the other way around. For example, our example SCM implies probabilistic model for the observed data based on the factorization $P^{\text{obs}}(G, H, A, Y) = P^{\text{obs}}(G)P^{\text{obs}}(H)P^{\text{obs}}(A \mid H)P^{\text{obs}}(Y \mid A, H)$. Thus, we could sample from the SCM in the same way we would from a probabilistic model: draw a set of noise variables ξ and evaluate each assignment operation in the SCM in order.

Beyond the probabilistic model, an SCM encodes additional assumptions about the effects of **interventions**. This can be formalized using the **do-calculus** (as in the verb "to do"), which we describe in Section 36.8; But in brief, interventions are represented by replacing assignment statements. For 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 $A \leftarrow 0$. We would denote this by $P(Y|\text{do}(A = 0), H)$. 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. (See also Section 4.7.3.)

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.

36.2.2 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 36.2. In this way, causal DAGs are related to SCMs in the same way that probabilistic graphical models (PGMs) are related



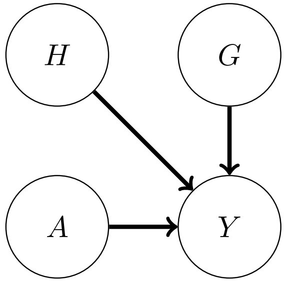


Figure 36.2: (a) Causal graph illustrating relationships between smoking A, cancer Y, health conciousness H, and genetic cancer pre-disposition G. (b) "Mutilated" causal graph illustrating relationships under an intervention on smoking A.

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 4.2.4.1). 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.

Conceptually, the difference between causal DAGs and PGMs is that probabilistic graphical models encode our assumptions about statistical relationships, whereas causal graphs encode our (stronger) assumptions about causal relationships. Such causal relationships can be used to derive how statistical relationships would change under intervention.

Causal graphs also allow us to reason about the causal and non-causal origins of statistical dependencies in observed data without specifying a full SCM. In a causal graph, two variables — say, A and D — can be statistically associated in different ways. First, there can be a directed path from (ancestor) A to (descendant) D. In this case, A is a causal ancestor of D and interventions on A will propagate through to change D; $\mathrm{P}(D|\mathrm{do}(A = a)) \neq \mathrm{P}(D|\mathrm{do}(A = a'))$. For example, smoking is a causal ancestor of cancer in our example. Alternatively, A and D could share a common cause — there is some variable C such that there is a directed path from C to A and from C to D. If A and D are associated only through such a path then interventions on A will not change the distribution of D. However, it is still the case that $\mathrm{P}(D|A = a) \neq \mathrm{P}(D|A = a')$ — observing different values of A changes our guess for the value of D. The reason is that A carries information about C, which carries information about D. For example, suppose we lived in a world where there was no effect of smoking on developing cancer (e.g., everybody vapes), there would nevertheless be an association between smoking and cancer because of the path $A \leftarrow H \rightarrow Y$. The existence of such "backdoor paths" is one core reason that statistical and causal association are not the same. Of course, more complicated variants of these associations are possible — e.g., C is itself only associated with A through a backdoor path — but this already captures the key distinction between causal and non-causal paths.

Recall that our aim in introducing SCMs and causal graphs is to enable us to formalize our causal knowledge of the world and to make precise what interventional quantities we'd like to estimate. Writing down a causal graph gives a simple formal way to encode our knowledge of the causal structure of a problem. Usefully, this causal structure is sufficient to directly reason about the implications of interventions without fully specifying the underlying SCM. The key observation is that if a variable A is intervened on then, after intervention, none of the other variables are causes of A. That is, when we replace a line of an SCM with a statement directly assigning a variable a particular value, we cut off all dependencies that variable had on its causal parents. Accordingly, in the causal graph, the intervened on variable has no parents. This leads us to the **graph surgery** notion of intervention: an intervention that sets A to a is the operation that deletes all incoming edges to A in the graph, and then conditions on $A = a$ in the resulting probability distribution (which is defined by the conditional independence structure of the post-surgery graph). We'll use Pearl's do notation to denote this operation. $\mathrm{P}(\mathbf{X}|\mathrm{do}(A = a))$ is the distribution of $\mathbf{X}$ given $A = a$ under the mutilated graph that results from deleting all edges going into A. Similarly, $\mathbb{E}[\mathbf{X}|\mathrm{do}(A = a)] \triangleq \mathbb{E}_{\mathrm{P}(\mathbf{X}|\mathrm{do}(A=a))}[\mathbf{X}]$. Thus, we can formalize statements such as "the average effect of receiving drug A" as

$$\mathrm{ATE} = E[Y|\mathrm{do}(A = 1)] - \mathbb{E}[Y|\mathrm{do}(A = 0)], \tag{36.9}$$

where ATE stands for average treatment effect.

For concreteness, consider our running example. We contrast the distribution that results by conditioning on A with the distribution that results from intervening on A:

$$\mathrm{P}(Y, H, G|A = a) = \mathrm{P}(Y|H, G, A = a)\mathrm{P}(G)\mathrm{P}(H|A = a) \tag{36.10}$$

$$\mathrm{P}(Y, H, G|\mathrm{do}(A = a)) = \mathrm{P}(Y|H, G, A = a)\mathrm{P}(G)\mathrm{P}(H) \tag{36.11}$$

The key difference between these two distributions is that the standard conditional distribution describes a population where health consciousness H has the distribution that we observe among individuals with smoking status $A = a$, while the interventional distribution described a population where health consciousness H follows the marginal distribution among all individuals. For example, we would expect $P(H \mid A = \texttt{smoker})$ to put more mass on lower values of H than the marginal health consciousness distribution than the marginal distribution $P(H)$, which would also include non-smokers. The intervention distribution thus incorporates a hypothesis of how smoking would affect the subpopulation individuals who tend to be too health conscious to smoke in the observed data.

36.2.3 Identification

A central challenge in causal inference is that many different SCM's can produce identical distributions of observed data. This means that, on the basis of observed data alone, we cannot uniquely identify the SCM that generated it. This is true no matter how large of a data sample is available to us.

For example, consider the setting where there is a treatment A that may or may not have an effect on outcome Y, and where both the treatment and outcome are known to be affected by some *unobserved* common binary cause U. Now, we might be interested in the causal estimand $E[Y|\mathrm{do}(A = 1)]$. In general, we can't learn this quantity from the observed data. The problem is that, we can't tell apart the case where the treatment has a strong effect from the case where the treatment has no effect, but $U = 1$ both causes people to tend to be treated and increases the probability of a positive outcome. The same observation shows we can't learn the (more complicated) interventional distribution $\mathrm{P}(Y|\mathrm{do}(A=1))$ (if we could learn this, then we'd get the average effect automatically).

Thus, an important part of causal inference is to augment the observed data with knowledge about the underlying causal structure of the process under consideration. Often, these assumptions can narrow the space of SCM's sufficiently so that there is only one value of the causal estimand that is compatible with the observed data. We say that the causal estimand is **identified** or **identifiable** under a given set of assumptions if those assumptions are sufficient to provide a unique answer. There are many different sets of sufficient conditions that yield identifiable causal effects; we call each set of sufficient conditions an **identification strategy**.

Given a set of assumptions about the underlying SCM, the most common way to show that a causal estimand is identified is by construction. Specifically, if the causal estimand can be written entirely in terms of observable probability distributions, then it is identifed. We call such a function of observed distributions a **statistical estimand**. Once such a statistical estimand has been recovered, we can then construct and analyze an estimator for that quantity using standard statistical tools. As an example of a statistical estimand, in the SCM above, it can be shown the ATE as defined in Equation (36.9), is equal to the following statistical estimand

$$\mathrm{ATE} \overset{(*)}{=} \tau^{\mathrm{ATE}} \triangleq \mathbb{E}[\mathbb{E}[Y|H, A=1] - \mathbb{E}[Y|H, A=0]], \tag{36.12}$$

where the equality $(*)$ only holds because of some specific properties of the SCM. Note that the RHS above only involves conditional expectations between observed variables (there are no do operators), so τ^{ATE} is only a function of observable probability distributions.

There are many kinds of assumptions we might make about the SCM governing the process under consideration. For example, the following are assertions we might make about the system in our running example:

1. The probability of developing cancer is additive on the logit scale in A, G, and H (i.e., logistic regression is a well-specified model).

2. For each individual, smoking can never decrease the probability of developing cancer.

3. Whether someone smokes is influenced by their health consciousness H, but not by their genetic predisposition to cancer G.

These assumptions range from strong parametric assumptions fully specifying the form of the SCM equations, to non-parametric assumptions that only specify what the inputs to each equation are, leaving the form fully unspecified. Typically, assumptions that fully specify the parametric form are very strong, and would require far more detailed knowledge of the system under consideration than we actually have. The goal in identification arguments is to find a set of assumptions that are weak enough that they might be plausibly true for the system under consideration, but which are also strong enough to allow for identification of the causal effect.

If we are not willing to make any assumptions about the functional form of the SCM, then our assumptions are just about which variables affect (and do not affect) the other variables. In this sense, such which-affects-which assumptions are minimal. These assumptions are exactly the assumptions captured by writing down a (possibly incomplete) causal DAG, showing which variables are parents of each other variable. The graph may be incomplete because we may not know whether each possible edge is present in the physical system. For example, we might be unsure whether the gene G actually has a causal effect on health consciousness H. It is natural to ask to what extent we can identify causal effects only on the basis of partially specified causal DAGs. It turns out much progress can be made based on such non-parametric assumptions; we discuss this in detail in Section 36.8.

We will also discuss certain assumptions that cannot be encoded in a causal graph, but that are still weaker than assuming that full functional forms are known. For example, we might assume that the outcome is affected additively by the treatment and any confounders, with no interaction terms between them. These weaker assumptions can enable causal identification even when assuming the causal graph alone does not.

It is worth emphasizing that every causal identification strategy relies on assumptions that have some content that cannot be validated in the observed data. This follows directly from the ill-posedness of causal problems: if the assumptions used to identify causal quantities could be validated, that would imply that the causal estimand was identifiable from the observed data alone. However, since we know that there are many values of the causal estimand that are compatible with observed data, it follows that the assumptions in our identification strategy must have unobservable implications.

36.2.4 Counterfactuals and the causal hierarchy

Structural causal models let us formalize and study a hierarchy of different kinds of query about the system under consideration. The most familiar is observational queries: questions that are purely about statistical associations (e.g., "Are smoking and lung cancer associated in the population this sample was drawn from?"). Next is interventional queries: questions about causal relationships at the population level (e.g., "How much does smoking increase the probability of cancer in a given population?"). The rest of this chapter is focused on the defintion, identification, and estimation of interventional queries. Finally, there are counterfactual queries: questions about causal relationships at the level of specific individuals, had something been different (e.g., "Would Alice have developed cancer had she not smoked?"). This causal hierarchy was popularized by [Pea09a, Ch. 1].

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 uncertainity about the "noise" variables ξ 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., $P(Y|\mathrm{do}(A=0))$. 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 ξ_i through differently intervened SCMs. Counterfactual outcomes are often written in terms of **potential outcomes** notation. In our running smoking example, this would look like:

$$Y_i(a) \triangleq f_Y(G_i, H_i, a, \xi_{3,i}). \tag{36.13}$$

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:

$$\text{ATE} = \mathbb{E}[Y_i(1) - Y_i(0)]. \tag{36.14}$$

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:

$$\mathbb{E}[Y_i(0) \mid Y_i(1) = 1, A_i = 1] \tag{36.15}$$

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: $\mathbb{E}[Y \mid \text{do}(A = 1)] - \mathbb{E}[Y \mid \text{do}(A = 0)]$.

Counterfactual queries require categorically more assumptions for identification than interventional ones. For identifying interventional queries, knowing the DAG structure of an SCM is often sufficient, while for counterfactual queries, some assumptions about the functional forms in the SCM are necessary. This is because only one potential outcome is ever observed for each individual, so the dependence between potential outcomes within individuals is not observable. For example, the data in our running example provide no information on how individual-level smoking and non-smoking cancer risk are related. Thus, answering a question like "Did smokers who developed cancer have lower non-smoking cancer risk than smokers who did not develop cancer?", requires additional assumptions about how characteristics encoded in ξ_i are translated to cancer outcomes. To answer this question without such assumptions, we would need to observe smokers who developed cancer in the alternate world where they did not smoke. Because they compare how individuals would have turned out under different generating processes, counterfactual queries are often referred to as "cross-world" quantities. On the other hand, interventional queries only require understanding the marginal distributions of potential outcomes $Y_i(0)$ and $Y_i(1)$ across individuals; thus, no cross-world information is necessary at the individual level.

We conclude this section by noting that counterfactual outcomes and potential outcomes notation are often conceptually useful, even if they are not used to explicitly answer counterfactual queries. Many causal queries are more intuitive to formalize in terms of potential outcomes. E.g., "Would I have smoked if I was more health conscious?" may be more intuitive than "Would a randomly sampled individual from the same population have smoked had they been subject to an intervention that made them more health concious?". In fact, some schools of causal inference use potential outcomes, rather than DAGs, as their primary conceptual building block [See IR15]. Causal graphs and potential outcomes both provide ways to formulate interventional queries and causal assumptions. Ultimately, these are mathematically equivalent. Nevertheless, practically, they have different strengths. The main advantage of potential outcomes is that counterfactual statements often map more directly to our mechanistic understanding of the world. This can make it easier to articulate causal desiderata and causal assumptions we may wish to use. On the other hand, the potential outcomes notation does not automatically distinguish between interventional and counterfactual queries. Additionally, causal graphs often give an intuitive and easy way of articulating assumptions about structural causal models involving many variables—potential outcomes get quickly unwieldly. In short: both formalizations have distinct advantages, and those advantages are simply about how easy it is to translate our causal understanding of the world into crisp mathematical assumptions.

36.3 Randomized control trials

We now turn to the business of estimating causal effects from data. We begin with **randomized control trials**, which are experiments designed to make the causal concerns as simple as possible.

The simplest situation for causal estimation is when there are no common causes of A and Y. The world is rarely so obliging as to make this the case. However, sometimes we can design an experiment to enforce the no-common-causes structure. In randomized control trials we assign each participant to either the treatment or control group at random. Because random assignment does not depend on any property of the units in the study, there are no causes of treatment assignment, and hence also no common causes of Y and A.

In this case, it's straightforward to see that $\mathrm{P}(Y|\mathrm{do}(A = a) = \mathrm{P}(Y|a)$. This is essentially by 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,

$$\mathrm{ATE} = E[Y|\mathrm{do}(A = 1)] - \mathbb{E}[Y|\mathrm{do}(A = 0)]. \tag{36.16}$$

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 τ^{RCT} of the observational distribution:

$$\tau^{\mathrm{RCT}} = \mathbb{E}[Y|A = 1] - \mathbb{E}[Y|A = 0].$$

Then, a natural estimator is:

$$\hat{\tau}^{\mathrm{RCT}} \triangleq \frac{1}{n_A} \sum_{i:A_i=1} Y_i - \frac{1}{n - n_A} \sum_{i:A_i=0} Y_i, \tag{36.17}$$

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.[1]

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

1. 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.

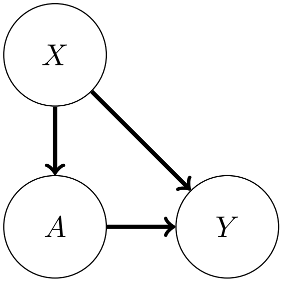


Figure 36.3: A causal DAG illustrating a situation where treatment A and outcome Y are both influenced by observed confounders X.

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.

36.4 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.

36.4.1 Causal estimand, statistical estimand, and identification

Consider a problem where we observe treatment A, outcome Y, and covariates X, which are drawn iid from some unknown distribution P. We wish to learn the average treatment effect: the expected difference between being assigned treatment and assigned no treatment for a randomly chosen member of the population. Following the discussion in the introduction, there are three steps to learning this quantity: mathematically formalize the causal estimand, give conditions for the causal estimand to be identified as a statistical estimand, and, finally, estimate this statistical estimand from data. We now turn to the first two steps.

The average treatment effect is defined to be the difference between the average outcome if we *intervened* and set A to be 0, versus the average outcome if we intervented and set A to be 1. Using the do notation, we can write this formally as

$$\text{ATE} = \mathbb{E}[Y|\text{do}(A = 1)] - \mathbb{E}[Y|\text{do}(A = 0)]. \tag{36.18}$$

The next step is to articulate sufficient conditions for the ATE to be identified as a statistical estimand (a parameter of distribution P). The key issue is the possible presence of **confounders**. Confounders are "common cause" variables that affect both the treatment and outcome. When there are confounding variables in observed data, the sub-population of people who are observed to have received one level of the treatment A will differ from the rest of the population in ways that are relevant to their observed Y. For example, there is a strong positive association between horseback riding in childhood (treatment) and healthiness as an adult (outcome) [RB16]. However, both of these quantities are influenced by wealth X. The population of people who rode horses as children ($A = 1$) is wealthier than the population of people who did not. Accordingly, the horseback-riding population will have better health outcomes even if there is no actual causal benefit of horseback riding for adult health.

We'll express the assumptions required for causal identification in the form of a causal DAG. Namely, we consider the simple triangle DAG in Figure 36.3, where the treatment and outcome are influenced by *observed* confounders X. It turns out that the assumption encoded by this DAG suffices for identification. To understand why this is so, recall that the target causal effect is defined according to the distribution we would see if the edge from X to A was removed (that's the meaning of do). The key insight is that because the intervention only modifies the relationship between X and A, the structural equation that generates outcomes Y given X and A, illustrated in Figure 36.3 as the $A \to Y \leftarrow X$, is the same even after the $X \to A$ edge is removed. For example, we might believe that the physiological processes by which smoking status A and confounders X produce lung cancer Y remain the same, regardless of how the decision to smoke or not smoke was made. Second, 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 36.3. In particular, X contains all common causes of A and Y and no variable in X is caused by A or Y.*

2. *(Overlap) $0 < \mathrm{P}(A = 1|X = x) < 1$ 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 $\mathrm{ATE} = \tau$, *where*

$$\tau = \mathbb{E}[\mathbb{E}[Y|A = 1, X]] - \mathbb{E}[\mathbb{E}[Y|A = 0, X]]. \tag{36.19}$$

Proof. First, we expand the ATE using the tower property of expectation, conditioning on X. Then, we apply the invariances discussed above:

$$\begin{aligned} ATE &= \mathbb{E}[Y|\mathrm{do}(A = 1)] - \mathbb{E}[Y|\mathrm{do}(A = 0)] && (36.20)\\ &= \mathbb{E}[\mathbb{E}[Y|\mathrm{do}(A = 1), X]] - \mathbb{E}[\mathbb{E}[Y|\mathrm{do}(A = 0), X]] && (36.21)\\ &= \mathbb{E}[\mathbb{E}[Y|A = 1, X]] - \mathbb{E}[\mathbb{E}[Y|A = 0, X]] && (36.22) \end{aligned}$$

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 $\mathrm{P}(Y|\mathrm{do}(A=1), X)$, so the equality holds.

The condition that $0 < \mathrm{P}(A=1|X=x) < 1$ is required for the first equality (the tower property) to be well defined. □

Note that Equation (36.19) 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 τ 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 τ explicitly as an integral over X:

$$\tau = \int \mathbb{E}[Y \mid A=1, X]P(X)dX - \int \mathbb{E}[Y \mid A=0, X]P(X)dX \tag{36.23}$$

We can write the naive estimand in a similar form by applying the tower property of expectation:

$$\mathbb{E}[Y \mid A=1] - \mathbb{E}[Y \mid A=0] = \int \mathbb{E}[Y \mid A=1, X]P(X \mid A=1)dX - \int \mathbb{E}[Y \mid A=0, X]P(X \mid A=0)dX \tag{36.24}$$

The key difference is the probability distribuiton over X that is being integrated over. The observational difference in means integrates over the distinct conditional distributions of confounders X, depending on the value of A. On the other hand, in the ATE estimand τ, we integrate over the same distribution $P(X)$ for both levels of the treatment.

Overlap In addition to the assumption on the causal structure, identification requires that there is sufficient random variation in how treatments are assigned.

Definition 1. *A distribution P on A, X satisfies* **overlap** *if $0 < P(A=1|x) < 1$ for all x. It satisfies* **strict overlap** *if $\epsilon < \mathrm{P}(A=1|x) < 1-\epsilon$ for all x and some $\epsilon > 0$.*

Overlap is the requirement that any unit could have either recieved the treatment or not.

To see the necessity of overlap, consider estimating the effectiveness of a drug in a study where patient sex is a confounder, but the drug was only ever prescribed to male patients. Then, conditional on a patient being female, we would know that patient was assigned to control. Without further assumptions, it's impossible to know the effect of the drug on a population with female patients, because there would be no data to inform the expected outcome for treated female patients, that is, $\mathbb{E}[Y \mid A=1, X=\texttt{female}]$. In this case, the statistical estimand equation 36.19 would not be identifiable. In the same vein, strict overlap ensures that the conditional distributions at each stratum of X can be estimated in finite samples.

Overlap can be particularly limiting in settings where we are adjusting for a large number of covariates (in an effort to satisfy no unobserved confounding). Then, certain combinations of traits may be very highly predictive of treatment assignment, even if individual traits are not. E.g., male patients over age 70 with BMI greater than 25 are very rarely assigned the drug. If such groups represent a significant fraction of the target population, or have significantly different treatment effects, then this issue can be problematic. In this case, the strict overlap assumption puts very strong restrictions on observational studies: for an observational study to satisfy overlap, most dimensions of the confounders X would need to closely mimic the balance we would expect in an RCT [D'A+21].

36.4.2 ATE estimation with observed confounders

We now return to estimating the ATE using observed — i.e., not experimental — data. We've shown that in the case where we observe all common causes of the treatment and outcome, the ATE is causally identified with a statistical estimand τ. We now consider several strategies for estimating this quantity using a finite data sample. Broadly, these techniques are known as backdoor adjustment.[2]

Recall that the defining characteristic of a confounding variable is that it affects both treatment and outcome. Thus, an adjustment strategy may aim to account for the influence of confounders on the observed outcome, the influence of confounders on treatment, or both. We discuss each of these strategies in turn.

36.4.2.1 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.[3] To begin, we define

Definition 2. *The* **conditional expected outcome** *is the function Q given by*

$$Q(a,x) = \mathbb{E}[Y|A=a, X=x]. \tag{36.25}$$

Substituting this definition into the definition of our estimand τ, Equation (36.19), we have $\tau = \mathbb{E}[Q(1,x) - Q(0,x)]$. This suggests a procedure for estimating τ: fit a model $\hat{Q}$ for Q and then report

$$\hat{\tau}^Q \triangleq \frac{1}{n}\sum_i \hat{Q}(1,x_i) - \hat{Q}(0,x_i). \tag{36.26}$$

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.[4] 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 $\hat{Q}(0,x_i)$.[5] If the unit actually did receive treatment ($a_i = 1$)

2. As we discuss in Section 36.8, this backdoor adjustment references the estimand returned by the do-calculus to eliminate confounding from a backdoor path. This also generalizes the approaches discussed here to some cases where we do not observe all common causes.

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 minimzation taken over measurable functions.

5. This interpretation is justified by the same conditions as Theorem 2.

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 $\mathrm{P}(X)$ — the true population distribution of the confounders — using the empirical distribution $\hat{\mathrm{P}}(X) = 1/n \sum_i \delta_{x_i}$. 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 β_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.

36.4.2.2 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 treatment. This strategy adjusts for confounding by directly addressing sampling bias in the treated and control groups. This bias arises from the relationship between the confounders and the treatment. Intuitively, the effect of confounding may be viewed as due to the difference between $\mathrm{P}(X|A = 1)$ and $\mathrm{P}(X|A = 0)$ — e.g., the population of people who rode horses as children is wealthier than the population of people who did not. When we observe all confounding variables X, this degree of over- or under-representation can be adjusted away by reweighting samples such that the confounders X have the same distribution in the treated and control groups. When the confounders are balanced between the two groups, then any differences between them must be attributable to the treatment.

A key quantity for balancing treatment and control groups is the **propensity score**, which summarises the relationship between confounders and treatment.

Definition 3. *The* **propensity score** *is the function g given by $g(x) = \mathrm{P}(A = 1|X = x)$.*

To make use of the propensity score in adjustment, we first rewrite the estimand τ in a suggestive form, leveraging the fact that $A \in \{0, 1\}$:

$$\tau = \mathbb{E}[\frac{YA}{g(X)} - \frac{Y(1-A)}{1-g(X)}]. \tag{36.27}$$

This identity can be verified by noting that $\mathbb{E}[YA|X] = \mathbb{E}[Y|A = 1, X]\mathrm{P}(A = 1|X) + 0$, rearranging for $\mathbb{E}[Y|A = 1, X]$, doing the same for $\mathbb{E}[Y|A = 0, X]$, and substituting in to Equation (36.19). Note that the identity is just a mathematical fact about the statistical estimand — it does not rely on any causal assumptions, and holds whether or not τ can be interpreted as a causal effect.

This expression suggests the **inverse probability of treatment weighted estimator**, or IPTW estimator:

$$\hat{\tau}^{\text{IPTW}} \triangleq \frac{1}{n} \sum_i \frac{Y_i A_i}{\hat{g}(X_i)} - \frac{Y_i(1 - A_i)}{1 - \hat{g}(X_i)}. \tag{36.28}$$

Here, $\hat{g}$ is an estimate of the propensity score function. Recall from Section 14.2.1 that if a model is well-specified and the loss function is a proper scoring rule then risk minimizer $g^* = \operatorname{argmin}_g \mathbb{E}[L(A, g(X))]$ will be $g^*(X) = \text{P}(A = 1|X)$. That is, we can estimate the propensity score by fitting a model that predicts A from X. Cross-entropy and squared loss are both proper scoring rules, so we may use standard supervised learning methods.

In summary, the procedure is to estimate the propensity score function (with machine learning), and then to plug the estimated propensity scores $\hat{g}(x_i)$ into Equation (36.28). The IPTW estimator computes a difference of weighted averages between the treated and untreated group. The effect is to upweight the outcomes of units who were unlikely to be treated but who nevertheless actually, by chance, recieved treatment (and similarly for untreated). Intuitively, such units are typical for the untreated population. So, their outcomes under treatment are informative about what would have happened had a typical untreated unit received treatment.

A word of warning is in order. Although the IPTW is asymtotically valid and popular in practice, it can be very unstable in finite samples. If estimated propensity scores are extreme for some values of x (that is, very close to 0 or 1), then the corresponding IPTW weights can be very large, resulting in a high-variance estimator. In some cases, this instability can be mitigated by instead using the Hajek version of the estimator.

$$\hat{\tau}^{\text{h}-\text{IPTW}} \triangleq \sum_i Y_i A_i \frac{1/\hat{g}(X_i)}{\sum_i A_i/\hat{g}(X_i)} - \sum_i Y_i(1 - A_i) \frac{1/(1-\hat{g}(X_i))}{\sum_i (1-A_i)/(1-\hat{g}(X_i))}. \tag{36.29}$$

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.

36.4.2.3 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 τ 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 τ, 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

$$\hat{\tau}^{\text{AIPTW}} \triangleq \frac{1}{n}\sum_i \hat{Q}(1, X_i) - \hat{Q}(0, X_i) + A_i \frac{Y_i - \hat{Q}(1, X_i)}{\hat{g}(X_i)} - (1 - A_i)\frac{Y_i - \hat{Q}(0, X_i)}{1 - \hat{g}(X_i)}. \tag{36.30}$$

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 refered 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 τ to be the function ϕ defined by[6]

$$\phi(X_i, A_i, Y_i; Q, g, \tau) \triangleq Q(1, X_i) - Q(0, X_i) + A_i \frac{Y_i - Q(1, X_i)}{g(x_i)} - (1 - A_i)\frac{Y_i - Q(0, X_i)}{1 - g(X_i)} - \tau. \tag{36.31}$$

By design, $\hat{\tau}^{\text{AIPTW}} - \tau = \frac{1}{n}\sum_i \phi(\mathbf{X}_i; \hat{Q}, \hat{g}, \tau)$, where $\mathbf{X}_i = (X_i, A_i, Y_i)$. 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 $\hat{\tau}^{\text{ideal}} = \frac{1}{n}\sum_i \phi(\mathbf{X}_i; Q, g, \tau) + \tau$ and, by the central limit theorem, we would have:

$$\sqrt{n}(\hat{\tau}^{\text{ideal}} - \tau) \xrightarrow{d} \text{Normal}(0, \mathbb{E}[\phi(\mathbf{X}_i; Q, g, \tau)^2]). \tag{36.32}$$

This result characterizes the estimation uncertainity in the best possible case. If we knew Q and g, we could rely on this result for, e.g., finding confidence intervals for our estimate.

The question is: what happens when Q and g need to be estimated? For general estimators and nuisance function models, we don't expect the $\sqrt{n}$-rate of Equation (36.32) to hold. For instance, $\sqrt{n}(\hat{\tau}^Q - \tau)$ only converges if $\sqrt{n}\mathbb{E}[(\hat{Q} - Q)^2]^{\frac{1}{2}} \to 0$. That is, for the naive estimator we only get the $\sqrt{n}$ rate for estimating τ if we can also estimate Q at the $\sqrt{n}$ rate — a much harder task! This is the issue that the double machine learning estimator helps with.

To understand how, we decompose the error in estimating τ as follows:

$$\sqrt{n}(\hat{\tau}^{\text{AIPTW}} - \tau) \tag{36.33}$$
$$= \frac{1}{\sqrt{n}}\sum_i \phi(\mathbf{X}_i; Q, g, \tau) \tag{36.34}$$
$$+ \frac{1}{\sqrt{n}}\sum_i \phi(\mathbf{X}_i; \hat{Q}, \hat{g}, \tau) - \phi(\mathbf{X}_i; Q, g, \tau) - \mathbb{E}[\phi(\mathbf{X}; \hat{Q}, \hat{g}, \tau) - \phi(\mathbf{X}; Q, g, \tau)] \tag{36.35}$$
$$+ \sqrt{n}\mathbb{E}[\phi(\mathbf{X}; \hat{Q}, \hat{g}, \tau) - \phi(\mathbf{X}; Q, g, \tau)] \tag{36.36}$$

We recognize the first term, Equation (36.34), as $\sqrt{n}(\hat{\tau}^{\text{ideal}} - \tau)$, the estimation error in the optimal case where we know Q and g. Ideally, we'd like the error of $\hat{\tau}^{\text{AIPTW}}$ to be asymptotically equal to this ideal case—which will happen if the other two terms go to 0.

6. 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.

The second term, Equation (36.35), is a penalty we pay for using the same data to estimate Q, g and to compute τ. For many model classes, it can be shown that such "empirical process" terms go to 0. This can also be guaranteed in general by using different data for fitting the nuisance functions and for computing the estimator (see the next section).

The third term, Equation (36.36), captures the penalty we pay for misestimating the nuisance functions. This is where the particular form of the AIPTW is key. With a little algebra, we can show that

$$\mathbb{E}[\phi(\mathbf{X}; \hat{Q}, \hat{g}) - \phi(\mathbf{X}; Q, g)] = \mathbb{E}[\frac{1}{g(X)}(\hat{g}(X) - g(X))(\hat{Q}(1, X) - Q(1, X)) \tag{36.37}$$

$$+ \frac{1}{1 - g(X)}(\hat{g}(X) - g(X))(\hat{Q}(0, X) - Q(0, X))]. \tag{36.38}$$

The important point is that estimation errors of Q and g are multiplied together. Using the Cauchy-Schwarz inequality, we find that $\sqrt{n}\mathbb{E}[\phi(\mathbf{X}; \hat{Q}, \hat{g}) - \phi(\mathbf{X}; Q, g)] \to 0$ as long as $\sqrt{n} \max_a \mathbb{E}[(\hat{Q}(a, X) - Q(a, X))^2]^{\frac{1}{2}} \mathbb{E}[(\hat{g}(X) - g(X))^2]^{\frac{1}{2}} \to 0$. That is, the misestimation penalty will vanish so long as the *product* of the misestimation errors is $o(\sqrt{n})$. For example, this means that τ can be estimated at the (optimal) $\sqrt{n}$ rate even when the estimation error of each of Q and g only decreases as $o(n^{-1/4})$.

The upshot here is that the double machine learning estimator has the special property that the weak condition $\sqrt{n}\mathbb{E}(\hat{Q}(T, X) - Q(T, X))^2 \mathbb{E}(\hat{g}(X) - g(X))^2 \to 0$ suffices to imply that

$$\sqrt{n}(\hat{\tau}^{\text{AIPTW}} - \tau) \xrightarrow{d} \text{Normal}(0, \mathbb{E}[\phi(\mathbf{X}_i; Q, g, \tau)^2]) \tag{36.39}$$

(though strictly speaking this requires some additional technical conditions we haven't discussed). This is *not* true for the earlier estimators we discussed, which require a much faster rate of convergence for the nuisance function estimation.

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 **doubly robust**: the estimator is consistent (converges to the true τ as $n \to \infty$) as long as at least one of either $\hat{Q}$ or $\hat{g}$ is consistent.

36.4.2.4 Cross fitting

The term Equation (36.35) 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 τ by plugging $\{\hat{Q}(a_i, x_i), \hat{g}(x_i)\}_i$ into Equation (36.30). 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.

36.4.3 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 uncertainity in our estimate. For example, in the form of a confidence interval. The asymptotic normality of $\sqrt{n}\hat{\tau}$ (Equation (36.39)) 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 τ. 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 uncertainity by simply plugging in our fitted nuisance models and our point estimate of τ into

$$\hat{\mathbb{V}}[\hat{\tau}] = 1/n \sum_i \phi(\mathbf{X}_i; \hat{Q}, \hat{g}, \hat{\tau})^2. \tag{36.40}$$

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 τ as $\hat{\tau} \pm 1.96\sqrt{\hat{\mathbb{V}}[\hat{\tau}]/n}$.

Alternatively, we could quantify the uncertainity 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.

36.4.4 Matching

One particularly popular approach to adjustment-based causal estimation is **matching**. Intuitively, the idea is to match each treated to unit to an untreated unit that has the same (or at least similar) values of the confounding variables and then compare the observed outcomes of the treated unit and its matched control. If we match on the full set of common causes, then the difference in outcomes is, intuitively, a noisy estimate of the effect the treatment had on that treated unit. We'll now build this up a bit more carefully. In the process we'll see that matching can be understood as, essentially, a particular kind of outcome model adjustment.

For simplicity, consider the case where X is a discrete random variable. Define $\mathcal{A}_x$ to be the set of treated units with covariate value x, and $\mathcal{C}_x$ to be the set of untreated units with covariate value x. In this case, the matching estimator is:

$$\hat{\tau}^{\text{matching}} = \sum_x \hat{\mathrm{P}}(x)\big(\frac{1}{|\mathcal{A}_x|}\sum_{i\in\mathcal{A}_x} Y_i - \frac{1}{|\mathcal{C}_x|}\sum_{j\in\mathcal{C}_x} Y_j\big), \tag{36.41}$$

where $\hat{\mathrm{P}}(x)$ is an estimator of $\mathrm{P}(X = x)$ — e.g., the fraction of units with $X = x$. Now, we can rewrite $Y_i = Q(A_i, X_i) + \xi_i$ where ξ_i is a unit-specific noise term defined by the equation. In particular, we have that $\mathbb{E}[\xi_i | A_i, X_i] = 0$. Substituting this in, we have:

$$\hat{\tau}^{\text{matching}} = \sum_x \hat{\mathrm{P}}(x)\big(Q(1,x) - Q(0,x)\big) + \sum_x \hat{\mathrm{P}}(x)\big(\frac{1}{|\mathcal{A}_x|}\sum_{i\in\mathcal{A}_x} \xi_i - \frac{1}{|\mathcal{C}_x|}\sum_{j\in\mathcal{C}_x} \xi_j\big). \tag{36.42}$$

We can recognize the first term as an estimator of usual target parameter τ (it will be equal to τ if $\hat{\mathrm{P}}(x) = \mathrm{P}(x)$). The second term is a difference of averages of random variables with expectation 0, 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 more data. Thus, we see that the matching estimator is a particular way of estimating the parameter τ. The procedure can be extended to continuous covariates by introducing some notion of values of X being close, and then matching close treatment and control variables.

There are two points we should emphasize here. First, notice that the argument here has nothing to do with causal identification. Matching is a particular technique for estimating the observational parameter τ. Whether or not τ can be interpreted as an average treatment effect is determined by the conditions of Theorem 2 — the particular estimation strategy doesn't say anything about this. Second, notice that in essence matching amounts to a particular choice of model for $\hat{Q}$. Namely, $\hat{Q}(1, x) = \frac{1}{|\mathcal{A}_x|} \sum_{i \in \mathcal{A}_x} Y_i$ and similarly for $\hat{Q}(0, x)$. That is, we estimate the conditional expected outcome as a sample mean over units with the same covariate value. Whether this is a good idea depends on the quality of our model for Q. In situations where better models are possible (e.g., a machine-learning model fits the data well), we might expect to get a more accurate estimate by using the conditional expected outcome predictor directly.

There is another important case we mention in passing. In general, when using adjustment based identification, it suffices to adjust for any function $\phi(X)$ of X such that $A \perp\!\!\!\perp X | \phi(X)$. To see that adjusting for only $\phi(X)$ suffices, first notice that $g(X) = \mathrm{P}(A = 1|X) = \mathrm{P}(A = 1|\phi(X))$ only depends on $\phi(X)$, and then recall that can write the target parameter as $\tau = \mathbb{E}[\frac{YA}{g(X)} - \frac{Y(1-A)}{1-g(X)}]$, whence τ only depends on X through $g(X)$. That is: replacing X by a reduced version $\phi(X)$ such that $g(X) = \mathrm{P}(A = 1|\phi(X))$ can't make any difference to τ. Indeed, the most popular choice of $\phi(X)$ is 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 τ, and says nothing about whether it's valid to interpret τ as a causal effect.

36.4.5 Practical considerations and procedures

when performing causal analysis, many issues can arise in practice, some of which we discuss below.

36.4.5.1 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 criterion can be generalized somewhat — see Section 36.8.

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 is 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

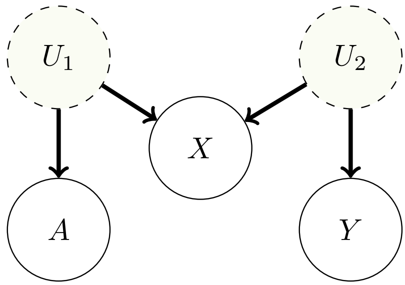


Figure 36.4: The M-bias causal graph. Here, A and Y are not confounded. However, conditioning on the covariate X opens a backdoor path, passing through U_1 and U_2 (because X is a colider). Thus, adjusting for X creates bias. This is true even though X need not be a pre-treatment variable.

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 on variables that are downstream of the treatment A. A natural response is to this is to limit conditioning to all pre-treatment variables, or those that are causally upstream of the treatment. Importantly, if there is a valid adjustment set in the observed covariates X, then there will also be a valid adjustment set among the pre-treatment covariates. This is because any open backdoor path between A and Y must include a parent of A, and the set of pre-treatment covariates includes these parents. However, it is still possible that conditioning on the full set of pre-treatment variables can induce new backdoor paths between A and Y through colliders. In particular, if there is a covariate D that is separately confounded with the treatment A and the outcome Y then D is a collider, and conditioning on D opens a new backdoor path. This phenomenon is known as m-bias because of the shape of the graph [Pea09c], see Figure 36.4.

A practical refinement of the pre-treatment variable heuristic is given in VanderWeele and Shpitser [VS11]. Their heuristic suggests conditioning on all pre-treatment variables that are causes of the treatment, outcome, or both. The essential qualifier in this heuristic is that the variable is causally upstream of treatment and/or outcome. This eliminates the possibility of conditioning on covariates that are only confounded with treatment and outcome, avoiding m-bias. Notably, this heuristic requires more causal knowledge than the above heuristics, but does not require detailed knowledge of how different covariates are causally related to each other.

The VanderWeele and Shpitser [VS11] criterion is a useful rule of thumb, but other practical considerations often arise. For example, if one has more knowledge about the causal structure among covariates, it is possible to optimize adjustment sets to minimize the variance of the resulting estimator [RS20]. One important example of reducing variance by pruning adjustment sets is the exclusion of variables that are known to only be a parent of the treatment, and not of the outcome (so called instruments, as discussed in Section 36.5).

Finally, adjustment set selection criteria operate under the assumption that there actually exists a valid adjustment set among observed covariates. When there is no set of observed covariates in X that block all backdoor paths, then any adjusted estimate will be biased. Importantly, in this case, the bias does not necessarily decrease as one conditions on more variables. For example, conditioning on an instrumental variable often results in an estimate that has higher bias, in addition to the higher variance discussed above. This phenomenon is known as bias amplification or z-bias; see Section 36.7.2. A general rule of thumb is that variables that explain away much more variation in the treatment than in the outcome can potentially amplify bias, and should be treated with caution.

36.4.5.2 Overlap

Recall that in addition to no-unobserved-confounders, identification of the average treatment effect requires overlap: the condition that $0 < P(A = 1|x) < 1$ for the population distribution P. With infinite data, any amount of overlap will suffice for estimating the causal effect. In realistic settings, even near failures can be problematic. Equation (36.39) gives an expression for the (asymptotic) 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 proportional to $1/g(X)$ and $1/(1 - g(X)$. Accordingly, the variance of our estimator will balloon if there are units where $g(x) \approx 0$ or $g(x) \approx 1$ (unless such units are rare enough that they don't contribute much to the expectation).

In practice, a simple way to deal with potential overlap violation is to fit a model $\hat{g}$ for the treatment assignment probability — which we need to do anyways — and check that the values $\hat{g}(x)$ are not too extreme. In the case that some values are too extreme, the simplest resolution is to cheat. 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.

36.4.5.3 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**,

$$\text{ATT} \triangleq \mathbb{E}_{X|A=1}[\mathbb{E}[Y|X, \text{do}(A = 1)] - E[Y|X, \text{do}(A = 0)]], \tag{36.43}$$

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 36.3, and the overlap condition $\mathrm{P}(A = 1|X = x) < 1$ for all $X = x$ then the ATT is causally identified as

$$\tau^{\mathrm{ATT}} = \mathbb{E}_{X|A=1}[\mathbb{E}[Y|A = 1, X] - E[Y|A = 0, X]]. \tag{36.44}$$

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 datapoints 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

$$\hat{\tau}^{\mathrm{ATT-AIPTW}} \triangleq \frac{1}{n}\sum_i \frac{A_i}{\mathrm{P}(A = 1)}(Y_i - \hat{Q}(0, X_i)) - \frac{(1 - A_i)\hat{g}(X_i)}{\mathrm{P}(A = 1)(1 - \hat{g}(X_i))}(Y_i - \hat{Q}(0, X_i). \tag{36.45}$$

. The variance of this estimator can be estimated by

$$\begin{aligned}\phi^{\mathrm{ATT}}(\mathbf{X}_i; Q, g, \tau) \triangleq \frac{1}{n}\sum_i \Bigg[&\frac{A_i}{\mathrm{P}(A = 1)}(Y_i - \hat{Q}(0, X_i)) \\ &- \frac{(1 - A_i)\hat{g}(X_i)}{\mathrm{P}(A = 1)(1 - \hat{g}(X_i))}(Y_i - \hat{Q}(0, X_i) - \frac{A_i\tau}{\mathrm{P}(A = 1)}\Bigg] \end{aligned}\tag{36.46}$$

$$\hat{\mathbb{V}}[\hat{\tau}^{\mathrm{ATT-AIPTW}}] \triangleq \frac{1}{n}\sum_i \phi^{\mathrm{ATT}}(\mathbf{X}_i; \hat{Q}, \hat{g}, \hat{\tau}^{\mathrm{ATT-AIPTW}})^2. \tag{36.47}$$

Notice that the estimator for the ATT doesn't require estimating $Q(1, X)$. This can be a considerable advantage when the treated units are rare. See Chernozhukov et al. [Che+17e] for details.

36.4.6 Summary and practical advice

We have seen a number of estimators that follow the general procedure:

1. Fit statistical or machine-learning models $\hat{Q}(a, x)$ as a predictor for Y, and/or $\hat{g}(x)$ as a predictor for A

2. Compute the predictions $\hat{Q}(0, x_i), \hat{Q}(1, x_i), \hat{g}(x_i)$ for each datapoint, and

3. Combine these predictions into an estimate of the average treatment effect.

Importantly, no single estimation approach is a silver bullet. For example, the double machine-learning estimator has appealing theoretical properties, such as asymptotic efficiency guarantees and a recipe for estimating uncertainity without needing to bootstrap the model fitting. However, in terms of the quality of point estimates, the double ML estimators can sometimes underperform their more naive counterparts [KS07]. In fact, there are cases where each of outcome regression, propensity weighting, or doubly robust methods will outperform the others.

One difficulty in choosing an estimator in practice is that there are fewer guardrails in causal inference than there are in standard predictive modeling. In predictive modeling, we construct a train-test split and validate our prediction models using the true labels or outcomes in the held-out dataset. However, for causal problems, the causal estimands are functionals of a different data-generating process from the one that we actually observed. As a result, it is impossible to empirically validate many aspects of causal estimation using standard techniques.

The effectiveness of a given approach is often determined by how much we trust the specification of our propensity score or outcome regression models $\hat{g}(x)$ and $\hat{Q}(a,x)$, and how well the treatment and control groups overlap in the dataset. Using flexible models for the nuisance functions g and Q can alleviate some of the concerns about model misspecification, but our freedom to use such models is often constrained by dataset size. When we have the luxury of large data, we can use flexible models; on the other hand, when the dataset is relatively small, we may need to use a smaller parametric family or stringent regularization to obtain stable estimates of Q and g. Similarly, if overlap is poor in some regions of the covariate space, then flexible models for Q may be highly variable, and inverse propensity score weights may be large. In these cases, IPTW or AIPTW estimates may fluctuate wildly as a function of large weights. Meanwhile, outcome regression estimates will be sensitive to the specification of the Q model and its regularization, and can incur bias that is difficult to measure if the specification or regularization does not match the true outcome process.

There are a number of practical steps that we can take to sanity-check causal estimates. The simplest check is to compute many different ATE estimators (e.g., outcome regression, IPTW, doubly robust) using several comparably complex estimators of Q and g. We can then check whether they agree, at least qualitatively. If they do agree then this can provide some peace of mind (although it is not a guarantee of accuracy). If they disagree, caution is warranted, particularly in choosing the specification of the Q and g models.

It is also important to check for failures of overlap. Often, issues such as disagreement between alternative estimators can be traced back to poor overlap. A common way to do this, particularly 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 takes 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 τ from a data sample. It does not speak to whether or not τ 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 36.7. 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.

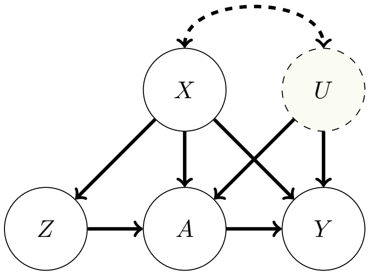


Figure 36.5: Causal graph illustrating the instrumental variable setup. The treatment A and outcome Y are both influenced by unobserved confounder U. Nevertheless, identification is sometimes possible due to the presence of the instrument Z. We also allow for observed covariates X that we may need to adjust for. The dashed arrow between U and X indicates a statistical dependency where we remain agnostic to the particular causal relationship.

36.5 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 strateges based on **instrumental variables**. The instrumental variable graph is shown in Figure 36.5. 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 to 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 to 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 relationships 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 its 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.

36.5.1 Additive unobserved confounding

We first consider **additive unobserved confounding**. That is, we assume that the structural caual model for the outcome has the form:[7]

$$Y \leftarrow f(A, X) + f_U(U). \tag{36.48}$$

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 $\mathbb{E}[Y|X, \mathrm{do}(A = a)] - \mathbb{E}[Y|X, \mathrm{do}(A = a')] = f(a, X) - f(a', X)$. 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*

$$\mathbb{E}[Y|x, \mathrm{do}(A = a)] - \mathbb{E}[Y|x, \mathrm{do}(A = a')] = \tilde{f}(a, x) - \tilde{f}(a', x), \tag{36.49}$$

7. We roll the unit-specific variables ξ into U to avoid notational overload.

for all x, a, a' and such that $\tilde{f}$ satisfies

$$\mathbb{E}[Y|z,x] = \int \tilde{f}(a,x)p(a|z,x)\mathrm{d}a. \tag{36.50}$$

Here, $p(a|z,x)$ is the conditional probability density of treatment.

In particular, if there is a unique function g that satisfies

$$\mathbb{E}[Y|z,x] = \int g(a,x)p(a|z,x)\mathrm{d}a, \tag{36.51}$$

then $g = \tilde{f}$ and this relation identifies the target causal effect.

Before giving the proof, lets understand the point of this identification result. The key insight is that both the left hand side of Equation (36.51) 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 (36.51) 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:

$$\begin{aligned} \mathbb{E}[Y|Z,X] &= \mathbb{E}[f(A,X)|Z,X] + \mathbb{E}[f_U(U)|Z,X] && (36.52)\\ &= \mathbb{E}[f(A,X)|Z,X] + \mathbb{E}[f_U(U)|X] && (36.53)\\ &= \mathbb{E}[\tilde{f}(A,X)|Z,X], && (36.54) \end{aligned}$$

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 identify contrasts between treatements: $f(a,x) - f(a',x) = \tilde{f}(a,x) - \tilde{f}(a',x)$. (The term $\mathbb{E}[f_U(U)|x]$ cancels out). Accordingly, we rewrite Equation (36.54) as:

$$\mathbb{E}[Y|z,x] = \int \tilde{f}(a,x)p(a|z,x)\mathrm{d}a. \tag{36.55}$$

□

It's worth dwelling briefly on how the IV assumptions come into play here. The exclusion restriction is implied by the additive unobserved confounding assumption, which we use explicilty. We also use the unconfoundedness assumption to conclude $U \perp\!\!\!\perp Z|X$. However, we do not use relevance. The role of relevance here is in ensuring that few functions solve the relation Equation (36.51). Informally, the solution g is constrained by the requirement that it hold for all values of Z. However, different values of Z only add non-trivial constraints if $p(a|z,x)$ differ depending on the value of z — this is exactly the relevance condition.

Estimation The basic estimation strategy is to fit models for $\mathbb{E}[Y|z,x]$ and $p(a|z,x)$ from the data, and then solve the implicit equation Equation (36.51) to find g consistent with the fitted models. The procedures for doing this can vary considerably depending on the particulars of the data (e.g., if Z is discrete or continuous) and the choice of modeling strategy. We omit a detailed discussion, but see e.g., [NP03; Dar+11; Har+17; SSG19; BKS19; Mua+20; Dik+20] for various concrete approaches.

It's also worth mentioning an additional nuance to the general procedure. Even if relevance holds, there will often be more than one function that satisfies Equation (36.51). So, we have only identified $\tilde{f}$ as a member of this set of functions. In practice, this ambiguity is defeated by making some additional structural assumption about $\tilde{f}$. For example, we model $\tilde{f}$ with a neural network, and then choose the network satisfying Equation (36.51) that has minimum $l2$-norm on the parameters (i.e., we pick the $l2$-regularized solution).

36.5.2 Instrument monotonicity and local average treatment effect

We now consider an alternative assumption to additive unobserved confounding that is applicable when both the instrument and treatment are binary. It will be convenient to conceptualize the instrument as assignment-to-treatment. Then, the population divides into four subpopulations:

1. Compliers, who take the treatment if assigned to it, and who don't take the treatment otherwise.
2. Always takers, who take the treatment no matter their assignment.
3. Never takers, who refuse the treatment no matter their assignment.
4. Defiers, who refuse the treatment if assigned to it, and who take the treatment if not assigned.

Our goal in this setting will be to identify the average treatment effect among the compliers. The **local average treatment effect** (or **complier average treatment effect**) is defined to be[8]

$$\text{LATE} = \mathbb{E}[Y|\text{do}(A=1), \text{complier}] - \mathbb{E}[Y|\text{do}(A=0), \text{complier}]. \tag{36.56}$$

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, $\text{P}(\text{defier}) = 0$.

We can then write down the identification result.

Theorem 4. *Given the instrumental variable assumptions and instrument monotonicty, the local average treatment is identified as a parameter τ^{LATE} of the observational distributional; that is,* $\text{LATE} = \tau^{\text{LATE}}$. *Namely,*

$$\tau^{\text{LATE}} = \frac{\mathbb{E}[\mathbb{E}[Y|X, Z=1] - \mathbb{E}[Y|X, Z=0]]}{\mathbb{E}[\text{P}(A=1|X, Z=1) - \text{P}(A=1|X, Z=0)]}. \tag{36.57}$$

Proof. We now show that, given the IV assumptions and monotonicity, $\text{LATE} = \tau^{\text{LATE}}$. First, notice that

$$\tau^{\text{LATE}} = \frac{\mathbb{E}[Y|\text{do}(Z=1)] - \mathbb{E}[Y|\text{do}(Z=0)]}{\text{P}(A=1|\text{do}(Z=1)) - \text{P}(A=1|\text{do}(Z=0))}. \tag{36.58}$$

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

8. We follow the econometrics literature in using "LATE" because "CATE" is already commonly used for conditional average treatment effect.

compliers, the units that ignore the instrument (the always/never takers), and the defiers. To that end, note that $\mathrm{P}(\text{complier}|\mathrm{do}(Z=z)) = \mathrm{P}(\text{complier})$ and similarly for always/never takers and defiers — interventions on the instrument don't change the composition of the population. Then,

$$
\begin{aligned}
&\mathbb{E}[Y|\mathrm{do}(Z=1)] - \mathbb{E}[Y|\mathrm{do}(Z=1)] && (36.59)\\
&\quad = \big(\mathbb{E}[Y|\text{complier}, \mathrm{do}(Z=1)] - \mathbb{E}[Y|\text{complier}, \mathrm{do}(Z=0)]\big)\mathrm{P}(\text{complier}) && (36.60)\\
&\quad + \big(\mathbb{E}[Y|\text{always/never}, \mathrm{do}(Z=1)] - \mathbb{E}[Y|\text{always/never}, \mathrm{do}(Z=0)]\big)\mathrm{P}(\text{always/never}) && (36.61)\\
&\quad + \big(\mathbb{E}[Y|\text{defier}, \mathrm{do}(Z=1)] - \mathbb{E}[Y|\text{defier}, \mathrm{do}(Z=0)]\big)\mathrm{P}(\text{defier}). && (36.62)
\end{aligned}
$$

The key is the effect on the complier subpopulation, Equation (36.60). First, by definition of the complier population, we have that:

$$\mathbb{E}[Y|\text{complier}, \mathrm{do}(Z=z)] = \mathbb{E}[Y|\text{complier}, \mathrm{do}(A=z)]. \tag{36.63}$$

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

$$\text{LATE} = \mathbb{E}[Y|\text{complier}, \mathrm{do}(Z=1)] - \mathbb{E}[Y|\text{complier}, \mathrm{do}(Z=0)]. \tag{36.64}$$

Further, we have that $\mathrm{P}(\text{complier}) = \mathrm{P}(A=1|\mathrm{do}(Z=1)) - \mathrm{P}(A=1|\mathrm{do}(Z=0))$. The reason is simply that, by definition of the subpopulations,

$$
\begin{aligned}
&\mathrm{P}(A=1|\mathrm{do}(Z=1)) = \mathrm{P}(\text{complier}) + \mathrm{P}(\text{always taker}) && (36.65)\\
&\mathrm{P}(A=1|\mathrm{do}(Z=0)) = \mathrm{P}(\text{always taker}). && (36.66)
\end{aligned}
$$

Now, plugging the expression for P(complier) and Equation (36.64) into Equation (36.60) we have that:

$$
\begin{aligned}
&\big(\mathbb{E}[Y|\text{complier}, \mathrm{do}(Z=1)] - \mathbb{E}[Y|\text{complier}, \mathrm{do}(Z=0)]\big)\mathrm{P}(\text{complier}) && (36.67)\\
&\quad = \text{LATE} \times (\mathrm{P}(A=1|\mathrm{do}(Z=1)) - \mathrm{P}(A=1|\mathrm{do}(Z=0))) && (36.68)
\end{aligned}
$$

This gives us an expression for the local average treatment effect in terms of the effect of the instrument on the compliers and the probability that a unit takes the treatment when assigned/not-assigned.

The next step is to show that the remaining instrument effect decomposition terms, Equations (36.61) and (36.62), are both 0. Equation (36.61) is the causal effect of the instrument on the always/never takers. It's equal to 0 because, by definition of this subpopulation, the instrument has no causal effect in the subpopulation — they ignore the instrument! Mathematically, this is just $\mathbb{E}[Y|\text{always/never}, \mathrm{do}(Z=1)] = \mathbb{E}[Y|\text{always/never}, \mathrm{do}(Z=0)]$. Finally, Equation (36.62) is 0 by the instrument monotonicity assumption: we assumed that $\mathrm{P}(\text{defier}) = 0$.

In totality, we now have that Equations (36.60) to (36.62) reduces to:

$$
\begin{aligned}
&\mathbb{E}[Y|\mathrm{do}(Z=1)] - \mathbb{E}[Y|\mathrm{do}(Z=1)] && (36.69)\\
&\quad = \text{LATE} \times \big(\mathrm{P}(A=1|\mathrm{do}(Z=1)) - \mathrm{P}(A=1|\mathrm{do}(Z=0))\big) + 0 + 0 && (36.70)
\end{aligned}
$$

Rearranging for LATE and plugging in to Equation (36.58) gives the claimed identification result. □

36.5.2.1 Estimation

For estimating the local average treatment effect under the monotone instrument assumption, there is a double-machine learning approach that works with generic supervised learning approaches. Here, we want an estimator $\hat{\tau}^{\text{LATE}}$ for the parameter

$$\tau^{\text{LATE}} = \frac{\mathbb{E}[\mathbb{E}[Y|X, Z=1] - \mathbb{E}[Y|X, Z=0]]}{\mathbb{E}[\mathrm{P}(A=1|X, Z=1) - \mathrm{P}(A=1|X, Z=0)]}. \tag{36.71}$$

To define the estimator, it's convenient to introduce some additional notation. First, we define the nuisance functions:

$$\mu(z, x) = \mathbb{E}[Y|z, x] \tag{36.72}$$
$$m(z, x) = \mathrm{P}(A=1|x, z) \tag{36.73}$$
$$p(x) = \mathrm{P}(Z=1|x). \tag{36.74}$$

We also define the score ϕ by:

$$\phi_{Z\to Y}(\mathbf{X}; \mu, p) \triangleq \mu(1, X) - \mu(0, X) + \frac{Z(Y - \mu(1, X))}{p(X)} - \frac{(1-Z)(Y - \mu(0, X))}{1 - p(X)} \tag{36.75}$$
$$\phi_{Z\to A}(\mathbf{X}; m, p) \triangleq m(1, X) - m(0, X) + \frac{Z(A - m(1, X))}{p(X)} - \frac{(1-Z)(A - m(0, X))}{1 - p(X)} \tag{36.76}$$
$$\phi(\mathbf{X}; \mu, m, p, \tau) \triangleq \phi_{Z\to Y}(\mathbf{X}; \mu, p) - \phi_{Z\to A}(\mathbf{X}; m, p) \times \tau \tag{36.77}$$

Then, the estimator is defined by a two stage procedure:

1. Fit models $\hat{\mu}, \hat{m}, \hat{p}$ for each of μ, m, p (using supervised machine learning).
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,

$$\hat{\tau}^{\text{LATE}} = \frac{\frac{1}{n}\sum_i \phi_{Z\to Y}(\mathbf{X}_i; \hat{\mu}, \hat{p})}{\frac{1}{n}\sum_i \phi_{Z\to A}(\mathbf{X}_i; \hat{m}, \hat{p})} \tag{36.78}$$

It may help intuitions to notice that the double machine learning estimator of the LATE is effectively the double machine learning estimator of 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 36.4, 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 36.4, 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 36.4.2.4. 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. $\mathbb{E}(\hat{\mu}(Z,X) - \mu(Z,X))^2 \to 0$, $\mathbb{E}(\hat{m}(Z,X) - m(Z,X))^2 \to 0$, and $\mathbb{E}(\hat{p}(X) - p(X))^2 \to 0$
2. $\sqrt{\mathbb{E}[(\hat{p}(X) - p(X))^2]} \times \left(\sqrt{\mathbb{E}[(\hat{\mu}(Z,X) - \mu(Z,X))^2]} + \sqrt{\mathbb{E}[(\hat{m}(Z,X) - m(Z,X))^2]}\right) = o(\sqrt{n})$

then (with some omitted technical conditions) we have asymptotic normality at the $\sqrt{n}$-rate:

$$\sqrt{n}(\hat{\tau}^{\mathrm{LATE-cf}} - \tau^{\mathrm{LATE}}) \xrightarrow{d} \mathrm{Normal}(0, \frac{\mathbb{E}[\phi(\mathbf{X}; \mu, m, p, \tau^{\mathrm{LATE}})^2]}{\mathbb{E}[m(1,X) - m(0,X)]^2}). \tag{36.79}$$

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 μ, 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 (μ) models.

The double machine learning estimator also gives a recipe for quantifying uncertainity. To that end, define

$$\hat{\tau}_{Z\to A} \triangleq \frac{1}{n}\sum_i \phi_{Z\to A}(\mathbf{X}_i; \hat{m}, \hat{p}) \tag{36.80}$$

$$\hat{\mathbb{V}}[\hat{\tau}^{\mathrm{LATE}}] \triangleq \frac{1}{\hat{\tau}^2_{Z\to A}} \frac{1}{n}\sum_i \phi(\mathbf{X}_i; \hat{\mu}, \hat{m}, \hat{p}, \hat{\tau}^{\mathrm{LATE}})^2. \tag{36.81}$$

Then, subject to suitable technical conditions, $\hat{\mathbb{V}}[\hat{\tau}^{\mathrm{LATE-cf}}]$ can be used as an estimate of the variance of the estimator. More precisely,

$$\sqrt{n}(\hat{\tau}^{\mathrm{LATE}} - \tau^{\mathrm{LATE}}) \xrightarrow{d} \mathrm{Normal}(0, \hat{\mathbb{V}}[\hat{\tau}^{\mathrm{LATE}}]). \tag{36.82}$$

Then, confidence intervals or p-values can be computed using this variance in the usual way. The main extra condition required for the variance estimator to be valid is that the nuisance parameters must all converge at rate $O(n^{-1/4})$ (so an excellent estimator for one can't fully compensate for terrible estimators of the others). In fact, even this condition is unnecessary in certain special cases — e.g., when p is known exactly, which occurs when the instrument is randomly assigned. See Chernozhukov et al. [Che+17e] for technical details.

36.5.3 Two stage least squares

Commonly, the IV assumptions are supplemented with the following linear model assumptions:

$$A_i \leftarrow \alpha_0 + \alpha Z_i + \delta_A X_i + \gamma_A X_i + \xi_i^A \tag{36.83}$$

$$Y_i \leftarrow \beta_0 + \beta A_i + \delta_Y X_i + \gamma_Y X_i + \xi_i^Y \tag{36.84}$$

That is, we assume that the real-world process for treatment assignment and the outcome are both linear. In this case, plugging Equation (36.83) into Equation (36.84) yields

$$Y_i \leftarrow \tilde{\beta}_0 + \beta\alpha Z_i + \tilde{\delta} X_i + \tilde{\gamma} X_i + \tilde{\xi}_i. \tag{36.85}$$

The point is that β, the average treatment effect of A on Y, is equal to the coefficient $\beta\alpha$ of the instrument in the outcome-instrument model divided by the coefficient α of the instrument in the treatment-instrument model. So, to estimate the treatment effect, we simply fit both linear models and divide the estimated coefficients. This procedure is called **two stage least squares**.

The simplicity of this procedure is seductive. However, the required linearity assumptions are hard to satisfy in practice and frequently lead to severe issues. A particularly pernicious version of this is that linear-model misspecfication together with weak relevance can yield standard errors for the estimate that are far too small. In practice, this can lead us to find large, significant estimates from two stage least squares when the truth is actually a weak or null effect. See [Rei16; You19; ASS19; Lal+21] for critical evaluations of two stage least squares in practice.

36.6 Difference in differences

Unsurprisingly, time plays an important role in causality. Causes precede effects, and we should be able to incorporate this knowledge into causal identification. We now turn to a particular strategy for causal identification that relies on observing each unit at multiple time points. Data of this kind is sometimes called **panel data**. We'll consider the simplest case. There are two time periods. In the first period, none of the units are treated, and we observe an outcome Y_{0i} for each unit. Then, a subset of the units are treated, denoted by $A_i = 1$. In the second time period, we again observe the outcomes Y_{1i} for each unit, where now the outcomes of the treated units are affected by the treatment. Our goal is to determine the average effect receiving the treatment had on the treated units. That is, we want to know the average difference between the outcomes we actually observed 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**.[9]

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).

9. See github.com/vveitch/causality-tutorials/blob/main/difference_in_differences.ipynb.

The assumption in classical difference-in-differences is the following structural equation:

$$Y_{ti} \leftarrow W_i + S_t + \tau A_i \mathbb{I}(t=1) + \xi_{ti}, \tag{36.86}$$

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 τ 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:

$$\tau = \mathbb{E}[Y_{1i} - Y_{0i}|A=1] - \mathbb{E}[Y_{1i} - Y_{0i}|A=0]. \tag{36.87}$$

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

$$\hat{\tau} = \frac{1}{n_A} \sum_{i:A_i=1} Y_{1i} - Y_{0i} - \frac{1}{n - n_A} \sum_{i:A_i=0} Y_{1i} - Y_{0i}, \tag{36.88}$$

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.[10] 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 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 assumption can be too strong. For instance, administrative data shows employment in Pennsylvania falling relative to employment in New Jersey between 1993 and 1996 [AP08, §5.2]. Although this doesn't directly contradict the parallel trends assumption used for indentification, which needs to hold only in 1992, it does make it seem less credible.

To weaken the assumption, we'll look at a version that requires parallel trends to hold only after adjusting for covariates. To motivate this, we note that there were several different types of fast food restaurant included in the employment data. These vary, e.g., in the type of food they serve, and in cost per meal. Now, it seems reasonable the trend in employment may depend on the type of restuarant. For example, more expensive chains (such as KFC) might be more affected by recessions than cheaper chains (such as McDonald's). If expensive chains are more common in New Jersey than in Pennsylvania, this effect can create a violation of parallel trends — if there's recession affecting both states, we'd expect employment to go down more in New Jersey than in Pennsylvania. However, we may find it credible that McDonald's restaurants in New Jersey have the same trend as McDonald's in Pennsylvania, and similarly for KFC.

10. This is similar to the issue that arises from unobserved confounding, except W_i need not be a cause of the treatment assignment.

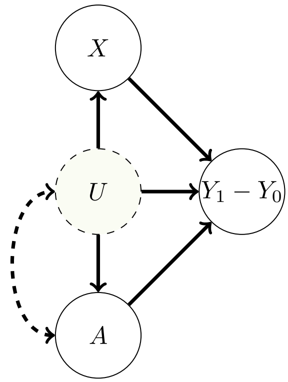


Figure 36.6: Causal graph assumed for the difference-in-differences setting. Here, the outcome of interest is the difference between the pre- and post-treatment period, $Y_1 - Y_0$. This difference is influenced by the treatment, unobserved factors U, and observed covariates X. The dashed arrow between 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.

The next step is to give a definition of the target causal effect that doesn't depend on a parametric model, and a non-parametric statement of the identification assumption to go with it. In words, the causal estimand will be the average treatment effect on the units that received the treatment. To make sense of this mathematically, we'll introduce a new piece of notation:

$$\mathrm{P}^{A=1}(Y|\mathrm{do}(A=a)) \triangleq \int \mathrm{P}(Y|A=a, \text{parents of } Y)\mathrm{dP}(\text{parents of } Y|A=1) \tag{36.89}$$

$$\mathbb{E}^{A=1}[Y|\mathrm{do}(A=a)] \triangleq \mathbb{E}_{\mathrm{P}^{A=1}(Y|\mathrm{do}(A=a))}[Y]. \tag{36.90}$$

In words: recall that the ordinary do operator works by replacing $\mathrm{P}(\text{parents}|A=a)$ by the marginal distribution $\mathrm{P}(\text{parents})$, thereby breaking the backdoor associations. Now, we're replacing the distribution $\mathrm{P}(\text{parents}|A=a)$ by $\mathrm{P}(\text{parents}|A=1)$, irrespective of the actual treatment value. This still breaks all backdoor associations, but is a better match for our target of estimating the treatment effect only among the treated units.

To formalize a causal estimand using the do-calculus, we need to assume some partial causal structure. We'll use the graph in Figure 36.6. With this in hand, our causal estimand is the average treatment effect on the units that received the treatment, namely:

$$\mathrm{ATT}^{\mathrm{DiD}} = \mathbb{E}^{A=1}[Y_1 - Y_0|\mathrm{do}(A=1)] - \mathbb{E}^{A=1}[Y_1 - Y_0|\mathrm{do}(A=0)]] \tag{36.91}$$

In the minimum wage example, this is the average effect of the minimum wage hike on employment in the restaurants affected by it (i.e., the ones in New Jersey).

Finally, we formalize the identification assumption that, conditional on X, the trends in the treated and untreated groups are the same. The **conditional parallel trends** assumption is:

$$\mathbb{E}^{A=1}[Y_1 - Y_0|X, \mathrm{do}(A=0)] = \mathbb{E}[Y_1 - Y_0|X, A=0]. \tag{36.92}$$

In words, this says that for treated units with covariates X, the trend we would have seen had we not assigned treatment is the same as the trend we actually saw for the untreated units with covariates X. That is, if New Jersey had not raised its minimum wage, then McDonald's in New Jersey would have the same expected change in employment as McDonald's in Pennsylvania.

With this in hand, we can give the main identification result:

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 36.6.*
2. *(Conditional parallel trends)* $\mathbb{E}^{A=1}[Y_1 - Y_0 | X, \mathrm{do}(A=0)] = \mathbb{E}[Y_1 - Y_0 | X, A = 0]$.
3. *(Overlap)* $\mathrm{P}(A=1) > 0$ *and* $\mathrm{P}(A=1|X=x) < 1$ *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 $\mathrm{ATT}^{\mathrm{DiD}} = \tau^{\mathrm{DiD}}$, *where*

$$\tau^{\mathrm{DiD}} = \mathbb{E}[\mathbb{E}[Y_1 - Y_0 | A = 1, X] - \mathbb{E}[Y_1 - Y_0 | A = 0, X] | A = 1]. \tag{36.93}$$

Proof. First, by unrolling definitions, we have that

$$\mathbb{E}^{A=1}[Y_1 - Y_0 | \mathrm{do}(A=1), X] = \mathbb{E}[Y_1 - Y_0 | A = 1, X]. \tag{36.94}$$

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,

$$\mathbb{E}^{A=1}[Y_1 - Y_0 | \mathrm{do}(A=0), X] = \mathbb{E}[Y_1 - Y_0 | A = 0, X]. \tag{36.95}$$

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). □

36.6.1 Estimation

With the identification result in hand, the next task is to estimate the observational estimand Equation (36.93). To that end, we define $\tilde{Y} \triangleq Y_1 - Y_0$. Then, we've assumed that $\tilde{Y}, X, A \overset{\text{iid}}{\sim} P$ for some unknown distribution P, and our target estimand is $\mathbb{E}[\mathbb{E}[\tilde{Y} | A = 1, X] - \mathbb{E}[\tilde{Y} | A = 0, X] | A = 1]$. We can immediately recognize this as the observational estimand that occurs in estimating the average treatment effect through adjustment, described in Section 36.4.5.3. That is, even though the causal situation and the identification argument are different between the adjustment setting and the difference in differences setting, the statistical estimation task we end up with is the same. Accordingly, we can use all of the estimation tools we developed for adjustment. That is, all of the techniques there — expected outcome modeling, propensity score methods, double machine learning, and so forth — were purely about the *statistical* task, which is the same between the two scenarios.

So, we're left with the same general recipe for estimation we saw in Section 36.4.6. Namely,

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 predictor for A.

2. Compute the predictions $\hat{Q}(0, x_i), \hat{Q}(1, x_i), \hat{g}(x_i)$ for each datapoint.

3. Combine these predictions into an estimate of the average treatment effect on the treated.

The estimator in the third step can be the expected outcome model estimator, the propensity weighted estimator, the double machine learning estimator, or any other strategy that's valid in the adjustment setting.

36.7 Credibility checks

Once we've chosen an identification strategy, fit our models, and produced an estimate, we're faced with a basic question: should we believe it? Whether the reported estimate succeeds in capturing the true causal effect depends on whether the assumptions required for causal identification hold, the quality of the machine learning models, and the variability in the estimate due to only having access to a finite data sample. The latter two problems are already familiar from machine learning and statistical practice. We should, e.g., assess our models by checking performance on held out data, examining feature importance, and so forth. Similarly, we should report measures of the uncertainity due to finite sample (e.g., in the form of confidence intervals). Because these procedures are already familiar practice, we will not dwell on them further. However, model evaluation and uncertainity quantification are key parts of any credible causal analysis.

Assessing the validity of identification assumptions is trickier. First, there are assumptions that can in fact be checked from data. For example, overlap should be checked in analysis using backdoor adjustment or difference in differences, and relevance should be checked in the instrumental variable setting. Again, checking these conditions is absolutely necessary for a credible causal analysis. But, again, this involves only familiar data analysis, so we will not discuss it further. Next, there are the causal assumptions that cannot be verified from data; e.g., no unobserved confounding in backdoor adjustment, the exclusion restriction in IV, and conditional parallel trends in DiD. Ultimately, the validity of these assumptions must be assessed using substantive causal knowledge of the particular problem under consideration. However, it is possible to conduct some supplementary analyses that make the required judgement easier. We now discuss two such two such techniques.

36.7.1 Placebo checks

In many situations we may be able to find a variable that can be interepreted 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 concientiousness, fear of needles, and so forth. Then, if our identification strategy is valid for the COVID vaccine, we'd also expect it to be 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.

36.7.2 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 36.4. 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 umeasured 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 explained away.

We would like a general procedure to allow domain experts to make judgments about whether plausible confounding is "mild" relative to the "large" effect. In particular, the domain expert must translate judgments about the strength of the unobserved confounding into judgments about the bias induced in the estimate of the effect. Accordingly, we must formalize what is meant by strength of unobserved confounding, and to show how to translate judgments about confounding strength into judgments about bias.

A prototypical example, due to Imbens [Imb03] (building on [RR83]), illustrates the broad approach. As above, the observed data consists of a treatment A, an outcome Y, and covariates X that may causally affect the treatment and outcome. Imbens [Imb03] then posits an additional unobserved binary confounder U for each patient, and supposes that the observed data and unobserved confounder were generated according to the following assumption, known as **Imbens' Sensitivity Model**:

$$U_i \overset{\text{iid}}{\sim} \text{Bern}(1/2) \tag{36.96}$$

$$A_i|X_i, U_i \overset{ind}{\sim} \text{Bern}(\text{sig}(\gamma X_i + \alpha U_i)) \tag{36.97}$$

$$Y_i|X_i, A_i, U_i \overset{ind}{\sim} \mathcal{N}(\tau A_i + \beta X_i + \delta U_i, \sigma^2). \tag{36.98}$$

where sig is the sigmoid function.

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 estimate of the average treatment effect. Since U_i is not observed, it is not possible to identify the parameters from the data. Instead, we make (subjective) judgments about plausible values of α — how strongly U_i affects the treatment assignment — and δ — how strongly U_i affects the outcome. Contingent on plausible $\alpha = \alpha^*$ and $\delta = \delta^*$, the other parameters can be estimated. This yields an estimate of the treatment effect $\hat{\tau}(\alpha^*, \delta^*)$ under the presumed values of the sensitivity parameters.

The approach just outlined has a major drawback: it relies on a parametric model for the full data generating process. The assumed model is equivalent to assuming that, had U been observed, it would have been appropriate to use logistic regression to model treatment assignment, and linear regression to model the outcome. This assumption also implies a simple, parametric model for the relationships governing the observed data. This restriction is out of step with modern practice, where we use flexible machine-learning methods to model these relationships. For example, the assumption forbids the use of neural networks or random forests, though such methods are often state-of-the-art for causal effect estimation.

Austen plots We now turn to developing an alternative an adaptation of Imbens' approach that fully decouples sensitivity analysis and modeling of the observed data. Namely, the **Austen plots** of [VZ20]. An example Austen plot is shown in Figure 36.7. The high-level idea is to posit a generative model that uses a simple, interpretable parametric form for the influence of the unobserved confounder, but that *puts no constraints on the model for the observed data.* We then use the parametric part of the model to formalize "confounding strength" and to compute the induced bias as a function of the confounding.

Austen plots further adapt two strategies pioneered by Imbens [Imb03]. First, we find a parameterization of the model so that the sensitivity parameters, measuring strength of confounding, are on a standardized, unitless scale. This allows us to compare the strength of hypothetical unobserved confounding to the strength of observed covariates, measured from data. Second, we plot the curve 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 36.7, 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.

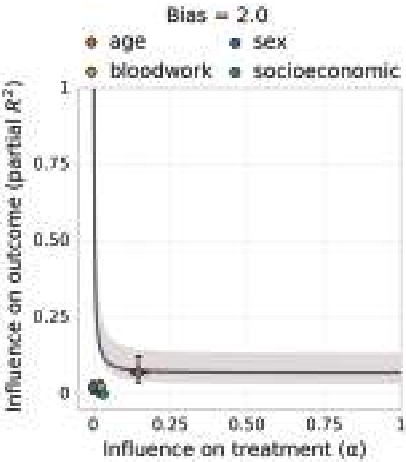


Figure 36.7: Austen plot showing how strong an unobserved confounder would need to be to induce a bias of 2 in an observational study of the effect of combination blood pressure medications on diastolic blood pressure [Dor+16]. We chose this bias to equal the nominal average treatment effect estimated 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.

Setup The data are generated independently and identically $(Y_i, A_i, X_i, U_i) \overset{\text{iid}}{\sim} P$, where U_i is not observed and P is some unknown probability distribution. The approach in Section 36.4 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, τ, of the observed data distribution, where

$$\tau = \mathbb{E}[\mathbb{E}[Y|X, A = 1] - \mathbb{E}[Y|X, A = 0]]. \tag{36.99}$$

This observational parameter is then estimated from a finite data sample. Recall from Section 36.4 that this involves estimating the conditional expected outcome $Q(A, X) = \mathbb{E}[Y|A, X]$ and the propensity score $g(X) = \mathrm{P}(A = 1|X)$, then plugging these into an estimator $\hat{\tau}$.

We are now concerned with the case of possible unobserved confounding. That is, where U causally affects Y and A. If there is unobserved confounding then the parameter τ is not equal to the ATE, so $\hat{\tau}$ is a biased estimate. Inference about the ATE then divides into two tasks. First, the statistical task: estimating τ as accurately as possible from the observed data. And, second, the causal (domain-specific) problem of assessing $\texttt{bias} = \texttt{ATE} - \tau$. We emphasize that our focus here is bias due to causal misidentification, not the statistical bias of the estimator. Our aim is to reason about the bias induced by unobserved confounding — the second task — in a way that imposes no constraints on the modeling choices for $\hat{Q}$, $\hat{g}$, and $\hat{\tau}$ used in the statistical analysis.

Sensitivity model Our sensitivity analysis should impose no constraints on how the *observed* data is modeled. However, sensitivity analysis demands some assumption on the relationship between the observed data and the *unobserved* confounder. It is convenient to formalize such assumptions by specifying a probabilistic model for how the data is generated. The strength of confounding is then formalized in terms of the parameters of the model (the sensitivity parameters). Then, the bias induced by the confounding can be derived from the assumed model. Our task is to posit a generative model that both yields a useful and easily interpretable sensitivity analysis, and that avoids imposing any assumptions about the observed data.

To begin, consider the functional form of the sensitivity model used by Imbens [Imb03].

$$\text{logit}(\text{P}(A=1|x,u)) = h(x) + \alpha u \tag{36.100}$$

$$\mathbb{E}[Y|a,x,u] = l(a,x) + \delta u, \tag{36.101}$$

for some functions h and l. That is, the propensity score is logit-linear in the unobserved confounder, and the conditional expected outcome is linear.

By rearranging Equation (36.100) to solve for u and plugging in to Equation (36.101), we see that it's equivalent to assume $\mathbb{E}[Y|t,x,u] = \tilde{l}(t,x) + \tilde{\delta}\text{logit}\text{P}(A=1|x,u)$. That is, the unobserved confounder u only influences the outcome through the propensity score. Accordingly, by positing a distribution on $\text{P}(A=1|x,u)$ directly, we can circumvent the need to explicitly articulate U (and h).

Definition 36.7.1. *Let $\tilde{g}(x,u) = \text{P}(A=1|x,u)$ denote the propensity score given observed covariates x and the unobserved confounder u.*

The insight is that we can posit a sensitivity model by defining a distribution on $\tilde{g}$ directly. We choose:

$$\tilde{g}(X,U)|X \sim \text{Beta}(g(X)(1/\alpha - 1), (1-g(X))(1/\alpha - 1)).$$

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 α plays the same role as in Imbens' model: it controls the influence of the unobserved confounder U on treatment assignment. When α 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 α 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 α 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):

$$\alpha = \mathbb{E}[\tilde{g}(X,U)|A=1] - \mathbb{E}[\tilde{g}(X,U)|A=0]. \tag{36.102}$$

With the $\tilde{g}$ model in hand, we define the **Austen sensitivity model** as follows:

$$\tilde{g}(X,U)|X \sim \text{Beta}(g(X)(1/\alpha - 1), (1-g(X))(1/\alpha - 1)) \tag{36.103}$$

$$A|X,U \sim \text{Bern}(\tilde{g}(X,U)) \tag{36.104}$$

$$\mathbb{E}[Y|A,X,U] = Q(A,X) + \delta\big(\text{logit}\tilde{g}(X,U) - \mathbb{E}[\text{logit}\tilde{g}(X,U)|A,X]\big). \tag{36.105}$$

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:

$$\begin{aligned} \mathrm{P}(A=1|X) &= \mathbb{E}[\mathbb{E}[T|X,U]|X] = \mathbb{E}[\tilde{g}(X,U)|X] = g(X) \\ \mathbb{E}[Y|A,X] &= \mathbb{E}[\mathbb{E}[Y|A,X,U]|A,X] = Q(A,X). \end{aligned}$$

The sensitivity parameters are α, controlling the dependence between the unobserved confounder the treatment assignment, and δ, 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:

$$\texttt{ATE} = \mathbb{E}[\mathbb{E}[Y|A=1,X,U] - \mathbb{E}[Y|A=0,X,U]].$$

Plugging in our sensitivity model yields,

$$\texttt{ATE} = \mathbb{E}[Q(1,X) - Q(0,X)] + \delta(\mathbb{E}[\mathrm{logit}\tilde{g}(X,U)|X,A=1] - \mathbb{E}[\mathrm{logit}\tilde{g}(X,U)|X,A=0]).$$

The first term is the observed-data estimate τ, so

$$\texttt{bias} = \delta(\mathbb{E}[\mathrm{logit}\tilde{g}(X,U)|X,A=1] - \mathbb{E}[\mathrm{logit}\tilde{g}(X,U)|X,A=0]).$$

Then, by invoking beta-Bernoulli conjugacy and standard beta identities,[11] we arrive at,

Theorem 6. *Under the Austen sensitivity model, Equation (36.105), an unobserved confounder with influence α and δ induces bias in the estimated treatment effect equal to*

$$\texttt{bias} = \frac{\delta}{1/\alpha - 1}\mathbb{E}\Big[\frac{1}{g(X)} + \frac{1}{1-g(X)}\Big].$$

That is, the amount of bias is determined by the sensitivity parameters and by the *realized* propensity score. Notice that more extreme propensity scores lead to more extreme bias in response to unobserved confounding. This means, in particular, that conditioning on a covariate that affects the treatment but that does not directly affect the outcome (an instrument) will increase any bias due to unobserved confounding. This general phenomena is known as **z-bias**.

Sensitivity parameters The Austen model provides a formalization of confounding strength in terms of the parameters α and δ and tells us how much bias is induced by a given strength of confounding. This lets us translate judgments about confounding strength to judgments about bias. However, it is not immediately obvious how to translate qualitative judgements such as "I think any unobserved confounder would be much less important than age" to judgements about the possible values of the sensitivity parameters.

First, because the scale of δ is not fixed, it may be difficult to compare the influence of potential unobserved confounders to the influence of reference variables. To resolve this, we reexpress the outcome-confounder strength in terms of the (non-parametric) partial coefficient of determination:

$$R^2_{Y,\mathrm{par}}(\alpha,\delta) = 1 - \frac{\mathbb{E}(Y - \mathbb{E}[Y|A,X,U])^2}{\mathbb{E}(Y - Q(A,X))^2}.$$

The key to computing the reparameterization is the following result

11. We also use the recurrence relation $\psi(x+1) - \psi(x) = 1/x$, where ψ is the digamma function.

Theorem 7. *Under the Austen sensitivity model, Equation (36.105), the outcome influence is*

$$R^2_{Y,\text{par}}(\alpha,\delta) = \delta^2 \sum_{a=0}^{1} \frac{\mathbb{E}\left[\psi_1\left(g(X)^a(1-g(X))^{1-a}(1/\alpha - 1) + 1[A=a]\right)\right]}{\mathbb{E}[(Y-Q(A,X))^2]},$$

where ψ_1 is the trigamma function.

See Veitch and Zaveri [VZ20] for the proof.

By design, α — the strength of confounding influence on on treatment assignment — is already on a fixed, unitless scale. However, because the measure is tied to the model it may be difficult to interpret, and it is not obvious how to compute reference confounding strength values from the observed data. The next result clarifies these issues.

Theorem 8. *Under the Austen sensitivity model, Equation (36.105),*

$$\alpha = 1 - \frac{\mathbb{E}[\tilde{g}(X,U)(1-\tilde{g}(X,U))]}{\mathbb{E}[g(X)(1-g(X))]}.$$

See Veitch and Zaveri [VZ20] for the proof. That is, the sensitivity parameter α is measures how much more extreme the propensity scores become when we condition on U. That is, α is a measure of the extra predictive power U adds for A, above and beyond the predictive power in X. It may also be insightful to notice that

$$\alpha = R^2_{A,\text{par}} = 1 - \frac{\mathbb{E}[(A-\tilde{g}(X,U))^2]}{\mathbb{E}[(A-g(X))^2]}. \tag{36.106}$$

That is, α is just the (non-parametric) partial coefficient of determination of U on A—the same measure used for the outcome influence. (To see this, just expand the expectations conditional on $A=1$ and $A=0$).

Estimating bias In combination, Theorems 6 and 7 yield an expression for the bias in terms of α and $R^2_{Y,\text{par}}$. 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.

36.7.2.1 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^2_Y and α. We want to measure the degree of influence of an observed covariate Z given the other observed covariates $X \backslash Z$.

For the outcome, this can be measured as:

$$R^2_{Y\cdot Z|T,X\backslash Z} \triangleq 1 - \frac{\mathbb{E}(Y-Q(A,X))^2}{\mathbb{E}(Y-\mathbb{E}[Y|A,X\backslash Z])^2}.$$

In practice, we can estimate the quantity by fitting a new regression model $\hat{Q}_Z$ that predicts Y from A and $X\backslash Z$. Then we compute

$$R^2_{Y\cdot Z|T,X\backslash Z} = 1 - \frac{\frac{1}{n}\sum_i (y_i - \hat{Q}(t_i, x_i))^2}{\frac{1}{n}\sum_i (y_i - \hat{Q}_Z(t_i, x_i\backslash z_i))^2}.$$

Using Theorem 8, we can measure influence of observed covariate Z on treatment assignment given $X\backslash Z$ in an analogous fashion to the outcome. We define $g_{X\backslash Z}(X\backslash Z) = \mathrm{P}(A = 1|X\backslash Z)$, then fit a model for $g_{X\backslash Z}$ by predicting A from $X\backslash Z$, and estimate

$$\hat{\alpha}_{Z|X\backslash Z} = 1 - \frac{\frac{1}{n}\sum_i \hat{g}(x_i)(1 - \hat{g}(x_i))}{\frac{1}{n}\sum_i \hat{g}_{X\backslash Z}(x_i\backslash z_i)(1 - \hat{g}_{X\backslash Z}(x_i\backslash z_i))}.$$

Grouping covariates The estimated values $\hat{\alpha}_{X\backslash Z}$ and $\hat{R}^2_{Y,X\backslash Z}$ 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'.

36.7.2.2 Practical use

We now have sufficient results to produce Austen plots such as Figure 36.7. 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).

3. Plot the values of α and $R^2_{Y,\mathrm{par}}$ that would suffice to induce that much bias. This is the black curve on the plot. To calculate these values, use Theorems 6 and 7 together with the estimated $\hat{g}$ and $\hat{Q}$.

4. Finally, compute reference influence level for (groups of) observed covariates. In particular, this requires fitting reduced models for the conditional expected outcome and propensity that do not use the reference covariate as a feature.

In practice, an analyst only needs to do the model fitting parts themselves. The bias calculations, reference value calculations, and plotting can be done automatically with standard libraries.[12].

Austen plots are predicated on Equation (36.105). This assumption replaces the purely parametric Equation (36.98) with a version that eliminates any parametric requirements on the observed data. However, we emphasize that Equation (36.105) does, implicitly, impose some parametric assumption on the structural causal relationship between U and A, Y. Ultimately, any conclusion drawn from

12. See github.com/vveitch/causality-tutorials/blob/main/Sensitivity_Analysis.ipynb.

the sensitivity analysis depends on this assumption, which is not justified on any substantive grounds. Accordingly, such sensitivity analyses can only be used to informally guide domain experts. They do not circumvent the need to thoroughly adjust for confounding. This reliance on a structural assumption is a generic property of sensitivity analysis.[13] Indeed, there are now many sensitivity analysis models that allow the use of any machine learning model in the data analysis [e.g., RRS00; FDF19; She+11; HS13; BK19; Ros10; Yad+18; ZSB19; Sch+21a]. However, none of these are yet in routine use in practice. We have presented Austen plots here not because they make an especially virtuous modeling assumption, but because they are (relatively) easy to understand and interpret.

Austen plots are most useful in situations where the conclusion from the plot would be 'obvious' to a domain expert. For instance, in Figure 36.7, we can be confident that an unobserved confounder similar to socioeconomic status would not induce enough bias to change the qualitative conclusion. By contrast, Austen plots should not be used to draw conclusions such as, "I think a latent confounder could only be 90% as strong as 'age', so there is evidence of a small non-zero effect". Such nuanced conclusions might depend on issues such as the particular sensitivity model we use, or finite-sample variation of our bias and influence estimates, or on incautious interpretation of the calibration dots. These issues are subtle, and it would be difficult resolve them to a sufficient degree that a sensitivity analysis would make an analysis credible.

Calibration using observed data The interpretation of the observed-data calibration requires some care. The sensitivity analysis requires the analyst to make judgements about the strength of influence of the unobserved confounder U, *conditional on the observed covariates* X. However, we report the strength of influence of observed covariate(s) Z, *conditional on the other observed covariates* $X \backslash Z$. The difference in conditioning sets can have subtle effects.

Cinelli and Hazlett [CH20] give an example where Z and U are identical variables in the true model, but where influence of U given A, X is larger than the influence of Z given $A, X \backslash Z$. (The influence of Z given $X \backslash Z, U$ would be the same as the influence of U given X). Accordingly, an analyst is *not* justified in a judgment such as, "I know that U and Z are very similar. I see Z has substantial influence, but the dot is below the line. Thus, U will not undo the study conclusions". In 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 artifically towards the top right part of the plot).

13. In extreme cases, there can be so little unexplained variation in A or Y that only a very weak confounder could be compatible with the data. In this case, essentially assumption free sensitivity analysis is possible [Man90].

36.8 The do-calculus

We have seen several strategies for identifying causal effects as parameters of observational distributions. Confounder adjustment (Section 36.4) 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. $\mathrm{P}(Y|\mathrm{do}(A=a))$, with statements involving only observational quantities, e.g. $\mathbb{E}_X[\mathrm{P}(Y|A=a, X)]$. 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.

36.8.1 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:

$$p(y|\mathrm{do}(x), z, w) = p(y|\mathrm{do}(x), w) \text{ if } (Y \perp Z|X, W)_{G_{\overline{X}}} \tag{36.107}$$

where $G_{\overline{X}}$ denotes cuting 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 conditioniong on irrelevant variables leaves the distribution invariant (as we would expect).

Rule 2 The second rule allows us to replace $\mathrm{do}(z)$ with conditioning on (seeing) z. The simplest case where can do this is: if Z is a root of the causal graph (i.e., it has no causal parents) then $p(y|\mathrm{do}(z)) = p(y|z)$. The reason is that the do operator is equivalent to conditioning in the mutilated causal graph where all the edges into Z are removed, but, because Z is a root, the mutilated graph is just the original causal graph. The general form of this rule is:

$$p(y|\mathrm{do}(x), \mathrm{do}(z), w) = p(y|\mathrm{do}(x), z, w) \text{ if } (Y \perp Z|X, W)_{G_{\overline{X}\underline{Z}}} \tag{36.108}$$

where $G_{\overline{X}\underline{Z}}$ cuts edges going into X and out of Z. Intutively, we can replace $\mathrm{do}(z)$ by z as long as there are no backdoor (non-directed) paths between z and y. If there are in fact no such paths, then cutting all the edges going out of Z will mean there are no paths connecting Z and Y, so that $Y \perp\!\!\!\perp Z$. The rule just generalizes this line of reasoning to allow for extra observed and intervened variables.

Rule 3 The third rule allows us to insert or delete actions $\mathrm{do}(z)$:

$$p(y|\mathrm{do}(x), \mathrm{do}(z), w) = p(y|\mathrm{do}(x), w) \text{ if } (Y \perp Z|X, W)_{G_{\overline{X}\overline{Z^*}}} \tag{36.109}$$

where $G_{\overline{X}\overline{Z^*}}$ cuts edges going into X and Z^*, and where Z^* is the set of Z-nodes that are not ancestors of any W-node in $G_{\overline{X}}$. Intuitively, this condition corresponds to intervening on X, and checking whether the distribution of Y is invariant to *any* intervention that we could apply on Z.

36.8.2 Revisiting backdoor adjustment

We begin with a more general form of the adjustment formula we used in Section 36.4.

First, suppose we observe all of A's parents, call them X. For notational simplicity, we'll assume for the moment that X is discrete. Then,

$$p(Y = y|\text{do}(A = a)) = \sum_x p(Y = y|x, \text{do}(A = a))p(x|\text{do}(A = a)) \tag{36.110}$$

$$= \sum_x p(Y = y|x, A = a)p(x). \tag{36.111}$$

The first line is just a standard probability relation (marginalizing over x). We are using causal assumptions in two ways in the second line. First, $p(x|\text{do}(A = a)) = p(x)$: the treatment has no causal effect on X, so interventions on A don't change the distribution of X. This is rule 3, Equation (36.109). Second, $p(Y = y|x, \text{do}(A = a)) = p(Y = y|x, A = a)$. This equality holds because conditioning on the parents blocks all non-directed paths from A to Y, reducing the causal effect to be the same as the observational effect. The equality is an application of rule 2, Equation (36.108).

Now, what if we don't observe all the parents of A? The key issue is **backdoor paths**: paths between A and Y that contain an arrow into A. These paths are the general form of the problem that occurs when A and Y share a common cause. Suppose that we can find a set of variables S such that (1) no node in S is a descendant of A; and (2) S blocks every backdoor path between A and Y. Such a set is said to satisfy the **backdoor criterion**. In this case, we can use S instead X in the adjustment formula, Equation (36.111). That is,

$$p(Y = y|\text{do}(A = a)) = \mathbb{E}_S[p(Y = y|S, A = a)]. \tag{36.112}$$

The proof follows the invocation of rules 3 and 2, in the same way as for the case where S is just the parents of A. Notice that requiring S to not contain any descendants of A means that we don't risk 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 36.4. That's because both the parents of A and the common causes satisfy the backdoor criterion.

In practice, the full distribution $p(Y = y|\text{do}(A = a))$ 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 τ can then be estimated from finite data using the methods described in Section 36.4, using S in place of the common causes X.

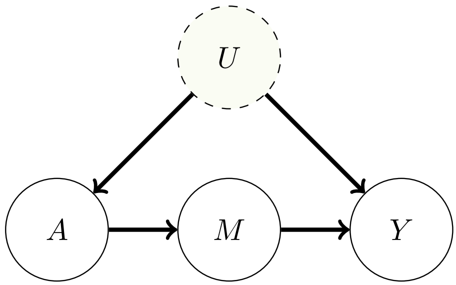


Figure 36.8: Causal graph illustrating the frontdoor criterion setup. The effect of the treatment A on outcome Y is entirely mediated by mediator M. This allows us infer the causal effect even if the treatment and outcome are confounded by U.

36.8.3 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 36.8 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 can combine these as:

$$p(Y|\text{do}(A=a)) = \sum_m p(Y|\text{do}(M=m))p(M=m|\text{do}(A=a)) \tag{36.113}$$

$$= \sum_m \sum_{a'} p(Y|a',m)p(a')p(m|a) \tag{36.114}$$

The second line is just backdoor adjustment applied to identify each of the do expressions (note that A blocks the M-Y backdoor path through U).

Equation (36.114) is called the **front-door formula** [Pea09b, §3.3.2]. To state the result in more general terms, let us introduce a definition. We say a set of variables M satisfies the **front-door criterion** relative to an ordered pair of variables (A, Y) if (1) M intercepts all directed paths from A to Y; (2) there is no unblocked backdoor path from A to M; and (3) all backdoor paths from M to Y are blocked by A. If M satisfies this criterion, and if $p(A, M) > 0$ for all values of A and M, then the causal effect of A on Y is identifiable and is given by Equation (36.114).

Let us interpret this theorem in terms of our smoking example. Condition 1 means that smoking A should have no effect on cancer Y except via tar and tissue damage M. Conditions 2 and 3 mean that the genotype U cannot have any effect on M except via smoking A. Finally, the requirement that $p(A, M) > 0$ for all values implies that high levels of tar in the lungs must arise not only due to smoking, but also other factors (e.g., pollutants). In other words, we require $p(A = 0, M = 1) > 0$ so we can assess the impact of the mediator in the untreated setting.

We can now use the do-calculus to derive the frontdoor criterion; following [PM18b, p236]. Assuming the causal graph G shown in Figure 36.8:

$$\begin{aligned}
p(y|\mathrm{do}(a)) &= \sum_m p(y|\mathrm{do}(a), m)p(m|\mathrm{do}(a)) && \text{(probability axioms)} \\
&= \sum_m p(y|\mathrm{do}(a), \mathrm{do}(m))p(m|\mathrm{do}(a)) && \text{(rule 2 using } G_{\overline{S}\underline{T}}) \\
&= \sum_m p(y|\mathrm{do}(a), \mathrm{do}(m))p(m|a) && \text{(rule 2 using } G_{\underline{S}}) \\
&= \sum_m p(y|\mathrm{do}(m))p(m|a) && \text{(rule 3 using } G_{\overline{S}\,\overline{T^*}}) \\
&= \sum_{a'} \sum_m p(y|\mathrm{do}(m), a')p(a'|\mathrm{do}(m))p(m|a) && \text{(probability axioms)} \\
&= \sum_{a'} \sum_m p(y|m, a')p(a'|\mathrm{do}(m))p(m|a) && \text{(rule 2 using } G_{\underline{T}}) \\
&= \sum_{a'} \sum_m p(y|m, a')p(a')p(m|a) && \text{(rule 3 using } G_{\overline{T^*}})
\end{aligned}$$

Estimation To estimate the causal distribution from data using the frontdoor criterion we need to 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 Y from M and A, and $\hat{p}(m|a)$ by predicting M from A. We can estimate $p(a)$ by the empirical distribution of A. Then,

$$\sum_{a'} \sum_m \hat{p}(a')\hat{p}(y|m, a')\hat{p}(m|a), \tag{36.115}$$

We usually have more modest targets than the full distribution $p(y|\mathrm{do}(a))$. For instance, we may be content with just estimating the average treatment effect. It's straightforward to derive a formula for this using the frontdoor adjustment. Similarly to backdoor adjustment, more advanced estimators of the ATE through frontdoor effect are possible in principle. For example, we might combine fitted models for $\mathbb{E}[Y|m, a]$ and $\mathrm{P}(M|a)$. See Fulcher et al. [Ful+20] for an approach to robust estimation via front door adjustment, as well as a generalization of the front door approach to more general settings.

36.9 Further reading

There is an enormous and growing literature on the intersection of causality and machine learning.

First, there are many textbooks on theoretical and practical elements of causal inference. These include Pearl [Pea09c], focused on causal graphs, Angrist and Pischke [AP08], focused on econometrics, Hernán and Robins [HR20b], with roots in epidemiology, Imbens and Rubin [IR15], with origin in statistics, and Morgan and Winship [MW15], for a social sciences perspective. The introduction to causality in Shalizi [Sha22, §7] is also recommended, particularly the treatment of matching.

Double machine-learning has featured prominently in this chapter. This is a particular instantiation of non-parametric estimation. This topic has substantial theoretical and practical importance in modern causal inference. The double machine learning work includes estimators for many commonly encountered scenarios [Che+17e; Che+17d]. Good references for a lucid explanation of how and why non-parametric estimation works include [Ken16; Ken17; FK21]. Usually, the key guarantees of non-parametric estimator are asymptotic. Generally, there are many estimators that share optimal asymptotic guarantees (e.g., the AIPTW estimator given in Equation (36.30)). Although these are asymptotically equivalent, in finite samples their behavior can be very different. There are estimators that preserve asymptotic guarantees but aim to improve performance in practical finite sample regimes [e.g., vR11].

There is also considerable interest in the estimation of heterogeneous treatment effects. The question here is: what effect would this treatment have when applied to a unit with such-and-such specific characteristics? E.g., what is the effect of this drug on women over the age of 50? The causal identification arguments used here are more-or-less the same as for the estimation of average case effects. However, the estimation problems can be substantially more involved. Some reading includes [Kün+19; NW20; Ken20; Yad+21].

There are several commonly applicable causal identification and estimation strategies beyond the ones we've covered in this chapter. **Regression discontinuity designs** rely on the presence of some sharp, arbitrary non-linearity in treatment assignment. For example, eligibility for some aid programs is determined by whether an individual has income below or above a fixed amount. The effect of the treatment can be studied by comparing units just below and just above this threshhold. **Synthetic controls** are a class of methods that try to study the effect of a treatment on a given unit by constructing a synthetic version of that unit that acts as a control. For example, to study the effect of legislation banning smoking indoors in California, we can construct a synthetic California as a weighted average of other states, with weights chosen to balance demographic characteristics. Then, we can compare the observed outcome of California with the outcome of the synthetic control, constructed as the weighted average of the outcomes of the donor states. See Angrist and Pischke [AP08] for a textbook treatment of both strategies. Closely related are methods that use time series modeling to create synthetic outcomes. For example, to study the effect of an advertising campaign beginning at time T on product sales Y_t, we might build a time series model for Y_t using data in the $t < T$ period, and then use this model to predict the values of $(\hat{Y}_t)_{t>T}$ we would have seen had the campaign not been run. We can estimate the causal effect by comparing the factual, realized Y_t to the predicted, counterfactual, $\hat{Y}_t$. See Brodersen et al. [Bro+15] for an instantiation of this idea.

In this chapter, our focus has been on using machine learning tools to estimate causal effects. There is also a growing interest in using the ideas of causality to improve machine learning tools. This is mainly aimed at building predictors that are robust when deployed in new domains [SS18b; SCS19; Arj+20; Mei18b; PBM16a; RC+18; Zha+13a; Sch+12b; Vei+21] or that do not rely on particular 'spurious' correlations in the training data [RPH21; Wu+21; Gar+19; Mit+20; WZ19; KCC20; KHL20; TAH20; Vei+21].

Index

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: (2021). arXiv: 2107.07511 [cs.LG].

[Aba] 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.

[Abn+21] S. Abnar, M. Dehghani, B. Neyshabur, and H. Sedghi. "Exploring the limits of large scale pre-training". In: *ICLR*. 2021.

[ABV21] S. Akbayrak, I. Bocharov, and B. de Vries. "Extended Variational Message Passing for Automated Approximate Bayesian Inference". In: *Entropy* 23.7 (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 (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 (2016), e1005110.

[ACL89] L. Atlas, D. Cohn, and R. Ladner. "Training connectionist networks with queries and selective sampling". In: *Advances in neural information processing systems* 2 (1989).

[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. Muelly, I. Goodfellow, M. Hardt, and B. Kim. *Sanity Checks for Saliency Maps*. 2020. arXiv: 1810.03292 [cs.CV].

[Ade+20b] J. Adebayo, M. Muelly, I. Liccardi, and B. Kim. "Debugging tests for model explanations". In: *arXiv preprint arXiv:2011.05429* (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.

[Ado] *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.

[ADT12] C. Andrieu, A. Doucet, and V. B. Tadić. "One-line Parameter Estimation in General State-Space Models using a Pseudo-Likelihood Approach". In: *IFAC Proceedings Volumes* 45.16 (July 2012), pp. 500–505.

[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. 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. Frosst, X. Zhang, B. Lengerich, R. Caruana, and G. Hinton. *Neural Additive Models: Interpretable Machine Learning with Neural Nets*. 2021. arXiv: 2004.13912 [cs.LG].

[Aga+22] A. Agarwal, N. Jiang, S. Kakade, and W. Sun. *Reinforcement Learning:Theory and Algorithms*. 2022.

[AGM14] P. Agrawal, R. Girshick, and J. Malik. "Analyzing the performance of multilayer neural networks for object recognition". In: *European conference on computer vision*. Springer. 2014, pp. 329–344.

[AGM19] V. Amrhein, S. Greenland, and B. McShane. "Scientists rise up against statistical significance". In: *Nature* 567.7748 (2019), p. 305.

[AH09] I. Arasaratnam and S. Haykin. "Cubature Kalman Filters". In: *IEEE Trans. Automat. Contr.* 54.6 (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 (2007), pp. 953–977.

[AHG20] L. Ambrogioni, M. Hinne, and M. van Gerven. "Automatic structured variational inference". In: (2020). arXiv: 2002.00643 [stat.ML].

[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. 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 (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.

[AHS85] 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. Tsochantaridis. "Support Vector Machine Learning for Interdependent and Structured Output Spaces". In: *Predicting Structured Data*. Ed. by G. Bakir, T. Hofmann, B. Scholkopf, 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.

[AI 19] AI Artists. *Creative Tools to Generate AI Art*. 2019.

[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: (2018). arXiv: 1808.07172 [cs.LG].

[AKZK19] B. Amos, V. Koltun, and J Zico Kolter. "The Limited Multi-Label Projection Layer". In: (2019). arXiv: 1906.08707 [cs.LG].

[AL+16] J. Ala-Luhtala, N. Whiteley, K. Heine, and R. Piche. "An Introduction to Twisted Particle Filters and Parameter Estimation in Non-linear State-space Models". In: *IEEE Trans. Signal Process.* 64.18 (2016), pp. 4875–4890.

[al21] M. A. et al. *Understanding Dataset Shift and Potential Remedies*. Tech. rep. Vector Institute, 2021.

[Ale+16] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy. "Deep Variational Information Bottleneck". In: *ICLR*. 2016.

[Ale+17] A. A. Alemi, I. S. Fischer, J. V. Dillon, and K. P. Murphy. "Deep Variational Information Bottleneck". In: *ArXiv* abs/1612.00410 (2017).

[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.

[Alq21] P. Alquier. "User-friendly introduction to PAC-Bayes bounds". In: (2021). arXiv: 2110.11216 [stat.ML].

[Alq22] P. Alquier. "Approximate Bayesian Inference". In: *Entropy* 22.11 (2022).

[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 Astronomical Society* 488.3 (2019), pp. 4440–4458.

[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.

[AM07] R. P. Adams and D. J. C. MacKay. "Bayesian Online Changepoint Detection". In: (2007). arXiv: 0710.3742 [stat.ML].

[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". en. In: *Sci. Rep.* 6 (2016), p. 26677.

[AM74] D. Andrews and C. Mallows. "Scale mixtures of Normal distributions". In: *J. of Royal Stat. Soc. Series B* 36 (1974), pp. 99–102.

[AM89] B. D. Anderson and J. B. Moore. *Optimal Control: Linear Quadratic Methods*. Prentice-Hall International, Inc., 1989.

[Ama09] S.-I. Amari. "α-Divergence Is Unique, Belonging to Both f-Divergence and Bregman Divergence Classes". In: *IEEE Trans. Inf. Theory* 55.11 (2009), pp. 4925–4931.

[Ama98] S Amari. "Natural Gradient Works Efficiently in Learning". In: *Neural Comput.* 10.2 (1998), pp. 251–276.

[Ame+19] S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, S. Iqbal, P. N. Bennett, K. Inkpen, et al. "Guidelines for human-AI interaction". In: *Proceedings of the 2019 chi conference on human factors in computing systems*. 2019, pp. 1–13.

[Ami01] E. Amir. "Efficient Approximation for Triangulation of Minimum Treewidth". In: *UAI*. 2001.

[AMJ18a] D. Alvarez-Melis and T. S. Jaakkola. "On the robustness of interpretability methods". In: *arXiv preprint arXiv:1806.08049* (2018).

[AMJ18b] D. Alvarez-Melis and T. S. Jaakkola. "Towards robust interpretability with self-explaining neural networks". In: *arXiv preprint arXiv:1806.07538* (2018).

[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.

[Amo+18] B. Amos, L. Dinh, S. Cabi, T. Rothörl, S. G. Colmenarejo, A. Muldal, T. Erez, Y. Tassa, N. de Freitas, and M. Denil. "Learning Awareness Models". In: *ICLR*. 2018.

[Amo22] B. Amos. "Tutorial on amortized optimization for learning to optimize over continuous domains". In: (2022). arXiv: 2202.00665 [cs.LG].

[AMO88] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. "Network flows". In: (1988).

[Ana+14] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. "Tensor Decompositions for Learning Latent Variable Models". In: *JMLR* 15 (2014), pp. 2773–2832.

[Ana+23] A. Anastasiou et al. "Stein's Method Meets Computational Statistics: A Review of Some Recent Developments". en. In: *SSO Schweiz. Monatsschr. Zahnheilkd.* 38.1 (Feb. 2023), pp. 120–139.

[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. 5–43.

[And+20] O. M. Andrychowicz et al. "Learning dexterous in-hand manipulation". In: *Int. J. Rob. Res.* 39.1 (2020), pp. 3–20.

[And82] B. D. O. Anderson. "Reverse-time diffusion equation models". In: *Stochastic Processes and thei Applications* 12.3 (May 1982), pp. 313–326.

[Ang+18] E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer, and C. Rudin. *Learning Certifiably Optimal Rule Lists for Categorical Data*. 2018. arXiv: 1704.01701 [stat.ML].

[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.

[Ang+21] A. N. Angelopoulos, S. Bates, E. J. Candès, M. I. Jordan, and L. Lei. "Learn then Test: Calibrating Predictive Algorithms to Achieve Risk Control". In: (2021). arXiv: 2110.01052 [cs.LG].

[Ang88] D. Angluin. "Queries and concept learning". In: *Machine learning* 2.4 (1988), pp. 319–342.

[Ani+18] R. Anirudh, J. J. Thiagarajan, B. Kailkhura, and T. Bremer. "An Unsupervised Approach to Solving Inverse Problems using Generative Adversarial Networks". In: (2018). arXiv: 1805.07281 [cs.CV].

[Ano19] Anonymous. "Neural Tangents: Fast and Easy Infinite Neural Networks in Python". In: (2019).

[Ant+22] I. Antonoglou, J. Schrittwieser, S. Ozair, T. K. Hubert, and D. Silver. "Planning in Stochastic Environments with a Learned Model". In: *ICLR*. 2022.

[AO03] J.-H. Ahn and J.-H. Oh. "A Constrained EM Algorithm for Principal Component Analysis". In: *Neural Computation* 15 (2003), pp. 57–65.

[AOM17] M. G. Azar, I. Osband, and R. Munos. "Minimax Regret Bounds for Reinforcement Learning". In: *ICML*. 2017, pp. 263–272.

[AP08] J. D. Angrist and J.-S. Pischke. *Mostly harmless econometrics: An empiricist's companion*. Princeton university press, 2008.

[AP09] J. Angrist and J.-S. Pischke. *Mostly Harmless Econometrics*. 2009.

[AP19] M. Abadi and G. D. Plotkin. "A simple differentiable programming language". In: *Proceedings of the ACM on Programming Languages* 4.POPL (2019), pp. 1–28.

[Ara+09] A. Aravkin, B. Bell, J. Burke, and G. Pillonetto. *An L1-Laplace Robust Kalman Smoother*. Tech. rep. U. Washington, 2009.

[Ara10] A. Aravkin. *Student's t Kalman Smoother*. Tech. rep. U. Washington, 2010.

[Ara+17] A. Aravkin, J. V. Burke, L. Ljung, A. Lozano, and G. Pillonetto. "Generalized Kalman smoothing: Modeling and algorithms". In: *Automatica* 86 (2017), pp. 63–86.

[Arb+18] M. Arbel, D. Sutherland, M. Bińkowski, and A. Gretton. "On gradient regularizers for MMD GANs". In: *Advances in neural information processing systems*. 2018, pp. 6700–6710.

[Arj+19] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. "Invariant Risk Minimization". In: (2019). arXiv: 1907.02893 [stat.ML].

[Arj+20] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. *Invariant Risk Minimization*. 2020. arXiv: 1907.02893 [stat.ML].

[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.

[Aro+13] S. Arora, R. Ge, Y. Halpern, D. Mimno, A. Moitra, D. Sontag, Y. Wu, and M. Zhu. "A Practical Algorithm for Topic Modeling with Provable Guarantees". In: *ICML*. 2013.

[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: (2019). arXiv: 1904.11955 [cs.LG].

[Aro+21] R. Arora et al. *Theory of deep learning*. 2021.

[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 (2013), pp. 709–729.

[Aru+02] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking". In: *IEEE Trans. on Signal Processing* 50.2 (2002), pp. 174–189.

[Aru+17] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. "A Brief Survey of Deep Reinforcement Learning". In: *IEEE Signal Processing Magazine, Special Issue on Deep Learning for Image Understanding* (2017).

[AS17] A. Achille and S. Soatto. "On the Emergence of Invariance and Disentangling in Deep Representations". In: (2017). arXiv: 1706.01350 [cs.LG].

[AS18] A. Achille and S. Soatto. "On the Emergence of Invariance and Disentangling in Deep Representations". In: *JMLR* 18 (2018), pp. 1–34.

[AS66] S. M. Ali and S. D. Silvey. "A General Class of Coefficients of Divergence of One Distribution from Another". In: *J. R. Stat. Soc. Series B Stat. Methodol.* 28.1 (1966), pp. 131–142.

[Asa00] C. Asavathiratham. "The Influence Model: A Tractable Representation for the Dynamics of Networked Markov Chains". PhD thesis. MIT, Dept. EECS, 2000.

[ASD20] A. Agrawal, D. Sheldon, and J. Domke. "Advances in Black-Box VI: Normalizing Flows, Importance Weighting, and Optimization". In: *NIPS*. 2020.

[ASM17] A. Azuma, M. Shimbo, and Y. Matsumoto. "An Algebraic Formalization of Forward and Forward-backward Algorithms". In: (2017). arXiv: 1702.06941 [cs.LG].

[ASN20] R. Agarwal, D. Schuurmans, and M. Norouzi. "An Optimistic Perspective on Offline Reinforcement Learning". In: *ICML*. 2020.

[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).

[AT08] C. Andrieu and J. Thoms. "A tutorial on adaptive MCMC". In: *Statistical Computing* 18 (2008), pp. 343–373.

[AT20] E. Agustsson and L. Theis. "Universally Quantized Neural Compression". 2020.

[Att00] H. Attias. "A Variational Bayesian Framework for Graphical Models". In: *NIPS-12*. 2000.

[Att03] H. Attias. "Planning by Probabilistic Inference". In: *AI-Stats*. 2003.

[Aue12] J. E. Auerbach. "Automated evolution of interesting images". In: *Artificial Life 13*. 2012.

[Aus+21] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. van den Berg. "Structured Denoising Diffusion Models in Discrete State-Spaces". In: *NIPS*. July 2021.

[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).

[AXK17] B. Amos, L. Xu, and J. Z. Kolter. "Input convex neural networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 146–155.

[AY19] B. Amos and D. Yarats. "The Differentiable Cross-Entropy Method". In: (2019). arXiv: 1909.12830 [cs.LG].

[AZ17] S. Arora and Y. Zhang. "Do gans actually learn the distribution? an empirical study". In: *arXiv preprint arXiv:1706.08224* (2017).

[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.* (2020), pp. 000–000.

[Azi+15] H. Azizpour, A. Sharif Razavian, J. Sullivan, A. Maki, and S. Carlsson. "From generic to specific deep representations for visual recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2015, pp. 36–45.

[BA03] D. Barber and F. Agakov. "The IM Algorithm: A Variational Approach to Information Maximization". In: *NIPS*. NIPS'03. MIT Press, 2003, pp. 201–208.

[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.

[Bac09] F. Bach. "High-Dimensional Non-Linear Variable Selection through Hierarchical Kernel Learning". In: (2009). arXiv: 0909.0844 [cs.LG].

[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.

[Bac+15a] 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.

[Bac+15b] S. H. Bach, M. Broecheler, B. Huang, and L. Getoor. "Hinge-Loss Markov Random Fields and Probabilistic Soft Logic". In: (2015). arXiv: 1505.04406 [cs.LG].

[Bac+18] E. Bach, J. Dusart, L. Hellerstein, and D. Kletenik. "Submodular goal value of Boolean functions". In: *Discrete Applied Mathematics* 238 (2018), pp. 1–13.

[Bad+18] M. A. Badgeley, 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.

[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.* (2020).

[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.

[Bai95] L. C. Baird. "Residual Algorithms: Reinforcement Learning with Function Approximation". In: *ICML*. 1995, pp. 30–37.

[Bak+17] J. Baker, P. Fearnhead, E. B. Fox, and C. Nemeth. "Control Variates for Stochastic Gradient MCMC". In: (2017). arXiv: 1706.05439 [stat.CO].

[Bal17] S. Baluja. "Learning deep models of optimization landscapes". In: *IEEE Symposium Series on Computational Intelligence (SSCI)* (2017).

[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.

[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.

[Ban06] A. Banerjee. "On bayesian bounds". In: *ICML*. 2006, pp. 81–88.

[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.

[Bao+22a] F. Bao, S. Nie, K. Xue, Y. Cao, C. Li, H. Su, and J. Zhu. "All are Worth Words: A ViT Backbone for Diffusion Models”. In: (Sept. 2022). arXiv: 2209.12152 [cs.CV].

[Bao+22b] H. Bao, L. Dong, S. Piao, and F. Wei. “BEiT: BERT Pre-Training of Image Transformers”. In: *International Conference on Learning Representations*. 2022.

[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.

[Bat+12] D. Batra, P. Yadollahpour, A. Guzman-Rivera, and G. Shakhnarovich. “Diverse M-Best Solutions in Markov Random Fields”. In: *ECCV*. Springer Berlin Heidelberg, 2012, pp. 1–16.

[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.

[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).

[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).

[Bau+70] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. “A maximization technique occuring in the statistical analysis of probabalistic functions in Markov chains”. In: *The Annals of Mathematical Statistics* 41 (1970), pp. 164–171.

[Bau74] B. G. Baumgart. “Geometric modeling for computer vision.” In: 1974.

[Bax00] J Baxter. “A Model of Inductive Bias Learning”. In: *JAIR* (2000).

[Bay+15] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. “Automatic differentiation in machine learning: a survey”. In: (2015). arXiv: 1502.05767 [cs.SC].

[BB08] O. Bousquet and L. Bottou. “The Tradeoffs of Large Scale Learning”. In: *NIPS*. 2008, pp. 161–168.

[BB11] L. Bottou and O. Bousquet. “The Tradeoffs of Large Scale Learning”. In: *Optimization for Machine Learning*. Ed. by S. Sra, S. Nowozin, and S. J. Wright. MIT Press, 2011, pp. 351–368.

[BB12] J. Bergstra and Y. Bengio. “Random Search for Hyper-Parameter Optimization”. In: *JMLR* 13 (2012), pp. 281–305.

[BB15a] A. Bendale and T. Boult. “Towards Open World Recognition”. In: *CVPR*. 2015.

[BB15b] J. Bornschein and Y. Bengio. “Reweighted Wake-Sleep”. In: *ICLR*. 2015.

[BB17] J. Bilmes and W. Bai. “Deep Submodular Functions”. In: *Arxiv* abs/1701.08939 (2017).

[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.

[BBM10] N. Bhatnagar, C. Bogdanov, and E. Mossel. *The Computational Complexity of Estimating Convergence Time*. Tech. rep. arxiv, 2010.

[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.

[BBS95] A. G. Barto, S. J. Bradtke, and S. P. Singh. “Learning to act using real-time dynamic programming”. In: *AIJ* 72.1 (1995), pp. 81–138.

[BBV11a] R. Benassi, J. Bect, and E. Vazquez. “Bayesian optimization using sequential Monte Carlo”. In: (2011). arXiv: 1111.4802 [math.OC].

[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 (LION)*. 2011, pp. 176–190.

[BC07] D. Barber and S. Chiappa. “Unified inference for variational Bayesian linear Gaussian state space models”. In: *NIPS*. 2007.

[BC08] M. Bădoiu and K. L. Clarkson. “Optimal coresets for balls”. In: *Computational Geometry* 40.1 (2008), pp. 14–22.

[BC14] J. Ba and R. Caruana. “Do Deep Nets Really Need to be Deep?” In: *Advances in Neural Information Processing Systems* 27 (2014).

[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.

[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.

[BC93] B. M. Bell and F. W. Cathey. “The iterated Kalman filter update as a Gauss-Newton method”. In: *IEEE Trans. Automat. Contr.* 38.2 (Feb. 1993), pp. 294–297.

[BC94] P. Baldi and Y. Chauvin. “Smooth online learning algorithms for Hidden Markov Models”. In: *Neural Computation* 6 (1994), pp. 305–316.

[BC95] S. Baluja and R. Caruana. “Removing the Genetics from the Standard Genetic Algorithm”. In: *ICML*. 1995, pp. 38–46.

[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: (2010). arXiv: 1012.2599 [cs.LG].

[BCH20] M. Briers, M. Charalambides, and C. Holmes. “Risk scoring calculation for the current NHSx contact tracing app”. In: (2020). arXiv: 2005.11057 [cs.CY].

[BCJ20] A. Buchholz, N. Chopin, and P. E. Jacob. “Adaptive Tuning Of Hamiltonian Monte Carlo Within Sequential Monte Carlo”. In: *Bayesian Anal.* (2020).

[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.

[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.

[BCV13] Y. Bengio, A. Courville, and P. Vincent. “Representation learning: A review and new perspectives”.

In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[BD+10] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. "A theory of learning from different domains". In: *Mach. Learn.* 79.1 (May 2010), pp. 151–175.

[BD11] A. Bhattacharya and D. B. Dunson. "Simplex factor models for multivariate unordered categorical data". In: *JASA* (2011).

[BD87] G. Box and N. Draper. *Empirical Model-Building and Response Surfaces*. Wiley, 1987.

[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.

[BD97] S. Baluja and S. Davies. "Using Optimal Dependency-Trees for Combinatorial Optimization: Learning the Structure of the Search Space". In: *ICML*. 1997.

[BDM09] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems*. SIAM, 2009.

[BDM10] M. Briers, A. Doucet, and S. Maskel. "Smoothing algorithms for state-space models". In: *Annals of the Institute of Statistical Mathematics* 62.1 (2010), pp. 61–89.

[BDM17] M. G. Bellemare, W. Dabney, and R. Munos. "A Distributional Perspective on Reinforcement Learning". In: *ICML*. 2017.

[BDM18] N. Brosse, A. Durmus, and E. Moulines. "The promises and pitfalls of Stochastic Gradient Langevin Dynamics". In: *NIPS*. 2018.

[BDS18] A. Brock, J. Donahue, and K. Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: (2018). arXiv: 1809.11096 [cs.LG].

[Bea03] M. Beal. "Variational Algorithms for Approximate Bayesian Inference". PhD thesis. Gatsby Unit, 2003.

[Bea19] M. A. Beaumont. "Approximate Bayesian Computation". In: *Annual Review of Statistics and Its Application* 6.1 (2019), pp. 379–403.

[Béd08] M. Bédard. "Optimal acceptance rates for Metropolis algorithms: Moving beyond 0.234". In: *Stochastic Process. Appl.* 118.12 (2008), pp. 2198–2222.

[Beh+19] J. Behrmann, W. Grathwohl, R. T. Q. Chen, D. Duvenaud, and J.-H. Jacobsen. "Invertible Residual Networks". In: *ICML*. 2019.

[Bel03] A. J. Bell. "The co-information lattice". In: *ICA conference*. 2003.

[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.

[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. PMLR, 2018, pp. 531–540.

[Bel+19] D. Belanger, S. Vora, Z. Mariet, R. Deshpande, D. Dohan, C. Angermueller, K. Murphy, O. Chapelle, and L. Colwell. "Biological Sequence Design using Batched Bayesian Optimization". In: *NIPS workshop on ML for the sciences*. 2019.

[Bel94] B. M. Bell. "The Iterated Kalman Smoother as a Gauss–Newton Method". In: *SIAM J. Optim.* 4.3 (Aug. 1994), pp. 626–636.

[Ben13] Y. Bengio. "Estimating or Propagating Gradients Through Stochastic Neurons". In: (2013). arXiv: 1305.2982 [cs.LG].

[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.

[Ben+21a] Y. Bengio, T. Deleu, E. J. Hu, S. Lahlou, M. Tiwari, and E. Bengio. "GFlowNet Foundations". In: (Nov. 2021). arXiv: 2111.09266 [cs.LG].

[Ben+21b] 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.

[Ben+22] K. Benidis et al. "Deep Learning for Time Series Forecasting: Tutorial and Literature Survey". In: *ACM Computing Surveys* (2022).

[Ber05] J. M. Bernardo. "Reference Analysis". In: *Handbook of Statistics*. Ed. by D. K. Dey and C. R. Rao. Vol. 25. Elsevier, 2005, pp. 17–90.

[Ber15] D. Bertsekas. *Convex Optimization Algorithms*. Athena Scientific, 2015.

[Ber16] D. Bertsekas. *Nonlinear Programming*. Third. Athena Scientific, 2016.

[Ber+18] R. van den Berg, L. Hasenclever, J. M. Tomczak, and M. Welling. "Sylvester normalizing flows for variational inference". In: *AISTATS*. 2018.

[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.

[Ber19] D. Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, 2019.

[Ber+21] J. Berner, P. Grohs, G. Kutyniok, and P. Petersen. "The Modern Mathematics of Deep Learning". In: (2021). arXiv: 2105.04026 [cs.LG].

[Ber85a] 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.

[Ber85b] J. Berger. *Statistical Decision Theory and Bayesian Analysis (2nd edition)*. Springer-Verlag, 1985.

[Ber97a] D. A. Berry. "Teaching Elementary Bayesian Statistics with Real Applications in Science". In: *Am. Stat.* 51.3 (1997), pp. 241–246.

[Ber97b] D. Bertsekas. *Parallel and Distribution Computation: Numerical Methods*. Athena Scientific, 1997.

[Ber99] A. Berchtold. "The double chain Markov model". In: *Comm. Stat. Theor. Methods* 28 (1999), pp. 2569–2589.

[Bes75] J. Besag. "Statistical analysis of non-lattice data". In: *The Statistician* 24 (1975), pp. 179–196.

[Bet13] M. Betancourt. "A General Metric for Riemannian Manifold Hamiltonian Monte Carlo". In: *Geometric Science of Information*. Springer Berlin Heidelberg, 2013, pp. 327–334.

[Bet17] M. Betancourt. "A Conceptual Introduction to Hamiltonian Monte Carlo". In: (2017). arXiv: 1701.02434 [stat.ME].

[Bet18] M. Betancourt. *Probability Theory (For Scientists and Engineers)*. 2018.

[Bey+20] L. Beyer, O. J. Hénaff, A. Kolesnikov, X. Zhai, and A. van den Oord. "Are we done with ImageNet?" In: (2020). arXiv: 2006.07159 [cs.CV].

[Béz+20] E. de Bézenac, S. S. Rangapuram, K. Benidis, M. Bohlke-Schneider, R. Kurle, L. Stella, H. Hasson, P. Gallinari, and T. Januschowski. "Normalizing Kalman Filters for Multivariate Time Series Analysis". In: *NIPS*. Vol. 33. 2020, pp. 2995–3007.

[BFH75] Y. Bishop, S. Fienberg, and P. Holland. *Discrete Multivariate Analysis: Theory and Practice*. MIT Press, 1975.

[BFY20] T. D. Barfoot, J. R. Forbes, and D. Yoon. "Exactly Sparse Gaussian Variational Inference with Application to Derivative-Free Batch Nonlinear State Estimation". In: *Intl. J. of Robotics Research* (2020).

[BG06] M. Beal and Z. Ghahramani. "Variational Bayesian Learning of Directed Graphical Models with Hidden Variables". In: *Bayesian Analysis* 1.4 (2006).

[BG13] M. J. Betancourt and M. Girolami. "Hamiltonian Monte Carlo for Hierarchical Models". In: (2013). arXiv: 1312.0906 [stat.ME].

[BG73] A. Björck and G. H. Golub. "Numerical methods for computing angles between linear subspaces". In: *Mathematics of computation* 27.123 (1973), pp. 579–594.

[BG96] A. Becker and D. Geiger. "A sufficiently fast algorithm for finding close to optimal junction trees". In: *UAI*. 1996.

[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.

[BGM17] J Ba, R Grosse, and J Martens. "Distributed Second-Order Optimization using Kronecker-Factored Approximations". In: *ICLR*. openreview.net, 2017.

[BGS16] Y. Burda, R. Grosse, and R. Salakhutdinov. "Importance Weighted Autoencoders". In: *ICLR*. 2016.

[BGT93] C. Berrou, A. Glavieux, and P. Thitimajashima. "Near Shannon limit error-correcting coding and decoding: Turbo codes". In: *Proc. IEEE Intl. Comm. Conf.* (1993).

[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.

[BH20] M. T. Bahadori and D. Heckerman. "Debiasing Concept-based Explanations with Causal Analysis". In: *International Conference on Learning Representations*. 2020.

[BH92] D. Barry and J. A. Hartigan. "Product partition models for change point problems". In: *Annals of statistics* 20 (1992), pp. 260–279.

[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.

[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.

[Bha+23] A. Bhatnagar, H. Wang, C. Xiong, and Y. Bai. "Improved Online Conformal Prediction via Strongly Adaptive Online Learning". In: (Feb. 2023). arXiv: 2302.07869 [cs.LG].

[BHB19] P. Bachman, R. D. Hjelm, and W. Buchwalter. *Learning Representations by Maximizing Mutual Information Across Views*. 2019. arXiv: 1906.00910 [cs.LG].

[BHB22] F. Berns, J. Hüwel, and C. Beecks. "Automated Model Inference for Gaussian Processes: An Overview of State-of-the-Art Methods and Algorithms". en. In: *SN Comput Sci* 3.4 (May 2022), p. 300.

[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.

[BHO75] P. J. Bickel, E. A. Hammel, and J. W. O'connell. "Sex bias in graduate admissions: data from berkeley". en. In: *Science* 187.4175 (1975), pp. 398–404.

[BHPI02] M. Bādoiu, S. Har-Peled, and P. Indyk. "Approximate clustering via core-sets". In: *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 2002, pp. 250–257.

[BHW16] P. G. Bissiri, C. Holmes, and S. Walker. "A General Framework for Updating Belief Distributions". In: *JRSSB* 78.5 (2016), 1103–1130.

[Bic09] D. Bickson. "Gaussian Belief Propagation: Theory and Application". PhD thesis. Hebrew University of Jerusalem, 2009.

[Bie06] G. J. Bierman. *Factorization Methods for Discrete Sequential Estimation (Dover Books on Mathematics)*. en. Illustrated edition. Dover Publications, 2006.

[Big+11] B. Biggio, G. Fumera, 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.

[Bil01] J. A. Bilmes. *Graphical Models and Automatic Speech Recognition*. Tech. rep. UWEETR-2001-0005. Univ. Washington, Dept. of Elec. Eng., 2001.

[Bil22] J. Bilmes. "Submodularity In Machine Learning and Artificial Intelligence". In: (2022). arXiv: 2202.00132 [cs.LG].

[Biń+18] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton. "Demystifying MMD GANs". In: *ICLR*. 2018.

[Bin+18] M. Binkowski, D. J. Sutherland, M. Arbel, and A. Gretton. "Demystifying MMD GANs". In: *International Conference on Learning Representations*. 2018.

[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.

[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.

[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.

[Bis06] C. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[Bis99] C. Bishop. "Bayesian PCA". In: *NIPS*. 1999.

[Bit16] S. Bitzer. *The UKF exposed: How it works, when it works and when it's better to sample*. 2016.

[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.

[BJ05] F. Bach and M. Jordan. *A probabilistic interpretation of canonical correlation analysis*. Tech. rep. 688. U. C. Berkeley, 2005.

[BJ06] W. Buntine and A. Jakulin. "Discrete Component Analysis". In: *Subspace, Latent Structure and Feature Selection: Statistical and Optimization Perspectives Workshop*. 2006.

[BJV97] J. S. D. Bonet, C. L. I. Jr., and P. A. Viola. "MIMIC: Finding Optima by Estimating Probability Densities". In: *NIPS*. MIT Press, 1997, pp. 424–430.

[BK10] R. Bardenet and B. Kegl. "Surrogating the surrogate: accelerating Gaussian-process-based global optimization with a mixture cross-entropy algorithm". In: *ICML*. 2010.

[BK15] D. Belanger and S. Kakade. "A Linear Dynamical System Model for Text". en. In: *ICML*. 2015, pp. 833–842.

[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].

[BKB17] O. Bastani, C. Kim, and H. Bastani. "Interpreting blackbox models via model extraction". In: *arXiv preprint arXiv:1705.08504* (2017).

[BKH16] J. L. Ba, J. R. Kiros, and G. E. Hinton. "Layer Normalization". In: (2016). arXiv: 1607.06450 [stat.ML].

[BKM16] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. "Variational Inference: A Review for Statisticians". In: *JASA* (2016).

[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.

[BL06] D. Blei and J. Lafferty. "Dynamic topic models". In: *ICML*. 2006, pp. 113–120.

[BL07a] C. M. Bishop and J. Lasserre. "Generative or discriminative? Getting the best of both worlds". In: *Bayesian Statistics 8*. 2007.

[BL07b] D. Blei and J. Lafferty. "A Correlated Topic Model of "Science"". In: *Annals of Applied Stat.* 1.1 (2007), pp. 17–35.

[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.

[Ble12] D. M. Blei. "Probabilistic topic models". In: *Commun. ACM* 55.4 (2012), pp. 77–84.

[BLH21] Y. Bengio, Y. Lecun, and G. Hinton. "Deep learning for AI". In: *Comm. of the ACM* 64.7 (June 2021), pp. 58–65.

[BLH22] J. Bornschein, Y. Li, and M. Hutter. "Sequential Learning Of Neural Networks for Prequential MDL". In: (Oct. 2022). arXiv: 2210.07931 [stat.ML].

[BLM16] S. Boucheron, G. Lugosi, and P. Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2016.

[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.

[BLS11] G. Blanchard, G. Lee, and C. Scott. "Generalizing from several related classification tasks to a new unlabeled sample". In: *NIPS*. 2011.

[BLS17] J. Ballé, V. Laparra, and E. P. Simoncelli. "End-to-end Optimized Image Compression". In: *ICLR*. 2017.

[Blu+15] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. "Weight Uncertainty in Neural Networks". In: *ICML*. 2015.

[BM+18] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, T. B. Dhruva, A. Muldal, N. Heess, and T. Lillicrap. "Distributed Distributional Deterministic Policy Gradients". In: *ICLR*. 2018.

[BM19] Y. Blau and T. Michaeli. "Rethinking Lossy Compression: The Rate-Distortion-Perception Tradeoff". In: *ICML*. 2019.

[BM93] H. A. Bourlard and N. Morgan. *Connectionist Speech Recognition: A Hybrid Approach*. USA: Kluwer Academic Publishers, 1993.

[BMK20] Z. Borsos, M. Mutny, and A. Krause. "Coresets via Bilevel Optimization for Continual Learning and Streaming". In: *Advances in Neural Information Processing Systems* 33 (2020).

[BMM00] J. Barnard, R. Mcculloch, and X.-L. Meng. "Modeling covariance matrices in terms of standard deviations and correlations, with application to shrinkage". In: *Stat. Sin.* 10 (2000).

[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 (2019).

[BMR97] J. Binder, K. Murphy, and S. Russell. "Space-efficient inference in dynamic probabilistic networks". In: *IJCAI*. 1997.

[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.

[BNJ03a] D. Blei, A. Ng, and M. Jordan. "Latent Dirichlet allocation". In: *JMLR* 3 (2003), pp. 993–1022.

[BNJ03b] D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent dirichlet allocation". In: *JMLR* 3 (2003), pp. 993–1022.

[BO14] J. Bayer and C. Osendorfer. "Learning Stochastic Recurrent Networks". In: *Workshop on Advances in Variational Inference*. 2014.

[Boe+05] P.-T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. "A Tutorial on the Cross-Entropy

Method". en. In: *Ann. Oper. Res.* 134.1 (2005), pp. 19–67.

[Boh92] D. Bohning. "Multinomial logistic regression algorithm". In: *Annals of the Inst. of Statistical Math.* 44 (1992), pp. 197–200.

[Bol02] W. Bolstad. "Teaching Bayesian Statistics to Undergraduates: Who, What, Where, When, Why and How". In: *ICOTS6 Intl. Conf. on Teaching Statistics*. 2002.

[Bol89] K. Bollen. *Structural Equation Models with Latent Variables*. John Wiley & Sons, 1989.

[Bon64] G. Bonnet. "Transformations des signaux aleátoires a travers les systemes non lineáires sans mem̀oire". In: *Annales des Telecommuncations* 19 (1964).

[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 (2017), pp. 1694–1705.

[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 Systems: 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.

[Box80] G. E. P. Box. "Sampling and Bayes' Inference in Scientific Modelling and Robustness". In: *J. of Royal Stat. Soc. Series A* 143.4 (1980), pp. 383–430.

[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. 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 (2016), pp. 7345–7352.

[BP21] T. Bricken and C. Pehlevan. "Attention Approximates Sparse Distributed Memory". In: *NIPS*. 2021.

[BPK16] S. Bulo, L. Porzi, and P. Kontschieder. "Distillation dropout". In: *ICML*. 2016.

[BPL21a] R. Balestriero, J. Pesenti, and Y. LeCun. "Learning in High Dimension Always Amounts to Extrapolation". In: (Oct. 2021). arXiv: 2110.09485 [cs.LG].

[BPL21b] A. Bardes, J. Ponce, and Y. LeCun. "Vicreg: Variance-invariance-covariance regularization for self-supervised learning". In: *arXiv preprint arXiv:2105.04906* (2021).

[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 a 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. Pavez, and K. Cranmer. "Mining gold from implicit models to improve likelihood-free inference". en. In: *Proc. Natl. Acad. Sci. U. S. A.* 117.10 (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 (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 (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: (2020). arXiv: 2010.06451 [stat.ME].

[Bri12] W. M. Briggs. "It is Time to Stop Teaching Frequentism to Non-statisticians". In: *arXiv* (2012). arXiv: 1201.2590 [stat.OT].

[Bri50] G. W. Brier. "Verification of forecasts expressed in terms of probability". In: *Monthly Weather Review* 78.1 (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, 2019.

[Bro+20a] D. Brookes, A. Busia, C. Fannjiang, K. Murphy, and J. Listgarten. "A view of estimation of distribution algorithms through the lens of expectation-maximization". In: *GECCO*. GECCO '20. Association for Computing Machinery, 2020, pp. 189–190.

[Bro+20b] P. Brouillard, S. Lachapelle, A. Lacoste, S. Lacoste-Julien, and A. Drouin. "Differentiable Causal Discovery from Interventional Data". In: *NIPS*. July 2020.

[Bro+20c] D. Brown, R. Coleman, R. Srinivasan, and S. Niekum. "Safe imitation learning via fast bayesian reward inference from preferences". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 1165–1177.

[Bro+20d] T. B. Brown et al. "Language Models are Few-Shot Learners". In: (2020). arXiv: 2005.14165 [cs.CL].

[BRS17] E. Balkanski, A. 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: (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.

[BS+20] S. Bobadilla-Suarez, C. Ahlheim, A. Mehrotra, A. Panos, and B. C. Love. "Measures of neural similarity". In: *Computational Brain & Behavior* 3.4 (2020), pp. 369–383.

[BS94] J. Bernardo and A. Smith. *Bayesian Theory*. John Wiley, 1994.

[BS95a] A. J. Bell and T. J. Sejnowski. "An information maximisation approach to blind separation and blind deconvolution". In: *Neural Computation* 7.6 (1995), pp. 1129–1159.

[BS95b] 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 (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, 2011.

[BT00] C. Bishop and M. Tipping. "Variational relevance vector machines". In: *UAI*. 2000.

[BT04] G. Bouchard and B. Triggs. "The tradeoff between generative and discriminative classifiers". In: *IASC International Symposium on Computational Statistics (COMPSTAT '04)*. 2004.

[BT08] D. Bertsekas and J. Tsitsiklis. *Introduction to Probability*. 2nd Edition. Athena Scientific, 2008.

[BT12] M. Botvinick and M. Toussaint. "Planning as inference". en. In: *Trends Cogn. Sci.* 16.10 (2012), pp. 485–488.

[BT73] G. Box and G. Tiao. *Bayesian inference in statistical analysis*. Addison-Wesley, 1973.

[BTEGN09] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *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 (2017), pp. 55–79.

[Buc21] J. Buchner. "Nested Sampling Methods". In: (Jan. 2021). arXiv: 2101.09675 [stat.CO].

[Bud+21] K. Budhathoki, D. Janzing, P. Bloebaum, and H. Ng. "Why did the distribution change?" In: *AISTATS*. Ed. by A. Banerjee and K. Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, 2021, pp. 1666–1674.

[Bul11] A. D. Bull. "Convergence rates of efficient global optimization algorithms". In: *JMLR* 12 (2011), 2879–2904.

[Bul+20] S. Bulusu, B. Kailkhura, 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.

[BW20] G. J. J. van den Burg and C. K. I. Williams. "An Evaluation of Change Point Detection Algorithms". In: (2020). arXiv: 2003.06222 [stat.ML].

[BW21] G. J. van den Burg and C. K. Williams. "On Memorization in Probabilistic Deep Generative Models". In: *NIPS*. 2021.

[BWM18] A. Buchholz, F. Wenzel, and S. Mandt. "Quasi-Monte Carlo Variational Inference". In: *ICML*. 2018.

[BWR16] M. Bauer, M. van der Wilk, and C. E. Rasmussen. "Understanding Probabilistic Sparse Gaussian Process Approximations". In: *NIPS*. 2016, pp. 1533–1541.

[BYH20] O. Bohdal, Y. Yang, and T. Hospedales. "Flexible Dataset Distillation: Learn Labels Instead of Images". In: *arXiv preprint arXiv:2006.08572* (2020).

[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.

[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 (2016), pp. 1008–1031.

[BZ20] A. Barbu and S.-C. Zhu. *Monte Carlo Methods*. en. Springer, 2020.

[CA13] E. F. Camacho and C. B. Alba. *Model predictive control*. Springer, 2013.

[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.

[CAII20] V. Coscrato, M. H. de Almeida Inácio, and R. Izbicki. "The NN-Stacking: Feature weighted linear stacking through neural networks". In: *Neurocomputing* (2020).

[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.

[Cal20] O. Calin. *Deep Learning Architectures: A Mathematical Approach*. en. 1st ed. Springer, 2020.

[Cam+21] A. Campbell, Y. Shi, T. Rainforth, and A. Doucet. "Online Variational Filtering and Parameter Learning". In: *NIPS*. 2021.

[Can04] J. Canny. "GaP: a factor model for discrete data". In: *SIGIR*. 2004, pp. 122–129.

[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 (2015), pp. 2415–2427.

[Cao+22] H. Cao, C. Tan, Z. Gao, G. Chen, P.-A. Heng, and S. Z. Li. "A Survey on Generative Diffusion Model". In: (Sept. 2022). arXiv: 2209.02646 [cs.AI].

[Cap+22] T. Capretto, C. Piho, R. Kumar, J. Westfall, T. Yarkoni, and O. A. Martin. "Bambi: A Simple Interface for Fitting Bayesian Linear Models in Python". In: *Journal of Statistical Software* 103.15 (2022), 1–29.

[Car03] P. Carbonetto. "Unsupervised Statistical Models for General Object Recognition". MA thesis. University of British Columbia, 2003.

[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.

[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).

[Car+21] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. *Emerging Properties in Self-Supervised Vision Transformers*. 2021. arXiv: 2104.14294 [cs.CV].

[Car97] R. Caruana. "Multitask Learning". In: *Machine Learning* 28.1 (1997), pp. 41–75.

[Cat08] A. Caticha. *Lectures on Probability, Entropy, and Statistical Physics*. 2008. arXiv: 0808.0012 [physics.data-an].

[Cat+11] A. Caticha, A. Mohammad-Djafari, J.-F. Bercher, and P. Bessiére. "Entropic Inference". In: *AIP Conference Proceedings* 1305.1 (2011), pp. 20–29. eprint: https://aip.scitation.org/doi/pdf/10.1063/1.3573619.

[CB20] Y. Chen and P. Bühlmann. "Domain adaptation under structural causal models". In: *JMLR* (2020).

[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.

[CBR20] Z. Chen, Y. Bei, and C. Rudin. "Concept whitening for interpretable image recognition". In: *Nature Machine Intelligence* 2.12 (2020), pp. 772–782.

[CC84] 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.

[CC96] M. Cowles and B. Carlin. "Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review". In: *JASA* 91 (1996), pp. 883–904.

[CCS22] A. Corenflos, N. Chopin, and S. Särkkä. "De-Sequentialized Monte Carlo: a parallel-in-time particle smoother". In: (2022). arXiv: 2202.02264 [stat.CO].

[CDC15] C. Chen, N. Ding, and L. Carin. "On the Convergence of Stochastic Gradient MCMC Algorithms with High-Order Integrators". In: *NIPS*. 2015.

[CDS02] M. Collins, S. Dasgupta, and R. E. Schapire. "A Generalization of Principal Components Analysis to the Exponential Family". In: *NIPS-14*. 2002.

[CDS19] A. Clark, J. Donahue, and K. Simonyan. "Adversarial video generation on complex datasets". In: *arXiv preprint arXiv:1907.06571* (2019).

[Cér+12] F Cérou, P Del Moral, T Furon, and A Guyader. "Sequential Monte Carlo for rare event estimation". In: *Stat. Comput.* 22.3 (2012), pp. 795–808.

[CFG14] T. Chen, E. B. Fox, and C. Guestrin. "Stochastic Gradient Hamiltonian Monte Carlo". In: *ICML*. 2014.

[CG00] S. S. Chen and R. A. Gopinath. "Gaussianization". In: *NIPS*. 2000, pp. 423–429.

[CG15] X. Chen and A. Gupta. "Webly supervised learning of convolutional networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1431–1439.

[CG18] C. Ceylan and M. U. Gutmann. "Conditional Noise-Contrastive Estimation of Unnormalised Models". In: *International Conference on Machine Learning*. 2018, pp. 726–734.

[CG96] S. Chen and J. Goodman. "An empirical study of smoothing techniques for language modeling". In: *Proc. 34th ACL*. 1996, pp. 310–318.

[CG98] S. Chen and J. Goodman. *An empirical study of smoothing techniques for language modeling*. Tech. rep. TR-10-98. Dept. Comp. Sci., Harvard, 1998.

[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 (2006), pp. 675–692.

[CGS15] T. Chen, I. Goodfellow, and J. Shlens. "Net2net: Accelerating learning via knowledge transfer". In: *International Conference on Learning Representations*. 2015.

[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.

[Cha12] K. M. A. Chai. "Variational Multinomial Logit Gaussian Process". In: *JMLR* 13.Jun (2012), pp. 1745–1808.

[Cha14] N. Chapados. "Effective Bayesian Modeling of Groups of Related Count Time Series". In: *ICML*. 2014.

[Cha+14] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. "Return of the devil in the details: Delving deep into convolutional nets". In: *British Machine Vision Conference*. 2014.

[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.

[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*. 2018.

[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.

[Cha+19b] J. J. Chandler, I. Martinez, M. M. Finucane, J. G. Terziev, and A. M. Resch. "Speaking on Data's Behalf: What Researchers Say and How Audiences Choose". en. In: *Eval. Rev.* (2019), p. 193841X19834968.

[Cha+19c] T. Chavdarova, G. Gidel, F. Fleuret, and S. Lacoste-Julien. "Reducing noise in GAN training with variance reduced extragradient". In: *Advances in Neural Information Processing Systems*. 2019, pp. 393–403.

[Cha21] S. H. Chan. *Introduction to Probability for Data Science*. Michigan Publishing, 2021.

[Cha+22] H. Chang, H. Zhang, L. Jiang, C. Liu, and W. T. Freeman. "MaskGIT: Masked Generative Image Transformer". In: *CVPR*. Feb. 2022.

[Che+05] G. Chechik, A. Globerson, N. Tishby, and Y. Weiss. "Information Bottleneck for Gaussian Variables". In: *JMLR* 6.Jan (2005), pp. 165–188.

[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.

[Che+16] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: *NIPS*. 2016.

[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).

[Che17] C. Chelba. *Language Modeling in the Era of Abundant Data*. AI with the best. 2017.

[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).

[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.

[Che+17c] X. Chen, N. Mishra, M. Rohaninejad, and P. Abbeel. "PixelSNAIL: An Improved Autoregressive Generative Model". In: (2017). arXiv: 1712.09763 [cs.LG].

[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.

[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).

[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 (2018), p. 12101.

[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).

[Che+18c] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. "Neural Ordinary Differential Equations". In: *NIPS*. 2018.

[Che+18d] X. Cheng, N. S. Chatterji, P. L. Bartlett, and M. I. Jordan. "Underdamped Langevin MCMC: A non-asymptotic analysis". In: *COLT*. 2018.

[Che+19] R. T. Q. Chen, J. Behrmann, D. Duvenaud, and J.-H. Jacobsen. "Residual Flows for Invertible Generative Modeling". In: *NIPS*. 2019.

[Che+20a] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever. "Generative pretraining from pixels". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 1691–1703.

[Che+20b] 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).

[Che+20c] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. "A simple framework for contrastive learning of visual representations". In: *ICML*. 2020.

[Che+22] D. Chen, D. Wang, T. Darrell, and S. Ebrahimi. "Contrastive Test-Time Adaptation". In: *CVPR*. Apr. 2022.

[Che95] Y. Cheng. "Mean shift, mode seeking, and clustering". In: *IEEE PAMI* 17.8 (1995).

[Chi14] S. Chiappa. "Explicit-Duration Markov Switching Models". In: *Foundations and Trends in Machine Learning* 7.6 (2014), pp. 803–886.

[Chi21a] R. Child. "Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images". In: *ICLR*. 2021.

[Chi21b] R. Child. "Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images". In: *ArXiv* abs/2011.10650 (2021).

[CHL+05] S. Chopra, R. Hadsell, Y. LeCun, et al. "Learning a similarity metric discriminatively, with application to face verification". In: *CVPR*. 2005, pp. 539–546.

[CHM97] D. M. Chickering, D. Heckerman, and C. Meek. "A Bayesian Approach to Learning Bayesian Networks with Local Structure". In: *UAI*. UAI'97. 1997, pp. 80–89.

[Cho02] N. Chopin. "A Sequential Particle Filter Method for Static Models". In: *Biometrika* 89.3 (2002), pp. 539–551.

[Cho11] M. J. Choi. "Trees and Beyond: Exploiting and Improving Tree-Structured Graphical Models". PhD thesis. MIT, 2011.

[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.

[Cho21] F. Chollet. *Deep learning with Python (second edition)*. Manning, 2021.

[Cho+22] J. Choi, J. Lee, C. Shin, S. Kim, H. Kim, and S. Yoon. "Perception Prioritized Training of Diffusion Models". In: *CVPR*. Apr. 2022.

[Cho57] N. Chomsky. *Syntactic Structures*. Mouton, 1957.

[Chr+21] R. Christiansen, N. Pfister, M. E. Jakobsen, N. Gnecco, and J. Peters. "A causal framework for distribution generalization". In: *IEEE PAMI* (2021).

[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.

[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.

[Chu+19] G. Chuang, G. DeSalvo, L. Karydas, J.-F. Kagy, A. Rostamizadeh, and A Theeraphol. "Active Learning Empirical Study". In: *NeurIPS LIRE Workshop*. 2019.

[Chw+15] K. Chwialkowski, A. Ramdas, D. Sejdinovic, and A. Gretton. "Fast Two-Sample Testing with Analytic Representations of Probability Measures". In: *NIPS*. 2015.

[CI13] M. Clydec and E. S. Iversen. "Bayesian model averaging in the M-open framework". In: *Bayesian Theory and Applications*. Ed. by P. Damien. Jan. 2013.

[CI81] W. J. Conover and R. L. Iman. "Rank Transformations as a Bridge Between Parametric and Nonparametric Statistics". In: *Am. Stat.* 35.3 (1981), pp. 124–129.

[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.

[CK05] M. Collins and T. Koo. "Discriminative Reranking for Natural Language Parsing". In: *Proc. ACL*. 2005.

[CK07] J. J. F. Commandeur and S. J. Koopman. *An Introduction to State Space Time Series Analysis (Practical Econometrics)*. en. 1st ed. Oxford University Press, 2007.

[CK21] 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.

[CK94a] 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.

[CK94b] C. Carter and R. Kohn. "On Gibbs sampling for state space models". In: *Biometrika* 81.3 (1994), pp. 541–553.

[CK96] C. Carter and R. Kohn. "Markov Chain Monte Carlo in Conditionally Gaussian State Space Models". In: *Biometrika* 83 (1996), pp. 589–601.

[CKK17] L. Chen, A. Krause, and A. Karbasi. "Interactive Submodular Bandit." In: *NIPS*. 2017, pp. 141–152.

[CL00] R. Chen and S. Liu. "Mixture Kalman Filters". In: *J. Royal Stat. Soc. B* (2000).

[CL07] L. Carvahlo and C. Lawrence. "Centroid estimation in discrete high-dimensional spaces with applications in biology". In: *PNAS* 105.4 (2007).

[CL11] O. Chapelle and L. Li. "An empirical evaluation of Thompson sampling". In: *NIPS*. 2011.

[CL18] Z. Chen and B. Liu. *Lifelong Machine Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan Claypool, 2018.

[CL96] B. P. Carlin and T. A. Louis. *Bayes and Empirical Bayes Methods for Data Analysis*. Chapman and Hall, 1996.

[Cla20] P. Clavier. "Sum-Product Network in the context of missing data". en. MA thesis. KTH, 2020.

[Cla21] A. Clayton. *Bernoulli's Fallacy: Statistical Illogic and the Crisis of Modern Science*. en. Columbia University Press, 2021.

[CLD18] C. Cremer, X. Li, and D. Duvenaud. "Inference Suboptimality in Variational Autoencoders". In: *ICML*. 2018.

[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.

[Clo20] Cloudera. *Causality for ML*. 2020.

[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 (2019), pp. 185–204.

[CLW18] Y. Chen, L. Li, and M. Wang. "Scalable Bilinear π-Learning Using State and Action Features". In: *ICML*. 2018, pp. 833–842.

[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.

[CMD17] C. Cremer, Q. Morris, and D. Duvenaud. "Reinterpreting Importance-Weighted Autoencoders". In: *ICLR Workshop*. 2017.

[CMJ22] P. G. Chang, K. P. Murphy, and M. Jones. "On diagonal approximations to the extended Kalman filter for online training of Bayesian neural networks". In: *Continual Lifelong Learning Workshop at ACML 2022*. Dec. 2022.

[CMR05] O. Cappe, E. Moulines, and T. Ryden. *Inference in Hidden Markov Models*. Springer, 2005.

[CMR12] C. Cortes, M. Mohri, and A. Rostamizadeh. "Algorithms for learning kernels based on centered alignment". In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 795–828.

[CMS12] D. Ciregan, U. Meier, and J. Schmidhuber. "Multi-column deep neural networks for image classification". In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3642–3649.

[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).

[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*. 2020.

[CO06] N. Chater and M. Oaksford. "Mental mechanisms". In: *Information sampling and adaptive cognition* (2006), pp. 210–236.

[COB18] L. Chizat, E. Oyallon, and F. Bach. "On Lazy Training in Differentiable Programming". In: (2018). arXiv: 1812.07956 [math.OC].

[Coh94] J. Cohen. "The earth is round ($p < .05$)". In: *American Psychologist* 49.12 (1994), pp. 997–1003.

[Col21] E. Collins. *LaMDA: our breakthrough conversation technology*. https://blog.google/technology/ai/lamda/. Accessed: NA-NA-NA. May 2021.

[Coo05] J. Cook. *Exact Calculation of Beta Inequalities*. Tech. rep. M. D. Anderson Cancer Center, Dept. Biostatistics, 2005.

[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.

[Cor17] G. Cormode. "Data sketching". In: *Communications of the ACM* 60.9 (2017), pp. 48–55.

[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.

[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 (1987), pp. 262–280.

[Cou16] Council of European Union. *General Data Protection Regulation*. 2016.

[Cou+20] J. Courts, A. Wills, T. Schön, and B. Ninness. "Variational System Identification for Nonlinear State-Space Models". In: (2020). arXiv: 2012.05072 [stat.ML].

[Cou+21] J. Courts, J. Hendriks, A. Wills, T. Schön, and B. Ninness. "Variational State and Parameter Estimation". In: *19th IFAC Symposium on System Identification SYSID 2021*. 2021.

[Cov99] T. M. Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[Cox06] D. R. Cox. *Principles of Statistical Inference*. en. Illustrated edition. Cambridge University Press, Aug. 2006.

[Cox46] R. T. Cox. "Probability, Frequency and Reasonable Expectation". In: *Am. J. Phys.* 14.1 (Jan. 1946), pp. 1–13.

[Cox61] R. Cox. *Algebra of Probable Inference*. en. Johns Hopkins University Press, 1961.

[CP20a] R. Chen and I. C. Paschalidis. *Distributionally Robust Learning*. NOW Foundations and Trends in Optimization, 2020.

[CP20b] N. Chopin and O. Papaspiliopoulos. *An Introduction to Sequential Monte Carlo*. en. 1st ed. Springer, 2020.

[CPD17] P. Constantinou and A Philip Dawid. "Extended conditional independence and applications in causal inference". en. In: *Ann. Stat.* 45.6 (2017), pp. 2618–2653.

[CPS10] C. Carvahlo, N. Polson, and J. Scott. "The horseshoe estimator for sparse signals". In: *Biometrika* 97.2 (2010), p. 465.

[Cri+02] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola. "On Kernel-Target Alignment". In: *Advances in Neural Information Processing Systems*. 2002, pp. 367–373.

[CRK19] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter. "Certified adversarial robustness via randomized smoothing". In: *arXiv preprint arXiv:1902.02918* (2019).

[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*. 2011, pp. 1–8.

[CS04] I. Csiszár and P. C. Shields. "Information theory and statistics: A tutorial". In: (2004).

[CS09] Y. Cho and L. K. Saul. "Kernel Methods for Deep Learning". In: *NIPS*. 2009, pp. 342–350.

[CS18] P. Chaudhari and S. Soatto. "Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks". In: *ICLR*. 2018.

[CSF16] A. W. Churchill, S. Sigtia, and C. Fernando. "Learning to Generate Genotypes with Neural Networks". In: (2016). arXiv: 1604.04153 [cs.NE].

[CSG16] K. Chwialkowski, H. Strathmann, and A. Gretton. "A Kernel Test of Goodness of Fit". In: *ICML*. Feb. 2016.

[Csi67] I. Csiszar. "Information-Type Measures of Difference of Probability Distributions and Indirect Observations". In: *Studia Scientiarum Mathematicarum Hungarica* 2 (1967), pp. 299–318.

[CSN21] J. Coullon, L. South, and C. Nemeth. "Stochastic Gradient MCMC with Multi-Armed Bandit Tuning". In: (2021). arXiv: 2105.13059 [stat.CO].

[CSZ06] O. Chapelle, B. Scholkopf, and A. Zien, eds. *Semi-Supervised Learning*. MIT Press, 2006.

[CT06] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. 2nd edition. John Wiley, 2006.

[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. Association for Computing Machinery, 2019, pp. 221–236.

[CT91] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley, 1991.

[CTM17] M. F. Cusumano-Towner and V. K. Mansinghka. "AIDE: An algorithm for measuring the accuracy of probabilistic inference algorithms". In: *NIPS*. 2017.

[CTN17] Y. Chali, M. Tanvee, and M. T. Nayeem. "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.

[CTS78] C. Cannings, E. A. Thompson, and M. H. Skolnick. "Probability functions in complex pedigrees". In: *Advances in Applied Probability* 10 (1978), pp. 26–61.

[CUH16] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". In: *ICLR*. 2016.

[Cui+18] Y. Cui, Y. Song, C. Sun, A. Howard, and S. Belongie. "Large scale fine-grained categorization and domain-specific transfer learning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4109–4118.

[Cun22] Y. L. Cun. *A path towards autonomous AI*. 2022.

[Cun83] W. H. Cunningham. "Decomposition of submodular functions". In: *Combinatorica* 3.1 (1983), pp. 53–68.

[Cut13] M. Cuturi. "Sinkhorn Distances: Lightspeed Computation of Optimal Transportation Distances". In: *NIPS*. 2013.

[CW07] C. M. Carvahlo and M. West. "Dynamic Matrix-Variate Graphical Models". In: *Bayesian Analysis* 2.1 (2007), pp. 69–98.

[CW16] T. Cohen and M. Welling. "Group Equivariant Convolutional Networks". en. In: *ICML*. 2016, pp. 2990–2999.

[CWG20] D. C. Castro, I. Walker, and B. Glocker. "Causality matters in medical imaging". en. In: *Nat. Commun.* 11.1 (2020), p. 3673.

[CWS21] J. Courts, A. G. Wills, and T. B. Schön. "Gaussian Variational State Estimation for Nonlinear State-Space Models". In: *IEEE Trans. Signal Process.* 69 (2021), pp. 5979–5993.

[CXH21] X. Chen, S. Xie, and K. He. "An empirical study of training self-supervised vision transformers". In: *arXiv preprint arXiv:2104.02057* (2021).

[CY20] G. Cormode and K. Yi. *Small Summaries for Big Data*. Cambridge University Press, 2020.

[Cza+20] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison. "DeepFactors: Real-Time Probabilistic Dense Monocular SLAM". In: *ICRA*. 2020.

[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).

[CZS22] A. Corenflos, Z. Zhao, and S. Särkkä. "Temporal Gaussian Process Regression in Logarithmic Time". In: *2022 25th International Conference on Information Fusion (FUSION)*. July 2022, pp. 1–5.

[D'A+20] A. D'Amour et al. "Underspecification Presents Challenges for Credibility in Modern Machine Learning". In: (2020). arXiv: 2011.03395 [cs.LG].

[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.

[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.

[Dai+17] H. Dai, B. Dai, Y.-M. Zhang, S. Li, and L. Song. "Recurrent Hidden Semi-Markov Model". In: *ICLR*. 2017.

[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.

[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.

[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.

[Dai+20a] C. Dai, J. Heng, P. E. Jacob, and N. Whiteley. "An invitation to sequential Monte Carlo samplers". In: (2020). arXiv: 2007.11936 [stat.CO].

[Dai+20b] Z. Dai, G. Lai, Y. Yang, and Q. V. Le. "Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing". In: *NIPS*. 2020.

[Dal+04] N. Dalvi, P. Domingos, S. Sanghai, and D. Verma. "Adversarial classification". In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2004, pp. 99–108.

[Dar03] A. Darwiche. "A Differential Approach to Inference in Bayesian Networks". In: *J. ACM* 50.3 (2003), pp. 280–305.

[Dar09] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge, 2009.

[Dar+11] S. Darolles, Y. Fan, J.-P. Florens, and E. Renault. "Nonparametric instrumental regression". In: *Econometrica* 79.5 (2011), pp. 1541–1565.

[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.

[Dau05] F Daum. "Nonlinear filters: beyond the Kalman filter". In: *IEEE Aerospace and Electronic Systems Magazine* 20.8 (Aug. 2005), pp. 57–69.

[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 (2004), pp. 353–376.

[Dav+18] T. R. Davidson, L. Falorsi, N. De Cao, T. Kipf, and J. M. Tomczak. "Hyperspherical Variational Auto-Encoders". In: *UAI*. 2018.

[Daw00] A. P. Dawid. "Causal Inference Without Counterfactuals". In: *JASA* 95.450 (2000), pp. 407–424.

[Daw02] A. P. Dawid. "Influence diagrams for causal modelling and inference". In: *Intl. Stat. Review* 70 (2002). Corrections p437, pp. 161–189.

[Daw15] A. P. Dawid. "Statistical Causality from a Decision-Theoretic Perspective". In: *Annu. Rev. Stat. Appl.* 2.1 (2015), pp. 273–303.

[Daw82] A. P. Dawid. "The Well-Calibrated Bayesian". In: *JASA* 77.379 (1982), pp. 605–610.

[Dax+21] E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig. "Laplace Redux–Effortless Bayesian Deep Learning". In: *NIPS*. 2021.

[Day+95] P. Dayan, G. Hinton, R. Neal, and R. Zemel. "The Helmholtz machine". In: *Neural Networks* 9.8 (1995).

[DBB20] S. Daulton, M. Balandat, and E. Bakshy. "Differentiable Expected Hypervolume Improvement for Parallel Multi-Objective Bayesian Optimization". In: *NIPS*. 2020.

[DBP19] C. Durkan, A. Bekasov, and I. M. G. Papamakarios. "Neural Spline Flows". In: *NIPS*. 2019.

[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.

[DCF+15] E. Denton, S. Chintala, R. Fergus, et al. "Deep generative image models using a Laplacian pyramid of adversarial networks". In: *NIPS*. 2015.

[DD22] S. Doyen and N. B. Dadario. "12 Plagues of AI in Healthcare: A Practical Guide to Current Issues With Using Machine Learning in a Medical Context". en. In: *Front Digit Health* 4 (May 2022), p. 765406.

[DDL97] S. DellaPietra, V. DellaPietra, and J. Lafferty. "Inducing features of random fields". In: *IEEE PAMI* 19.4 (1997).

[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.

[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.

[Dec96] R. Dechter. "Bucket elimination: a unifying framework for probabilistic inference". In: *UAI*. 1996.

[DeG70] M. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, 1970.

[DEL20] H. M. Dolatabadi, S. Erfani, and C. Leckie. "Invertible Generative Modeling using Linear Rational Splines". In: *AISTATS*. 2020, pp. 4236–4246.

[Del+21] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. "A continual learning survey: Defying forgetting in classification tasks". en. In: *IEEE PAMI* (2021).

[Den+02] D. Denison, C. Holmes, B. Mallick, and A. Smith. *Bayesian methods for nonlinear classification and regression*. Wiley, 2002.

[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.

[Dev+18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[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.

[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: (2021). arXiv: 2108.02498 [stat.CO].

[Dev85] P. A. Devijver. "Baum's forward-backward algorithm revisited". In: *Pattern Recognition Letters* 3.6 (1985), pp. 369–373.

[Dex] *Dex: Research language for array processing in the Haskell/ML family*. https://github.com/google-research/dex-lang. 2019.

[DF18] E. Denton and R. Fergus. "Stochastic Video Generation with a Learned Prior". In: *ICML*. 2018.

[DF19] X. Ding and D. J. Freedman. "Learning Deep Generative Models with Annealed Importance Sampling". In: (2019). arXiv: 1906.04904 [stat.ML].

[DF21] F. D'Angelo and V. Fortuin. "Repulsive Deep Ensembles are Bayesian". In: *NIPS*. May 2021.

[DFF21] S. Dozinski, U. Feige, and M. Feldman. "Are Gross Substitutes a Substitute for Submodular Valuations?" In: *arXiv preprint arXiv:2102.13343* (2021).

[DFO20] M. Deisenroth, A. Faisal, and C. S. Ong. *Mathematics for machine learning*. Cambridge, 2020.

[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 (2015), pp. 408–423.

[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.

[DG17] P. Dabkowski and Y. Gal. "Real time image saliency for black box classifiers". In: *NeurIPS* (2017).

[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.

[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.

[DGE15] C. Doersch, A. Gupta, and A. A. Efros. "Unsupervised visual representation learning by context prediction". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1422–1430.

[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.

[DH22] F. Dellaert and S. Hutchinson. *Introducion to Robotics and Perception*. 2022.

[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.

[Dia88a] P Diaconis. "Recent progress on de Finetti's notions of exchangeability". In: *Bayesian Statistics* 3 (1988).

[Dia88b] P. Diaconis. "Sufficiency as statistical symmetry". In: *Proceedings of the AMS Centennial Symposium*. 1988, pp. 15–26.

[Die00] T. G. Dietterich. "Ensemble Methods in Machine Learning". In: *Multiple Classifier Systems*. Springer Berlin Heidelberg, 2000, pp. 1–15.

[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 (2007).

[Die10] L. Dietz. *Directed Factor Graph Notation for Generative Models*. Tech. rep. MPI, 2010.

[Die+17] A. B. Dieng, C. Wang, J. Gao, and J. Paisley. "TopicRNN: A Recurrent Neural Network with Long-Range Semantic Dependency". In: *ICLR*. 2017.

[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.

[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).

[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.

[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].

[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 Rozovsk. 2011.

[DJ21] K. Desai and J. Johnson. *VirTex: Learning Visual Representations from Textual Annotations*. 2021. arXiv: 2006.06666 [cs.CV].

[DJK18] J. Djolonga, S. Jegelka, and A. Krause. "Provable Variational Inference for Constrained Log-Submodular Models". In: *NeurIPS*. 2018.

[DK12] J. Durbin and S. J. Koopman. *Time Series Analysis by State Space Methods (Second Edition)*. en. Revised ed. edition. Oxford University Press, 2012.

[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.

[DK15a] J. Djolonga and A. Krause. "Scalable variational inference in log-supermodular models". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1804–1813.

[DK15b] G. Durrett and D. Klein. "Neural CRF Parsing". In: *Proc. ACL*. 2015.

[DKB15] L. Dinh, D. Krueger, and Y. Bengio. "NICE: Non-linear Independent Components Estimation". In: *ICLR*. 2015.

[DKD16] J. Donahue, P. Krahenbuhl, and T. Darrell. "Adversarial feature learning". In: *arXiv preprint arXiv:1605.09782* (2016).

[DKD17] J. Donahue, P. Krähenbühl, and T. Darrell. *Adversarial Feature Learning*. 2017. arXiv: 1605.09782 [cs.LG].

[DKS13] J. Dick, F. Y. Kuo, and I. H. Sloan. "High-dimensional integration: the quasi-Monte Carlo way". In: *Acta Numerica* 22 (2013), 133–288.

[DL09] P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for AI*. Morgan & Claypool, 2009.

[DL10] J. V. Dillon and G. Lebanon. "Stochastic Composite Likelihood". In: *J. Mach. Learn. Res.* 11 (2010), pp. 2597–2633.

[DL13] A. Damianou and N. Lawrence. "Deep Gaussian Processes". en. In: *AISTATS*. 2013, pp. 207–215.

[DL93] P. Dagum and M. Luby. "Approximating probabilistic inference in Bayesian belief networks is NP-hard". In: *Artificial Intelligence* 60 (1993), pp. 141–153.

[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.

[DLM09] H. Daumé III, J. Langford, and D. Marcu. "Search-based Structured Prediction". In: *MLJ* 75.3 (2009), pp. 297–325.

[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.

[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.

[DLT21] M. De Lange and T. Tuytelaars. "Continual Prototype Evolution: Learning Online from Non-Stationary Data Streams". In: *ICCV*. 2021.

[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.

[DM19a] Y. Du and I. Mordatch. "Implicit Generation and Generalization in Energy-Based Models". In: (2019). arXiv: 1903.08689 [cs.LG].

[DM19b] Y. Du and I. Mordatch. "Implicit Generation and Modeling with Energy Based Models". In: *NIPS*. 2019, pp. 3608–3618.

[DM22] A. P. Dawid and M. Musio. "Effects of Causes and Causes of Effects". In: *Annu. Rev. Stat. Appl.* 9.1 (Mar. 2022), pp. 261–287.

[DMDJ12] P. Del Moral, A. Doucet, and A. Jasra. "An adaptive sequential Monte Carlo method for approximate Bayesian computation". In: *Stat. Comput.* 22.5 (2012), pp. 1009–1020.

[DMKM22] G. Duran-Martin, A. Kara, and K. Murphy. "Efficient Online Bayesian Inference for Neural Bandits". In: *AISTATS*. 2022.

[DMM17] A. P. Dawid, M. Musio, and R. Murtas. "The probability of causation". In: *Law, Probability and Risk* 16.4 (2017), pp. 163–179.

[DMP18] C. Donahue, J. McAuley, and M. Puckette. "Adversarial Audio Synthesis". In: *International Conference on Learning Representations*. 2018.

[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.

[DN21a] P. Dhariwal and A. Nichol. "Diffusion Models Beat GANs on Image Synthesis". In: *NIPS*. May 2021.

[DN21b] P. Dhariwal and A. Q. Nichol. "Diffusion Models Beat GANs on Image Synthesis". In: *NIPS*. May 2021.

[Dnp] .

[DNR11] D. Duvenaud, H. Nickisch, and C. E. Rasmussen. "Additive Gaussian Processes". In: *NIPS*. 2011.

[Dom+06] P. Domingos, S. Kok, H. Poon, M. Richardson, and P. Singla. "Unifying Logical and Statistical AI". In: *IJCAI*. 2006.

[Dom+19] A.-K. Dombrowski, M. Alber, C. J. Anders, M. Ackermann, K.-R. Müller, and P. Kessel. "Explanations can be manipulated and geometry is to blame". In: *arXiv preprint arXiv:1906.07983* (2019).

[Don+14] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. "Decaf: A deep convolutional activation feature for generic visual recognition". In: *International conference on machine learning*. 2014, pp. 647–655.

[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.

[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).

[Don+21] X. Dong, J. Bao, T. Zhang, D. Chen, W. Zhang, L. Yuan, D. Chen, F. Wen, and N. Yu. "Peco: Perceptual codebook for bert pre-training of vision transformers". In: *arXiv preprint arXiv:2111.12710* (2021).

[Doo+17] J. van Doorn, A. Ly, M. Marsman, and E.-J. Wagenmakers. "Bayesian Rank-Based Hypothesis Testing for the Rank Sum Test, the Signed Rank Test, and Spearman's ρ". In: (Dec. 2017). arXiv: 1712.06941 [stat.ME].

[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.

[Dos+21] A. Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations*. 2021.

[Doy+07] K. Doya, S. Ishii, A. Pouget, and R. P. N. Rao, eds. *Bayesian Brain: Probabilistic Approaches to Neural Coding*. MIT Press, 2007.

[DR11] M. P. Deisenroth and C. E. Rasmussen. "PILCO: A Model-Based and Data-Efficient Approach to Policy Search". In: *ICML*. 2011.

[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.

[DR21] H. Duanmu and D. M. Roy. "On extended admissibale procedures and their nonstandard Bayes risk". In: *Annals of Statistics* (2021).

[DRB19] A. B. Dieng, F. J. R. Ruiz, and D. M. Blei. "The Dynamic Embedded Topic Model". In: (2019). arXiv: 1907.05545 [cs.CL].

[DRG15] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani. "Training generative neural networks via Maximum Mean Discrepancy optimization". In: *ICML*. 2015.

[Dru08] J. Drugowitsch. *Bayesian linear regression*. Tech. rep. U. Rochester, 2008.

[DS18] J. Domke and D. R. Sheldon. "Importance Weighting and Variational Inference". In: *NIPS*. 2018, pp. 4474–4483.

[DS19] J. Donahue and K. Simonyan. "Large scale adversarial representation learning". In: *arXiv preprint arXiv:1907.02544* (2019).

[DSDB17] L. Dinh, J. Sohl-Dickstein, and S. Bengio. "Density estimation using Real NVP". In: *ICLR*. 2017.

[DSK16] V. Dumoulin, J. Shlens, and M. Kudlur. "A Learned Representation For Artistic Style". In: (2016).

[DSP21] M. Dowling, P. Sokół, and I. M. Park. "Hida-Matérn Kernel". In: (July 2021). arXiv: 2107.07098 [stat.ML].

[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.

[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.

[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.

[Du+19] S. 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: (2019). arXiv: 1905.13192 [cs.LG].

[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).

[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*. KDD '21. Association for Computing Machinery, 2021, pp. 2792–2801.

[Dua+20] S. Duan, N. Watters, L. Matthey, C. P. Burgess, A. Lerchner, and I. Higgins. "A Heuristic for Unsupervised Model Selection for Variational Disentangled Representation Learning". In: *ArXiv* abs/1905.12614 (2020).

[Dua+87] S. Duane, A. Kennedy, B. Pendleton, and D. Roweth. "Hybrid Monte Carlo". In: *Physics Letters B* 195.2 (1987), pp. 216–222.

[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". en. In: *NIPS*. Vol. 29. 2016, pp. 1154–1162.

[Dub20] Y. Dubois. *Neural Process Family*. http://yanndubs.github.io/Neural-Process-Family/. 2020.

[Dud13] J. Duda. "Asymmetric numeral systems: entropy coding combining speed of Huffman coding with compression rate of arithmetic coding". In: (2013). arXiv: 1311.2540 [cs.IT].

[Duf02] M. Duff. "Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes". PhD thesis. U. Mass. Dept. Comp. Sci., 2002.

[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).

[Dum+17] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. *Adversarially Learned Inference*. 2017. arXiv: 1606.00704 [stat.ML].

[Dum+18] A. Dumitrache, O. Inel, B. Timmermans, C. Ortiz, R.-J. Sips, L. Aroyo, and C. Welty. "Empirical Methodology for Crowdsourcing Ground Truth". In: *Semantic Web Journal* (2018).

[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.

[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.

[Duv+13] D. Duvenaud, J. Lloyd, R. Grosse, J. Tenenbaum, and G. Zoubin. "Structure Discovery in Nonparametric Regression through Compositional Kernel Search". en. In: *ICML*. 2013, pp. 1166–1174.

[Duv14] D. Duvenaud. "Automatic Model Construction with Gaussian Processes". PhD thesis. Computational and Biological Learning Laboratory, University of Cambridge, 2014.

[dV04] 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.

[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.

[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.

[DV99] A. P. Dawid and V. Vovk. "Prequential probability: Principles and properties". In: *Bernoulli* 5 (1999), pp. 125–162.

[DVK17] F. Doshi-Velez and B. Kim. *Towards A Rigorous Science of Interpretable Machine Learning*. 2017. eprint: 1702.08608 (stat.ML).

[DW19] B. Dai and D. Wipf. "Diagnosing and Enhancing VAE Models". In: *ICLR*. 2019.

[DWS12] T. Degris, M. White, and R. S. Sutton. "Off-Policy Actor-Critic". In: *ICML*. 2012.

[DWW19] B. Dai, Z. Wang, and D. Wipf. "The Usual Suspects? Reassessing Blame for VAE Posterior Collapse". In: (2019). arXiv: 1912.10702 [cs.LG].

[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.

[DY79] P. Diaconis and D. Ylvisaker. "Conjugate priors for exponential families". In: vol. 7. 1979, pp. 269–281.

[ECM18] P. Etoori, M. Chinnakotla, and R. Mamidi. "Automatic Spelling Correction for Resource-Scarce Languages using Deep Learning". In: *Proceedings of ACL 2018, Student Research Workshop*. Association for Computational Linguistics, 2018, pp. 146–152.

[ED05] D. Earl and M. Deem. "Parallel tempering: Theory, applications, and new perspectives". In: *Phys. Chem. Chem. Phys.* 7 (2005), p. 3910.

[Edm69] H. P. Edmundson. "New methods in automatic extracting". In: *Journal of the ACM (JACM)* 16.2 (1969), pp. 264–285.

[Edm70] J. Edmonds. "Matroids, submodular functions, and certain polyhedra". In: *Combinatorial Structures and Their Applications* (1970), pp. 69–87.

[EFL04] E. Erosheva, S. Fienberg, and J. Lafferty. "Mixed-membership models of scientific publications". In: *PNAS* 101 (2004), pp. 5220–2227.

[Efr86] B. Efron. "Why Isn't Everyone a Bayesian?" In: *The American Statistician* 40.1 (1986).

[EGW05] D. Ernst, P. Geurts, and L. Wehenkel. "Tree-Based Batch Mode Reinforcement Learning". In: *JMLR* 6 (2005), pp. 503–556.

[EH16] B. Efron and T. Hastie. *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. en. Cambridge University Press, June 2016.

[Eis16] J. Eisner. "Inside-Outside and Forward-Backward Algorithms Are Just Backprop (Tutorial Paper)". In: *EMNLP Workshop on Structured Prediction for NLP*. 2016.

[Eke+13] M. Ekeberg, C. Lövkvist, Y. Lan, M. Weigt, and E. Aurell. "Improved contact prediction in proteins: using pseudolikelihoods to infer Potts models". en. In: *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.* 87.1 (2013), p. 012707.

[EM07] D. Eaton and K. Murphy. "Exact Bayesian structure learning from uncertain interventions". In: *AI/Statistics*. 2007.

[EMH19] T. Elsken, J. H. Metzen, and F. Hutter. "Neural Architecture Search: A Survey". In: *JMLR* 20 (2019), pp. 1–21.

[EMK06] G. Elidan, I. McGraw, and D. Koller. "Residual belief propagation: Informed scheduling for asynchronous message passing". In: *UAI*. 2006.

[Eng+18] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts. "GANSynth: Adversarial Neural Audio Synthesis". In: *International Conference on Learning Representations*. 2018.

[Erh+09] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. "Visualizing higher-layer features of a deep network". In: *University of Montreal* 1341.3 (2009), p. 1.

[Erm+13] S. Ermon, C. P. Gomes, A. Sabharwal, and B. Selman. "Taming the Curse of Dimensionality: Discrete Integration by Hashing and Optimization". In: *ICML*. Feb. 2013.

[ERO21] P. Esser, R. Rombach, and B. Ommer. "Taming Transformers for High-Resolution Image Synthesis". In: *CVPR*. 2021.

[ET93] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. en. 1st ed. Chapman and Hall/CRC, Jan. 1993.

[Etz+18] A. Etz, J. M. Haaf, J. N. Rouder, and J. Vandekerckhove. "Bayesian Inference and Testing Any Hypothesis You Can Specify". In: *Advances in Methods and Practices in Psychological Science* 1.2 (June 2018), pp. 281–295.

[Eva+18] R. Evans et al. "De novo structure prediction with deep-learning based scoring". In: (2018).

[Eve09] G. Evensen. *Data Assimilation: The Ensemble Kalman Filter*. en. 2nd ed. 2009 edition. Springer, 2009.

[EW10] El-Yaniv and Wiener. "On the Foundations of Noise-free Selective Classification". In: *JMLR* (2010).

[EY09] M. Elad and I. Yavnch. "A plurality of sparse representations is better than the sparsest one alone". In: *IEEE Trans. on Info. Theory* 55.10 (2009), pp. 4701–4714.

[Eyk+18] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. "Robust Physical-World Attacks on Deep Learning Models". In: *CVPR*. 2018.

[Eys+21] B. Eysenbach, A. Khazatsky, S. Levine, and R. Salakhutdinov. "Mismatched No More: Joint Model-Policy Optimization for Model-Based RL". In: (2021). arXiv: 2110.02758 [cs.LG].

[FA20] I. Fischer and A. A. Alemi. "CEB Improves Model Robustness". In: *Entropy* 22.10 (2020), p. 1081.

[Fad+20] S. G. Fadel, S. Mair, R. da S. Torres, and U. Brefeld. "Principled Interpolation in Normalizing Flows". In: (2020). arXiv: 2010.12059 [stat.ML].

[Fag+18] F. Faghri, D. J. Fleet, J. R. Kiros, and S. Fidler. "VSE++: Improving Visual-Semantic Embeddings with Hard Negatives". In: *BMVC*. 2018.

[FAL17] C. Finn, P. Abbeel, and S. Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *ICML*. 2017.

[Far22] S. Farquhar. "Understanding Approximation for Bayesian Inference in Neural Networks". PhD thesis. Oxford, Nov. 2022.

[Fau+18] L. Faury, F. Vasile, C. Calauzènes, and O. Fercoq. "Neural Generative Models for Global Optimization with Gradients". In: (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 (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 hidden Markov models via particle filters". en. In: *J. of Royal Stat. Soc. Series B* 65.4 (2003), pp. 887–899.

[FD07a] B. Frey and D. Dueck. "Clustering by Passing Messages 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 Observable Implications". In: *Journal of the American Statistical Association* 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 Regression Models". In: (2019). arXiv: 1911.06743 [stat.ME].

[FE73] M. Fischler and R. Elschlager. "The representation and matching of pictorial structures". In: *IEEE Trans. on Computer* 22.1 (1973).

[Fed+18] W. Fedus, M. Rosca, B. Lakshminarayanan, A. M. Dai, S. Mohamed, and I. Goodfellow. "Many Paths to Equilibrium: GANs Do Not Need to Decrease a Divergence At Every Step". In: *International Conference on Learning Representations*. 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 Reinforcement 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 Augmentation Approaches for NLP". In: (2021). arXiv: 2105.03075 [cs.CL].

[Feu+15] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. "Efficient and Robust Automated Machine 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: *Probability Theory and Related Fields* 162.3 (2015), pp. 707–738.

[FG18] S. Farquhar and Y. Gal. "Towards Robust Evaluations of Continual Learning". In: (2018). arXiv: 1805.09733 [stat.ML].

[FGG97] N. Friedman, D. Geiger, and M. Goldszmidt. "Bayesian network classifiers". In: *MLJ* 29 (1997), pp. 131–163.

[FH12] P. F. Felzenszwalb and D. P. Huttenlocher. "Distance Transforms of Sampled Functions". In: *Theory of Computing* 8.19 (2012), pp. 415–428.

[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 (2020).

[FH75] K Fukunaga and L Hostetler. "The estimation of the gradient of a density function, with applications in pattern recognition". In: *IEEE Trans. Inf. Theory* 21.1 (1975), pp. 32–40.

[FH97] B. J. Frey and G. Hinton. "Efficient stochastic source coding and an application to a Bayesian network source model". In: *Computer Journal* (1997).

[FHDV20] J. Futoma, M. C. Hughes, and F. Doshi-Velez. "Popcorn: Partially observed prediction constrained reinforcement learning". In: *AISTATS* (2020).

[FHHL20] G. Flamich, M. Havasi, and J. M. Hernández-Lobato. "Compressing images by encoding their latent representations with relative entropy coding". In: vol. 33. 2020, pp. 16131–16141.

[FHK03] P. Felzenszwalb, D. Huttenlocher, and J. Kleinberg. "Fast Algorithms for Large State Space HMMs with Applications to Web Usage Analysis". In: *NIPS*. 2003.

[FHL19] S. Fort, H. Hu, and B. Lakshminarayanan. "Deep Ensembles: A Loss Landscape Perspective". In: (2019). arXiv: 1912.02757 [stat.ML].

[FHM18] S. Fujimoto, H. van Hoof, and D. Meger. "Addressing Function Approximation Error in Actor-Critic Methods". In: *ICLR*. 2018.

[FHT08] J. Friedman, T. Hastie, and R. Tibshirani. "Sparse inverse covariance estimation the graphical lasso". In: *Biostatistics* 9.3 (2008), pp. 432–441.

[FI10] A. Fischer and C. Igel. "Empirical analysis of the divergence of Gibbs sampling based learning algorithms for restricted Boltzmann machines". In: *International conference on artificial neural networks*. Springer. 2010, pp. 208–217.

[Fie70] S. Fienberg. "An Iterative Procedure for Estimation in Contingency Tables". In: *Annals of Mathematical Statistics* 41.3 (1970), pp. 907–917.

[Fin+16] C. Finn, P. Christiano, P. Abbeel, and S. Levine. "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models". In: *arXiv preprint arXiv:1611.03852* (2016).

[Fis20] I. Fischer. "The Conditional Entropy Bottleneck". In: *Entropy* 22.9 (2020).

[Fis25] R. Fisher. *Statistical Methods for Research Workers*. Biological monographs and manuals. Oliver and Boyd, 1925.

[FJ02] M. A. T. Figueiredo and A. K. Jain. "Unsupervised Learning of Finite Mixture Models". In: *IEEE PAMI* 24.3 (2002), pp. 381–396.

[FJL18] R. Frostig, M. J. Johnson, and C. Leary. "Compiling machine learning programs via high-level tracing". In: *Machine Learning and Systems (MLSys)* (2018).

[FK13a] V. Feldman and P. Kothari. "Learning Coverage Functions". In: *CoRR* abs/1304.2079 (2013). arXiv: 1304.2079.

[FK13b] M Frei and H. R. Künsch. "Bridging the ensemble Kalman and particle filters". In: *Biometrika* 100.4 (2013), pp. 781–800.

[FK14] V. Feldman and P. Kothari. "Learning Coverage Functions and Private Release of Marginals". In: *Proceedings of The 27th Conference on Learning Theory*. Ed. by M. F. Balcan, V. Feldman, and C. Szepesvári. Vol. 35. Proceedings of Machine Learning Research. Barcelona, Spain: PMLR, 2014, pp. 679–702.

[FK21] A. Fisher and E. H. Kennedy. "Visually Communicating and Teaching Intuition for Influence Functions". In: *The American Statistician* 75.2 (2021), pp. 162–172. eprint: https://doi.org/10.1080/00031305.2020.1717620.

[FKH17] S. Falkner, A. Klein, and F. Hutter. "Combining Hyperband and Bayesian Optimization". In: *NIPS 2017 Bayesian Optimization Workshop*. 2017.

[FKV13] V. Feldman, P. Kothari, and J. Vondrák. "Representation, Approximation and Learning of Submodular Functions Using Low-rank Decision Trees". In: *Proceedings of the 26th Annual Conference on Learning Theory*. Ed. by S. Shalev-Shwartz and I. Steinwart. Vol. 30. Proceedings of Machine Learning Research. Princeton, NJ, USA: PMLR, 2013, pp. 711–740.

[FKV14] V. Feldman, P. Kothari, and J. Vondrák. "Nearly tight bounds on l1 approximation of self-bounding functions". In: *CoRR, abs/1404.4702* 1 (2014).

[FKV17] V. Feldman, P. Kothari, and J. Vondrák. "Tight Bounds on ℓ_1 Approximation and Learning of Self-Bounding Functions". In: *International Conference on Algorithmic Learning Theory*. PMLR. 2017, pp. 540–559.

[FKV20] V. Feldman, P. Kothari, and J. Vondrák. "Tight bounds on ℓ_1 approximation and learning of self-bounding functions". In: *Theoretical Computer Science* 808 (2020), pp. 86–98.

[FL07] P. Fearnhead and Z. Liu. "Online Inference for Multiple Changepoint Problems". In: *J. of Royal Stat. Soc. Series B* 69 (2007), pp. 589–605.

[FL11] P. Fearnhead and Z. Liu. "Efficient Bayesian analysis of multiple changepoint models with dependence across segments". In: *Statistics and Computing* 21.2 (2011), pp. 217–229.

[FL+18] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau. "An Introduction to Deep Reinforcement Learning". In: *Foundations and Trends in Machine Learning* 11.3 (2018).

[FLA16] C. Finn, S. Levine, and P. Abbeel. "Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization". In: *ICML*. 2016, pp. 49–58.

[Fla+16] S. Flaxman, D. Sejdinovic, J. P. Cunningham, and S. Filippi. "Bayesian Learning of Kernel Embeddings". In: *UAI*. 2016.

[FLL18] J. Fu, K. Luo, and S. Levine. "Learning Robust Rewards with Adverserial Inverse Reinforcement Learning". In: *ICLR*. 2018.

[FLL19] Y. Feng, L. Li, and Q. Liu. "A Kernel Loss for Solving the Bellman Equation". In: *NeurIPS*. 2019, pp. 15430–15441.

[FLMM21] D. T. Frazier, R. Loaiza-Maya, and G. M. Martin. "Variational Bayes in State Space Models: Inferential and Predictive Accuracy". In: (June 2021). arXiv: 2106.12262 [stat.ME].

[FMM18] M. Figurnov, S. Mohamed, and A. Mnih. "Implicit Reparameterization Gradients". In: *NIPS*. 2018.

[FMP19] S. Fujimoto, D. Meger, and D. Precup. "Off-Policy Deep Reinforcement Learning without Exploration". In: *ICML*. 2019, pp. 2052–2062.

[FNG00] N. de Freitas, M. Niranjan, and A. Gee. "Hierarchical Bayesian models for regularisation in sequential learning". In: *Neural Computation* 12.4 (2000), pp. 955–993.

[FNW78] M. Fisher, G. Nemhauser, and L. Wolsey. "An analysis of approximations for maximizing submodular set functions—II". In: *Polyhedral combinatorics* (1978), pp. 73–87.

[FO20] F. Farnia and A. Ozdaglar. "Do GANs always have Nash equilibria?" In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by H. D. III and A. Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 3029–3039.

[Fon+21] D. Fontanel, F. Cermelli, M. Mancini, and B. Caputo. "On the Challenges of Open World Recognition under Shifting Visual Domains". In: *ICRA*. 2021.

[For01] G. D. Forney. "Codes on graphs: normal realizations". In: *IEEE Trans. Inf. Theory* 47.2 (2001), pp. 520–548.

[For+18a] V. Fortuin, G. Dresdner, H. Strathmann, and G. Rätsch. "Scalable Gaussian Processes on Discrete Domains". In: (2018). arXiv: 1810.10368 [stat.ML].

[For+18b] M. Fortunato et al. "Noisy Networks for Exploration". In: *ICLR*. 2018.

[For+19] N. Ford, J. Gilmer, N. Carlini, and D. Cubuk. "Adversarial Examples Are a Natural Consequence of Test Error in Noise". In: (2019). arXiv: 1901.10513 [cs.LG].

[For22] V. Fortuin. "Priors in Bayesian Deep Learning: A Review". In: *Intl. Statisical Review* (2022). arXiv: 2105.06868 [stat.ML].

[For+95] J. Forbes, T. Huang, K. Kanazawa, and S. Russell. "The BATmobile: Towards a Bayesian Automated Taxi". In: *IJCAI*. 1995.

[Fot+14] N. Foti, J. Xu, D. Laird, and E. Fox. "Stochastic variational inference for hidden Markov models". In: *NIPS*. 2014, pp. 3599–3607.

[Fox+08] E. Fox, E. Sudderth, M. Jordan, and A. Willsky. "An HDP-HMM for Systems with State Persistence". In: *ICML*. 2008.

[Fox09] E. B. Fox. "Bayesian nonparametric learning of complex dynamical phenomena". PhD thesis. Massachusetts Institute of Technology, 2009.

[FP08] N Friel and A. N. Pettitt. "Marginal Likelihood Estimation via Power Posteriors". In: *J. of Royal Stat. Soc. Series B* 70.3 (2008), pp. 589–607.

[FP69] D. C. Fraser and J. E. Potter. "The optimum linear smoother as a combination of two optimum linear filters". In: *IEEE Trans. on Automatical Control* (1969), pp. 387–390.

[FPD09] P. Frazier, W. Powell, and S. Dayanik. "The knowledge-gradient policy for correlated normal beliefs". In: *INFORMS J. on Computing* 21.4 (2009), pp. 599–613.

[Fra08] A. Fraser. *Hidden Markov Models and Dynamical Systems*. SIAM Press, 2008.

[Fra+16] M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther. "Sequential Neural Models with Stochastic Layers". In: *NIPS*. 2016.

[Fra18] P. I. Frazier. "Bayesian Optimization". In: *Recent Advances in Optimization and Modeling of Contemporary Problems*. INFORMS TutORials in Operations Research. INFORMS, 2018, pp. 255–278.

[Fre14] A. A. Freitas. "Comprehensible classification models: a position paper". In: *ACM SIGKDD explorations newsletter* 15.1 (2014), pp. 1–10.

[Fre+22] M. Freitag, D. Grangier, Q. Tan, and B. Liang. "High Quality Rather than High Model Probability: Minimum Bayes Risk Decoding with Neural Metrics". In: *TACL*. 2022.

[Fre98] B. Frey. *Graphical Models for Machine Learning and Digital Communication*. MIT Press, 1998.

[Fre99] R. M. French. "Catastrophic forgetting in connectionist networks". In: *Trends in Cognitive Science* (1999).

[Fri09] K. Friston. "The free-energy principle: a rough guide to the brain?" en. In: *Trends Cogn. Sci.* 13.7 (2009), pp. 293–301.

[Fro+13] A. Frome, G. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov. "Devise: A deep visual-semantic embedding model". In: (2013).

[Fro+21] R. Frostig, M. Johnson, D. Maclaurin, A. Paszke, and A. Radul. "Decomposing reverse-mode automatic differentiation". In: *LAFI workshop at POPL 2021*. 2021.

[FS07] S. Fruhwirth-Schnatter. *Finite Mixture and Markov Switching Models*. Springer, 2007.

[FSF10] S. Fruhwirth-Schnatter and R. Fruhwirth. "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.

[FST98] S. Fine, Y. Singer, and N. Tishby. "The Hierarchical Hidden Markov Model: Analysis and Applications". In: *Machine Learning* 32 (1998), p. 41.

[FT05] M. Fashing and C. Tomasi. "Mean shift is a bound optimization". en. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 27.3 (2005), pp. 471–474.

[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].

[FT74] J. H. Friedman and J. W. Tukey. "A Projection Pursuit Algorithm for Exploratory Data Analysis". In: *IEEE Trans. Comput.* C-23.9 (1974), pp. 881–890.

[Fu15] M. Fu, ed. *Handbook of Simulation Optimization*. 1st ed. Springer-Verlag New York, 2015.

[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).

[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*. 2019.

[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.

[Fuj05] S. Fujishige. *Submodular functions and optimization*. Vol. 58. Elsevier Science, 2005.

[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.

[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.

[FV16] V. Feldman and J. Vondrák. "Optimal bounds on approximation of submodular and XOS functions by juntas". In: *SIAM Journal on Computing* 45.3 (2016), pp. 1129–1170.

[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.

[FW12] N. Friel and J. Wyse. "Estimating the evidence – a review". In: *Stat. Neerl.* 66.3 (2012), pp. 288–308.

[FW21] R. Friedman and Y. Weiss. "Posterior Sampling for Image Restoration using Explicit Patch Priors". In: (2021). arXiv: 2104.09895 [cs.CV].

[FWW21] M. Finzi, M. Welling, and A. G. Wilson. "A Practical Method for Constructing Equivariant Multilayer Perceptrons for Arbitrary Matrix Groups". In: *ICML*. 2021.

[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.

[Gan07] M Ganapathiraju. "Application of language technologies in biology: Feature extraction and modeling for transmembrane helix prediction". en. PhD thesis. 2007.

[Gan+16a] Y Ganin, E Ustinova, H Ajakan, P Germain, and others. "Domain-adversarial training of neural networks". In: *JMLR* (2016).

[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.

[Gan+23] A. Gane et al. "ProtNLM: Model-based Natural Language Protein Annotation". In: (2023).

[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).

[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.

[Gao+21] R. Gao, Y. Song, B. Poole, Y. N. Wu, and D. P. Kingma. "Learning Energy-Based Models by Diffusion Recovery Likelihood". In: *ICLR*. 2021.

[Gär03] T. Gärtner. "A Survey of Kernels for Structured Data". In: *SIGKDD Explor. Newsl.* 5.1 (2003), pp. 49–58.

[GAR16] R. B. Grosse, S. Ancha, and D. M. Roy. "Measuring the reliability of MCMC inference with bidirectional Monte Carlo". In: *NIPS*. 2016.

[Gar+18a] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson. "GPyTorch: 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.

[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.

[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*. 2018.

[Gar+18d] M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. M. A. Eslami. "Conditional Neural Processes". In: *ICML*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1704–1713.

[Gar+18e] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. Ali Eslami, and Y. W. Teh. "Neural Processes". In: *ICML workshop on Theoretical Foundations and Applications of Deep Generative Models*. 2018.

[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.

[Gar+20] S. Garg, Y. Wu, S. Balakrishnan, and Z. C. Lipton. "A unified view of label shift estimation". In: *NIPS*. Mar. 2020.

[Gar23] R. Garnett. *Bayesian Optimization*. Cambridge University Press, 2023.

[GARD18] G. Gur-Ari, D. A. Roberts, and E. Dyer. "Gradient Descent Happens in a Tiny Subspace". In: (2018). arXiv: 1812.04754 [cs.LG].

[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.

[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.

[GB00] Z. Ghahramani and M. Beal. "Variational inference for Bayesian mixtures of factor analysers". In: *NIPS-12*. 2000.

[GB09] A. Guillory and J. Bilmes. "Label Selection on Graphs". In: *NIPS*. Vancouver, Canada, 2009.

[GB10] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *AISTATS*. 2010, pp. 249–256.

[GB11] A. Guillory and J. Bilmes. "Active Semi-Supervised Learning using Submodular Functions". In: *UAI*. Barcelona, Spain: AUAI, 2011.

[GB+18] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, 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 (2018), pp. 268–276.

[GBB11] X. Glorot, A. Bordes, and Y. Bengio. "Deep Sparse Rectifer Neural Networks". In: *AISTATS*. 2011.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[GBGM23] R. Gozalo-Brizuela and E. C. Garrido-Merchan. "ChatGPT is not all you need. A State of the Art Review of large Generative AI models". In: (Jan. 2023). arXiv: 2301.04655 [cs.LG].

[GBJ18] R. Giordano, T. Broderick, and M. I. Jordan. "Covariances, Robustness, and Variational Bayes". In: *JMLR* 19.51 (2018), pp. 1–49.

[GBP18] C. Gurau, A. Bewley, and I. Posner. "Dropout Distillation for Efficiently Estimating Model Confidence". 2018.

[GC11] M. Girolami and B. Calderhead. "Riemann manifold Langevin and Hamiltonian Monte Carlo methods". In: *J. of Royal Stat. Soc. Series B* 73.2 (2011), pp. 123–214.

[GC90] R. P. Goldman and E. Charniak. "Dynamic Construction of Belief Networks". In: *UAI*. 1990.

[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.

[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.

[GDFY16] S. Ghosh, F. M. Delle Fave, and J. Yedidia. "Assumed Density Filtering Methods for Learning Bayesian Neural Networks". In: *AAAI*. 2016.

[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.

[Ged+20] D. Gedon, N. Wahlström, T. B. Schön, and L. Ljung. "Deep State Space Models for Nonlinear System Identification". In: (Mar. 2020). arXiv: 2003.14162 [eess.SY].

[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: *Nature Machine Intelligence* (2020).

[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.

[Gel+04] A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian data analysis*. 2nd edition. Chapman and Hall, 2004.

[Gel06] A. Gelman. "Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper)". en. In: *Bayesian Anal.* 1.3 (2006), pp. 515–534.

[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 (2008), pp. 1360–1383.

[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.

[Gel+14b] A. Gelman, A. Vehtari, P. Jylänki, C. Robert, N. Chopin, and J. P. Cunningham. "Expectation propagation as a way of life". In: (2014). arXiv: 1412.4869 [stat.CO].

[Gel16] A. Gelman. "The problems with p-values are not just with p-values". In: *American Statistician* (2016).

[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: (2020). arXiv: 2011.01808 [stat.ME].

[Gel+22] A. Gelman, J. Hill, B. Goodrich, J. Gabry, D. Simpson, and A. Vehtari. *Applied Regression and Multilevel Models*. To appear. 2022.

[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.

[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.

[Geo+17] D. George et al. "A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs". In: *Science* 358.6368 (2017), eaag2612. eprint: https://www.science.org/doi/pdf/10.1126/science.aag2612.

[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.

[Geo88] H.-O. Georgii. *Gibbs Measures and Phase Transitions*. en. Walter De Gruyter Inc, 1988.

[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. PMLR, 2015, pp. 881–889.

[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.

[Ger19] S. J. Gershman. "What does the free energy principle tell us about the brain?" In: *Neurons, Behavior, Data Analysis, and Theory* (2019).

[GEY19] Y. Geifman and R. El-Yaniv. "SelectiveNet: A Deep Neural Network with an Integrated Reject Option". In: *ICML*. 2019.

[Gey92] C. Geyer. "Practical Markov chain Monte Carlo". In: *Statistical Science* 7 (1992), pp. 473–483.

[GF00] E. George and D. Foster. "Calibration and empirical Bayes variable selection". In: *Biometrika* 87.4 (2000), pp. 731–747.

[GF09] I. E. Givoni and B. J. Frey. "A Binary Variable Model for Affinity Propagation". In: *Neural Computation* 21.6 (2009), pp. 1589–1600.

[GF+15] Á. F. García-Fernández, L. Svensson, M. R. Morelande, and S. Särkkä. "Posterior Linearization Filter: Principles and Implementation Using Sigma Points". In: *IEEE Trans. Signal Process.* 63.20 (Oct. 2015), pp. 5561–5573.

[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.

[GF21] A. Grim and P. Felzenszwalb. "Convex Combination Belief Propagation Algorithms". In: (May 2021). arXiv: 2105.12815 [cs.AI].

[GFSS17] Á. F. García-Fernández, L. Svensson, and S. Särkkä. "Iterated Posterior Linearization Smoother". In: *IEEE Trans. Automat. Contr.* 62.4 (2017), pp. 2056–2063.

[GFTS19] Á. F. García-Fernández, F. Tronarp, and S. Särkkä. "Gaussian Process Classification Using Posterior Linearization". In: *IEEE Signal Process. Lett.* 26.5 (2019), pp. 735–739.

[GFWO20] T. Galy-Fajou, F. Wenzel, and M. Opper. "Automated Augmented Conjugate Inference for Non-conjugate Gaussian Process Models". In: *AISTATS*. 2020.

[GG14] S. J. Gershman and N. D. Goodman. "Amortized Inference in Probabilistic Reasoning". In: *36th Annual Conference of the Cognitive Science Society*. 2014.

[GG16] Y. Gal and Z. Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *ICML*. 2016.

[GG84] S. Geman and D. Geman. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". In: *IEEE PAMI* 6.6 (1984).

[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.

[GGS21] M. Grcic, I. Grubisic, and S. Segvic. "DenseFlow: Official implementation of Densely connected normalizing flows". en. In: *NIPS*. 2021.

[GH07] A. Gelman and J. Hill. *Data analysis using regression and multilevel/ hierarchical models*. Cambridge, 2007.

[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.

[GH12] M. Gutmann and J.-i. Hirayama. "Bregman divergence as general framework to estimate unnormalized statistical models". In: *arXiv:1202.3727* (2012).

[GH96a] Z. Ghahramani and G. Hinton. *Parameter estimation for linear dynamical systems*. Tech. rep. CRG-TR-96-2. Dept. Comp. Sci., Univ. Toronto, 1996.

[GH96b] Z. Ghahramani and G. Hinton. *The EM Algorithm for Mixtures of Factor Analyzers*. Tech. rep. Dept. of Comp. Sci., Uni. Toronto, 1996.

[GH98] Z. Ghahramani and G. Hinton. "Variational learning for switching state-space models". In: *Neural Computation* 12.4 (1998), pp. 963–996.

[Gha+15] M. Ghavamzadeh, S. Mannor, J. Pineau, and A. Tamar. "Bayesian Reinforcement Learning: A Survey". In: *Foundations and Trends in ML* (2015).

[Gha+19] M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer. "Mask-Predict: Parallel Decoding of Conditional Masked Language Models". In: *EMNLP*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6112–6121.

[Gha+21] A. Ghandeharioun, 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].

[GHC20] C. Geng, S.-J. Huang, and S. Chen. "Recent Advances in Open Set Recognition: A Survey". In: *IEEE PAMI* (2020).

[GHC21] S. Gould, R. Hartley, and D. J. Campbell. "Deep Declarative Networks". en. In: *IEEE PAMI* PP (2021).

[GHK17] Y. Gal, J. Hron, and A. Kendall. "Concrete Dropout". In: (2017). arXiv: 1705.07832 [stat.ML].

[Gho+19] A. Ghorbani, J. Wexler, J. Zou, and B. Kim. *Towards Automatic Concept-based Explanations*. 2019. arXiv: 1902.03129 [stat.ML].

[Gho+21] A. Ghosh, A. Honoré, D. Liu, G. E. Henter, and S. Chatterjee. "Normalizing Flow based Hidden Markov Models for Classification of Speech Phones with Explainability". In: (July 2021). arXiv: 2107.00730 [cs.LG].

[GHV14] A. Gelman, J. Hwang, and A. Vehtari. "Understanding predictive information criteria for Bayesian models". In: *Statistics and Computing* 24.6 (2014), pp. 997–1016.

[GHV20a] A. Gelman, J. Hill, and A. Vehtari. *Regression and Other Stories*. en. 1st ed. Cambridge University Press, 2020.

[GHV20b] A. Gelman, J. Hill, and A. Vehtari. *Regression and Other Stories*. Cambridge, 2020.

[Gib97] M. Gibbs. "Bayesian Gaussian Processes for Regression and Classification". PhD thesis. U. Cambridge, 1997.

[GIG17] Y. Gal, R. Islam, and Z. Ghahramani. "Deep bayesian active learning with image data". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1183–1192.

[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).

[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).

[Gil88] J. R. Gilbert. "Some nested dissection order is nearly optimal". In: *Inf. Process. Lett.* 26.6 (1988), pp. 325–328.

[Gir+14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[Gir+15] R. Girshick, F. Iandola, T. Darrell, and J. Malik. "Deformable Part Models are Convolutional Neural Networks". In: *CVPR*. 2015.

[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.

[Git89] J. Gittins. *Multi-armed Bandit Allocation Indices*. Wiley, 1989.

[GJ97] Z. Ghahramani and M. Jordan. "Factorial Hidden Markov Models". In: *Machine Learning* 29 (1997), pp. 245–273.

[GK19] L. Graesser and W. L. Keng. *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*. en. 1 edition. Addison-Wesley Professional, 2019.

[GKS05] C. Guestrin, 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.

[GL10] K. Gregor and Y. LeCun. "Learning fast approximations of sparse coding". In: *ICML*. 2010, pp. 399–406.

[GL97] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[Gla03] P. Glasserman. *Monte Carlo Methods in Financial Engineering*. 1st ed. Stochastic Modelling and Applied Probability. Springer-Verlag New York, 2003.

[Gle02] S. Glennan. "Rethinking mechanistic explanation". In: *Philosophy of science* 69.S3 (2002), S342–S353.

[GLM15] J. Ghosh, Y. Li, and R. Mitra. "On the Use of Cauchy Prior Distributions for Bayesian Logistic Regression". In: (2015). arXiv: 1507.07170 [stat.ME].

[GLP21] I. Gulrajani and D. Lopez-Paz. "In Search of Lost Domain Generalization". In: *ICLR*. 2021.

[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.

[GM12] G. Gordon and J. McNulty. *Matroids: a geometric introduction*. Cambridge University Press, 2012.

[GM15] J. Gorham and L. Mackey. "Measuring sample quality with Stein's method". In: *Advances in Neural Information Processing Systems*. 2015, pp. 226–234.

[GM16] R. Grosse and J. Martens. "A Kronecker-factored approximate Fisher matrix for convolution layers". In: *ICML*. 2016.

[GM17] P. L. Green and S Maskell. "Estimating the parameters of dynamical systems from Big Data using Sequential Monte Carlo samplers". In: *Mech. Syst. Signal Process.* 93 (2017), pp. 379–396.

[GM98] A. Gelman and X.-L. Meng. "Simulating normalizing constants: from importance sampling to bridge sampling to path sampling". In: *Statisical Science* 13 (1998), pp. 163–185.

[GMAR16] Y. Gal, R. T. Mc Allister, and C. E. Rasmussen. "Improving PILCO with Bayesian Neural Network Dynamics Models". In: *ICML workshop on Data-efficient machine learning*. 2016.

[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.

[GNM19] D. Greenberg, M. Nonnenmacher, and J. Macke. "Automatic Posterior Transformation for Likelihood-Free Inference". In: *ICML*. 2019.

[Goe+09] M. X. Goemans, N. J. Harvey, S. Iwata, and V. Mirrokni. "Approximating submodular functions everywhere". In: *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*. SIAM. 2009, pp. 535–544.

[Gol+04] J. Goldberger, S. T. Roweis, G. E. Hinton, and R. Salakhutdinov. "Neighbourhood Components Analysis". In: *NIPS*. 2004.

[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.* (2017), p. 106088.

[Gol17] Y. Goldberg. *Neural Network Methods for Natural Language Processing*. Morgan & Claypool Publishers, 2017.

[Gol+19] Z. Goldfeld, E. van den Berg, K. H. Greenewald, I. Melnyk, N. Nguyen, B. Kingsbury, and Y. Polyanskiy. "Estimating Information Flow in Deep Neural Networks". In: *ICML*. 2019.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. Addison-Wesley Longman Publishing Co., Inc., 1989.

[Gom+17] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse. "The Reversible Residual Network: Backpropagation Without Storing Activations". In: *NIPS*. 2017.

[Gon+11] J. Gonzalez, Y. Low, A. Gretton, and C. Guestrin. "Parallel gibbs sampling: From colored fields to thin junction trees". In: *AISTATS*. 2011, pp. 324–332.

[Gon+14] B. Gong, W.-L. Chao, K. Grauman, and F. Sha. "Diverse sequential subset selection for supervised video summarization". In: *Advances in neural information processing systems* 27 (2014), pp. 2069–2077.

[Gon+20] P. J. Goncalves et al. "Training deep neural density estimators to identify mechanistic models of neural dynamics". In: *Elife* 9 (2020).

[Goo+14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Networks". In: *NIPS*. 2014.

[Goo16] I. Goodfellow. "NIPS 2016 Tutorial: Generative Adversarial Networks". In: *NIPS Tutorial*. 2016.

[Goo85] I. Good. "Weight of evidence: A brief survey". In: *Bayesian statistics* 2 (1985), pp. 249–270.

[Gor+14] A. D. Gordon, T. A. Henzinger, A. V. Nori, and S. K. Rajamani. "Probabilistic programming". In: *Intl. Conf. on Software Engineering*. 2014.

[Gor+19] J. Gordon, J. Bronskill, M. Bauer, S. Nowozin, and R. E. Turner. "Meta-Learning Probabilistic Inference For Prediction". In: *ICLR*. 2019.

[Gor93] N. Gordon. "Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation". In: *IEE Proceedings (F)* 140.2 (1993), pp. 107–113.

[Gor95] G. J. Gordon. "Stable Function Approximation in Dynamic Programming". In: *ICML*. 1995, pp. 261–268.

[Gou+96] C. Gourieroux, M. Gourieroux, A. Monfort, and D. A. Monfort. *Simulation-based econometric methods*. Oxford university press, 1996.

[Goy+19] Y. Goyal, Z. Wu, J. Ernst, D. Batra, D. Parikh, and S. Lee. *Counterfactual Visual Explanations*. 2019. arXiv: 1904.07451 [cs.LG].

[Goy+22] S. Goyal, M. Sun, A. Raghunathan, and Z. Kolter. "Test-Time Adaptation via Conjugate Pseudo-labels". In: (July 2022). arXiv: 2207.09640 [cs.LG].

[GPS89] D. Greig, B. Porteous, and A. Seheult. "Exact maximum a posteriori estimation for binary images". In: *J. of Royal Stat. Soc. Series B* 51.2 (1989), pp. 271–279.

[GR06a] M Girolami and S Rogers. "Variational Bayesian Multinomial Probit Regression with Gaussian Process Priors". In: *Neural Comput.* 18.8 (2006), pp. 1790–1817.

[GR06b] M. Girolami and S. Rogers. "Variational Bayesian multinomial probit regression with Gaussian process priors". In: *Neural Comptuation* 18.8 (2006), pp. 1790 –1817.

[GR07] T. Gneiting and A. E. Raftery. "Strictly Proper Scoring Rules, Prediction, and Estimation". In: *JASA* 102.477 (2007), pp. 359–378.

[Gra+10] T. Graepel, J. Quinonero-Candela, T. Borchert, and R. Herbrich. "Web-Scale Bayesian Click-Through Rate Prediction for Sponsored Search Advertising in Microsoft's Bing Search Engine". In: *ICML*. 2010.

[Gra11] A. Graves. "Practical Variational Inference for Neural Networks". In: *NIPS*. 2011.

[Gra+19] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. "FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models". In: (2019).

[Gra+20a] W. Grathwohl, J. Kelly, M. Hashemi, M. Norouzi, K. Swersky, and D. Duvenaud. "No MCMC for me: Amortized sampling for fast and stable training of energy-based models". In: *arXiv preprint arXiv:2010.04230* (2020).

[Gra+20b] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky. "Your classifier is secretly an energy based model and you should treat it like one". In: *ICLR*. 2020.

[Gra+20c] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, and R. Zemel. "Cutting out the Middle-Man: Training and Evaluating Energy-Based Models without Sampling". In: *arXiv preprint arXiv:2002.05616* (2020).

[Gre03] P. Green. "Tutorial on trans-dimensional MCMC". In: *Highly Structured Stochastic Systems*. Ed. by P. Green, N. Hjort, and S. Richardson. OUP, 2003.

[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.

[Gre+14] K. Gregor, I. Danihelka, A. Mnih, C. Blundell, and D. Wierstra. "Deep AutoRegressive Networks". In: *ICML*. 2014.

[Gre20] F. Greenlee. *Transformer VAE*. 2020.

[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).

[Gre95] P. Green. "Reversible Jump Markov Chain Monte Carlo computation and Bayesian model determination". In: *Biometrika* 82 (1995), pp. 711–732.

[Gri+04] T. Griffiths, M. Steyvers, D. Blei, and J. Tenenbaum. "Integrating Topics and Syntax". In: *NIPS*. 2004.

[Gri+20] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, et al. "Bootstrap your own latent: A new approach to self-supervised learning". In: *arXiv preprint arXiv:2006.07733* (2020).

[GRS96] W. Gilks, S. Richardson, and D. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, 1996.

[GS04] T. Griffiths and M. Steyvers. "Finding scientific topics". In: *PNAS* 101 (2004), pp. 5228–5235.

[GS08] Y Guo and D Schuurmans. "Efficient global optimization for exponential family PCA and low-rank matrix factorization". In: *2008 46th Annual Allerton Conference on Communication, Control, and Computing*. 2008, pp. 1100–1107.

[GS13] A. Gelman and C. R. Shalizi. "Philosophy and the practice of Bayesian statistics". en. In: *Br. J. Math. Stat. Psychol.* 66.1 (Feb. 2013), pp. 8–38.

[GS15] R. B. Grosse and R. Salakhutdinov. "Scaling Up Natural Gradient by Sparsely Factorizing the Inverse Fisher Matrix". In: *ICML*. 2015.

[GS90] A. Gelfand and A. Smith. "Sampling-based approaches to calculating marginal densities". In: *JASA* 85 (1990), pp. 385–409.

[GS92] G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford, 1992.

[GS97] C. M. Grinstead and J. L. Snell. *Introduction to probability (2nd edition)*. American Mathematical Society, 1997.

[GSA14] M. A. Gelbart, J. Snoek, and R. P. Adams. "Bayesian Optimization with Unknown Constraints". In: *UAI*. 2014.

[GSD13] A. Guez, D. Silver, and P. Dayan. "Scalable and Efficient Bayes-Adaptive Reinforcement Learning Based on Monte-Carlo Tree Search". In: *JAIR* 48 (2013), pp. 841–883.

[GSJ19] A. Gretton, D. Sutherland, and W. Jitkrittum. *NIPS tutorial on interpretable comparison of distributions and models*. 2019.

[GSK18] S. Gidaris, P. Singh, and N. Komodakis. "Unsupervised Representation Learning by Predicting Image Rotations". In: *International Conference on Learning Representations*. 2018.

[GSM18] U. Garciarena, R. Santana, and A. Mendiburu. "Expanding Variational Autoencoders for Learning and Exploiting Latent Representations in Search Distributions". In: *Proc. of the Conf. on Genetic and Evolutionary Computation*. 2018, pp. 849–856.

[GSR12] S. Garg, A. Singh, and F. Ramos. "Learning non-stationary Space-Time models for environmental monitoring". In: *AAAI* 26.1 (2012), pp. 288–294.

[GSR13] J. Gama, R. Sebastião, and P. P. Rodrigues. "On evaluating stream learning algorithms". In: *MLJ* 90.3 (Mar. 2013), pp. 317–346.

[GSS15] I. J. Goodfellow, J. Shlens, and C. Szegedy. "Explaining and Harnessing Adversarial Examples". In: *ICLR*. 2015.

[GSZ21] P. Grünwald, T. Steinke, and L. Zakynthinou. "PAC-Bayes, MAC-Bayes and Conditional Mutual Information: Fast rate bounds that handle general VC classes". In: *COLT*. 2021.

[GT86] J. R. Gilbert and R. E. Tarjan. "The analysis of a nested dissection algorithm". In: *Numer. Math.* 50.4 (1986), pp. 377–404.

[GU16] C. A. Gomez-Uribe. "Online Algorithms For Parameter Mean And Variance Estimation In Dynamic Regression Models". In: (May 2016). arXiv: 1605.05697 [stat.ML].

[Gue19] B. Guedj. "A primer on PAC-Bayesian learning". In: *arXiv preprint arXiv:1901.05353* (2019).

[GUK21] C. A. Gomez-Uribe and B. Karrer. "The Decoupled Extended Kalman Filter for Dynamic Exponential-Family Factorization Models". In: *JMLR* 22.5 (2021), pp. 1–25.

[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.

[Gul+20] C. Gulcehre et al. *RL Unplugged: Benchmarks for Offline Reinforcement Learning*. arXiv:2006.13888. 2020.

[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.

[Guo09] Y. Guo. "Supervised exponential family principal component analysis via convex optimization". In: *NIPS*. 2009.

[Guo+14] Q. Guo, D. Tu, J. Lei, and G. Li. "Hybrid CNN-HMM model for street view house number recognition". In: *ACCV Workshops*. Lecture notes in computer science. Cham: Springer International Publishing, 2014, pp. 303–315.

[Guo+17] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. "On Calibration of Modern Neural Networks". In: *ICML*. 2017.

[Gup+16] M. Gupta, A. Cotter, J. Pfeifer, K. Voevodski, K. Canini, A. Mangylov, W. Moczydlowski, and A. van Esbroeck. "Monotonic Calibrated Interpolated Look-Up Tables". In: *Journal of Machine Learning Research* 17.109 (2016), pp. 1–47.

[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.

[Gus01] M. Gustafsson. "A probabilistic derivation of the partial least-squares algorithm". In: *Journal of Chemical Information and Modeling* 41 (2001), pp. 288–294.

[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).

[Gut22] M. U. Gutmann. "Pen and Paper Exercises in Machine Learning". In: (June 2022). arXiv: 2206.13446 [cs.LG].

[GV17] S. Ghosal and A. van der Vaart. *Fundamentals of Nonparametric Bayesian Inference*. en. 1st ed. Cambridge University Press, 2017.

[GW08] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Second. Society for Industrial and Applied Mathematics, 2008.

[GW92] W. Gilks and P. Wild. "Adaptive rejection sampling for Gibbs sampling". In: *Applied Statistics* 41 (1992), pp. 337–348.

[GXG18] H. Ge, K. Xu, and Z. Ghahramani. "Turing: a language for flexible probabilistic inference". In: *AISTATS*. 2018, pp. 1682–1690.

[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.

[HA21] R. J. Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice*. en. 3rd ed. Otexts, 2021.

[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.

[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.

[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.

[Haa+18c] T. Haarnoja et al. "Soft Actor-Critic Algorithms and Applications". In: (2018). arXiv: 1812.05905 [cs.LG].

[HAB17] J. H. Huggins, R. P. Adams, and T. Broderick. "PASS-GLM: polynomial approximate sufficient statistics for scalable Bayesian GLM inference". In: *NIPS*. 2017.

[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 (2020), pp. 1028–1040.

[HAE16] M. Huh, P. Agrawal, and A. A. Efros. "What makes ImageNet good for transfer learning?" In: *arXiv preprint arXiv:1608.08614* (2016).

[Haf18] D. Hafner. *Building Variational Auto-Encoders in TensorFlow*. Blog post. 2018.

[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.

[Haf+20] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. "Dream to Control: Learning Behaviors by Latent Imagination". In: *ICLR*. 2020.

[Hag+17] M. Hagen, M. Potthast, M. Gohsen, 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. ACM, 2017, pp. 1261–1264.

[Haj88] B. Hajek. "Cooling Schedules for Optimal Annealing". In: *Math. Oper. Res.* 13.2 (1988), pp. 311–329.

[Häl+21] H. Hälvä, S. L. Corff, L. Leh'ericy, J. So, Y. Zhu, E. Gassiat, and A. Hyvärinen. "Disentangling Identifiable Features from Noisy Data with Structured Nonlinear ICA". In: *NIPS*. 2021.

[Ham90] J. Hamilton. "Analysis of time series subject to changes in regime". In: *J. Econometrics* 45 (1990), pp. 39–70.

[Han16] N. Hansen. "The CMA Evolution Strategy: A Tutorial". In: (2016). arXiv: 1604.00772 [cs.LG].

[Han+20] K. Han et al. "A Survey on Vision Transformer". In: (2020). arXiv: 2012.12556 [cs.CV].

[Han80] T. S. Han. "Multiple mutual informations and multiple interactions in frequency data". In: *Information and Control* 46.1 (1980), pp. 26–45.

[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.

[Har18] K. Hartnett. "To Build Truly Intelligent Machines, Teach Them Cause and Effect". In: *Quanta Magazine* (2018).

[Har90] A. C. Harvey. *Forecasting, Structural Time Series Models, and the Kalman Filter*. Cambridge Univerity Press, 1990.

[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.

[Has70] W. Hastings. "Monte Carlo Sampling Methods Using Markov Chains and Their Applications". In: *Biometrika* 57.1 (1970), pp. 97–109.

[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.

[Hau+11] M. Hauschild, M. Pelikan, M. Hauschild, and M. Pelikan. "An introduction and survey of estimation of distribution algorithms". In: *Swarm and Evolutionary Computation*. 2011.

[Haz22] E. Hazan. *Introduction to Online Convex Optimization*. en. 2nd ed. The MIT Press, Sept. 2022.

[HBB10] M. Hoffman, D. Blei, and F. Bach. "Online learning for latent Dirichlet allocation". In: *NIPS*. 2010.

[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.

[HCL06] R. Hadsell, S. Chopra, and Y. LeCun. "Dimensionality Reduction by Learning an Invariant Mapping". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* 2 (2006), pp. 1735–1742.

[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. Spokoyny, G. Neubig, and T. Berg-Kirkpatrick. "Lagging Inference Networks and Posterior Collapse in Variational Autoencoders". In: *ICLR*. 2019.

[He+20] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. "Momentum contrast for unsupervised visual representation learning". In: *CVPR*. 2020, pp. 9729–9738.

[He+21] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. *Masked Autoencoders Are Scalable Vision Learners*. 2021. arXiv: 2111.06377 [cs.CV].

[He+22] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. "Masked autoencoders are scalable vision learners". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16000–16009.

[Heg06] P. Heggernes. "Minimal triangulations of graphs: A survey". In: *Discrete Math.* 306.3 (2006), pp. 297–317.

[Hei+16] M. Heinonen, H. Mannerström, J. Rousu, S. Kaski, and H. Lähdesmäki. "Non-Stationary Gaussian Process Regression with Hamiltonian Monte Carlo". In: *AISTATS*. Ed. by A. Gretton and C. C. Robert. Vol. 51. Proceedings of Machine Learning Research. Cadiz, Spain: PMLR, 2016, pp. 732–740.

[Hel17] J. Helske. "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: (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]* (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: (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*. 2021.

[Hes00] T. Heskes. "On "Natural" Learning and Pruning in Multilayered Perceptrons". In: *Neural Comput.* 12.4 (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.

[HFL13] J. Hensman, N. Fusi, and N. D. Lawrence. "Gaussian Processes for Big Data". In: *UAI*. 2013.

[HFM17] D. W. Hogg and D. Foreman-Mackey. "Data analysis recipes: Using Markov Chain Monte Carlo". In: (2017). arXiv: 1710.06068 [astro-ph.IM].

[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.

[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.

[HG16] D. Hendrycks and K. Gimpel. "Gaussian Error Linear Units (GELUs)". In: *arXiv [cs.LG]* (2016).

[HGMG18] J. Hron, A. G. de G. Matthews, and Z. Ghahramani. "Variational Bayesian dropout: pitfalls and fixes". In: *ICML*. 2018.

[HGS16] H. van Hasselt, A. Guez, and D. Silver. "Deep Reinforcement Learning with Double Q-Learning". In: *AAAI*. AAAI'16. AAAI Press, 2016, pp. 2094–2100.

[HH06] C. Holmes and L. Held. "Bayesian auxiliary variable models for binary and multinomial regression". In: *Bayesian Analysis* 1.1 (2006), pp. 145–168.

[HHH09] A. Hyvarinen, J. Hurri, and P. Hoyer. *Natural Image Statistics: a probabilistic approach to early computational vision*. Springer, 2009.

[HHK19] M. Haußmann, F. A. Hamprecht, and M. Kandemir. "Sampling-Free Variational Inference of Bayesian Neural Networks by Variance Backpropagation". In: *UAI*. 2019.

[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.

[Hig+17a] 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.

[Hig+17b] I. Higgins, L. Matthey, A. Pal, C. P. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *ICLR*. 2017.

[Hin02] G. E. Hinton. "Training products of experts by minimizing contrastive divergence". en. In: *Neural Computation* 14.8 (2002), pp. 1771–1800.

[Hin10] G. Hinton. *A Practical Guide to Training Restricted Boltzmann Machines*. Tech. rep. U. Toronto, 2010.

[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 (1995), pp. 1158–1161.

[HIY19] K. Hayashi, M. Imaizumi, and Y. Yoshida. "On Random Subsampling of Gaussian Process Regression: A Graphon-Based Analysis". In: (2019). arXiv: 1901.09541 [stat.ML].

[HJ12] T. Hazan and T. Jaakkola. "On the Partition Function and Random Maximum A-Posteriori Perturbations". In: *ICML*. June 2012.

[HJ20] J. Huang and N. Jiang. "From Importance Sampling to Doubly Robust Policy Gradient". In: *ICML*. 2020.

[HJA20] J. Ho, A. Jain, and P. Abbeel. "Denoising Diffusion Probabilistic Models". In: *NIPS*. 2020.

[Hje+18] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. "Learning deep representations by mutual information estimation and maximization". In: *International Conference on Learning Representations*. 2018.

[Hjo+10] N. Hjort, C. Holmes, P. Muller, and S. Walker, eds. *Bayesian Nonparametrics*. Cambridge, 2010.

[HJT18] M. D. Hoffman, M. J. Johnson, and D. Tran. "Autoconj: Recognizing and Exploiting Conjugacy Without a Domain-Specific Language". In: *NIPS*. 2018.

[HKH22] M. Hobbhahn, A. Kristiadi, and P. Hennig. "Fast Predictive Uncertainty for Classification with Bayesian Deep Networks". In: *UAI*. 2022.

[HKO22] P. Hennig, H. Kersting, and M. Osborne. *Probabilistic Numerics: Computation as Machine Learning*. 2022.

[HKP91] J. Hertz, A. Krogh, and R. G. Palmer. *An Introduction to the Theory of Neural Comptuation*. Addison-Wesley, 1991.

[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.

[HL04] D. R. Hunter and K. Lange. "A Tutorial on MM Algorithms". In: *The American Statistician* 58 (2004), pp. 30–37.

[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*. 2016, pp. 1511–1520.

[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).

[HL20] X. Hu and J. Lei. "A Distribution-Free Test of Covariate Shift Using Conformal Prediction". In: (2020). arXiv: 2010.07147 [stat.ME].

[HLA15a] J. M. Hernández-Lobato and R. P. Adams. "Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks". In: *ICML*. 2015.

[HLA15b] J. M. Hernández-Lobato and R. P. Adams. "Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks". In: *ICML*. 2015.

[HLA19] J. Ho, E. Lohn, and P. Abbeel. "Compression with Flows via Local Bits-Back Coding". In: *NeurIPS*. 2019.

[HLC19] Y.-P. Hsieh, C. Liu, and V. Cevher. "Finding mixed nash equilibria of generative adversarial networks". In: *International Conference on Machine Learning*. 2019, pp. 2810–2819.

[HLHG14] J. Hernandez-Lobato, M. W. Hoffman, and Z. Ghahramani. "Predictive entropy search for efficient global optimization of black-box functions". In: *NIPS*. 2014.

[HLR16] K. Hofmann, L. Li, and F. Radlinski. "Online Evaluation for Information Retrieval". In: *Foun-*

dations and Trends in Information Retrieval 10.1 (2016), pp. 1–117.

[HLRW14] J. R. Hershey, J. Le Roux, and F. Weninger. "Deep Unfolding: Model-Based Inspiration of Novel Deep Architectures". In: (2014). arXiv: 1409.2574 [cs.LG].

[HLS03] R. Herbrich, N. D. Lawrence, and M. Seeger. "Fast Sparse Gaussian Process Methods: The Informative Vector Machine". In: *NIPS*. MIT Press, 2003, pp. 625–632.

[HM81] R. Howard and J. Matheson. "Influence diagrams". In: *Readings on the Principles and Applications of Decision Analysis, volume II*. Ed. by R. Howard and J. Matheson. Strategic Decisions Group, 1981.

[HMD18] J. C. Higuera, D Meger, and G Dudek. "Synthesizing Neural Network Controllers with Probabilistic Model-Based Reinforcement Learning". In: *IROS*. 2018, pp. 2538–2544.

[HMD19] D. Hendrycks, M. Mazeika, and T. Dietterich. "Deep Anomaly Detection with Outlier Exposure". In: *ICLR*. 2019.

[HMK04] D. Heckerman, C. Meek, and D. Koller. *Probabilistic Models for Relational Data*. Tech. rep. MSR-TR-2004-30. Microsoft Research, 2004.

[HNBK18] J. He, G. Neubig, and T. Berg-Kirkpatrick. "Unsupervised Learning of Syntactic Structure with Invertible Neural Projections". In: *EMNLP*. 2018.

[HNP09] A. Halevy, P. Norvig, and F. Pereira. "The unreasonable effectiveness of data". In: *IEEE Intelligent Systems* 24.2 (2009), pp. 8–12.

[HO00] A. Hyvarinen and E. Oja. "Independent component analysis: algorithms and applications". In: *Neural Networks* 13 (2000), pp. 411–430.

[Ho+21] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans. "Cascaded Diffusion Models for High Fidelity Image Generation". In: (May 2021). arXiv: 2106.15282 [cs.CV].

[HO48] C. G. Hempel and P. Oppenheim. "Studies in the Logic of Explanation". In: *Philosophy of science* 15.2 (1948), pp. 135–175.

[Hob69] A. Hobson. "A new theorem of information theory". In: *Journal of Statistical Physics* 1.3 (1969), pp. 383–391.

[Hoe+14] R. Hoekstra, R. D. Morey, J. N. Rouder, and E.-J. Wagenmakers. "Robust misinterpretation of confidence intervals". en. In: *Psychon. Bull. Rev.* 21.5 (2014), pp. 1157–1164.

[Hoe+21] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste. "Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks". In: (2021). arXiv: 2102.00554 [cs.LG].

[Hoe+99] J. Hoeting, D. Madigan, A. Raftery, and C. Volinsky. "Bayesian Model Averaging: A Tutorial". In: *Statistical Science* 4.4 (1999).

[Hof09] P. D. Hoff. *A First Course in Bayesian Statistical Methods*. Springer, 2009.

[Hof+13] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. "Stochastic Variational Inference". In: *JMLR* 14 (2013), pp. 1303–1347.

[Hof17] M. D. Hoffman. "Learning Deep Latent Gaussian Models with Markov Chain Monte Carlo". In: *ICML*. 2017, pp. 1510–1519.

[Hof+18] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell. "Cycada: Cycle-consistent adversarial domain adaptation". In: *International conference on machine learning*. PMLR. 2018, pp. 1989–1998.

[Hof+19] M. Hoffman, P. Sountsov, J. V. Dillon, I. Langmore, D. Tran, and S. Vasudevan. "NeuTra-lizing Bad Geometry in Hamiltonian Monte Carlo Using Neural Transport". In: (2019). arXiv: 1903.03704 [stat.CO].

[Hof99] T. Hofmann. "Probabilistic latent semantic indexing". In: *Research and Development in Information Retrieval* (1999), pp. 50–57.

[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.

[Hol92] J. H. Holland. *Adaptation in Natural and Artificial Systems*. https://mitpress.mit.edu/books/adaptation-natural-and-artificial-systems. Accessed: 2017-11-26. 1992.

[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.

[Hoo+21] E. Hoogeboom, D. Nielsen, P. Jaini, P. Forré, and M. Welling. "Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions". In: *NIPS*. Feb. 2021.

[Hop82] J. J. Hopfield. "Neural networks and physical systems with emergent collective computational abilities". In: *PNAS* 79.8 (1982), 2554–2558.

[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 (1961), pp. 331–347.

[Hos+20a] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. "Meta-Learning in Neural Networks: A Survey". In: (2020). arXiv: 2004.05439 [cs.LG].

[Hos+20b] R. Hostettler, F. Tronarp, Á. F. García-Fernández, and S. Särkkä. "Importance Densities for Particle Filtering Using Iterated Conditional Expectations". In: *IEEE Signal Process. Lett.* 27 (2020), pp. 211–215.

[HOT06a] G. Hinton, S. Osindero, and Y. Teh. "A fast learning algorithm for deep belief nets". In: *Neural Computation* 18 (2006), pp. 1527–1554.

[HOT06b] G. E. Hinton, S. Osindero, and Y. W. 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.

[Hou+12] N. Houlsby, F. Huszar, Z. Ghahramani, and J. M. Hernández-lobato. "Collaborative Gaussian Processes for Preference Learning". In: *NIPS*. 2012, pp. 2096–2104.

[Hou+19] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. "Parameter-efficient transfer learning for NLP". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2790–2799.

[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.

[HPHL19] M. Havasi, R. Peharz, and J. M. Hernández-Lobato. "Minimal Random Code Learning: Getting Bits Back from Compressed Model Parameters". In: 2019.

[HPR19] C. Herzog né Hoffmann, E. Petersen, and P. Rostalski. "Iterative Approximate Nonlinear Inference via Gaussian Message Passing on Factor Graphs". In: *IEEE Control Systems Letters* 3.4 (2019), pp. 978–983.

[HR17] C. Hoffmann and P. Rostalski. "Linear Optimal Control on Factor Graphs — A Message Passing Perspective". In: *Intl. Federation of Automatic Control* 50.1 (2017), pp. 6314–6319.

[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.

[HS05] H. Hoos and T. Stutzle. *Stochastic local search: Foundations and applications*. Morgan Kauffman, 2005.

[HS06a] G. Hinton and R. Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *Science* 313.5786 (2006), pp. 504–507.

[HS06b] G. E. Hinton and R. 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.

[HS10] J. Hartikainen and S. Särkkä. "Kalman filtering and smoothing solutions to temporal Gaussian process regression models". In: *2010 IEEE International Workshop on Machine Learning for Signal Processing*. Aug. 2010, pp. 379–384.

[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.

[HS21] J. Ho and T. Salimans. "Classifier-Free Diffusion Guidance". In: *NIPS Workshop on Deep Generative Models and Downstream Applications*. 2021.

[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.

[HSDK12] C. Hillar, J. Sohl-Dickstein, and K. Koepsell. *Efficient and Optimal Binary Hopfield Associative Memory Storage Using Minimum Probability Flow*. Tech. rep. 2012. arXiv: 1204.2916.

[HSG06] J. D. Hol, T. B. Schon, and F. Gustafsson. "On Resampling Algorithms for Particle Filters". In: *IEEE Nonlinear Statistical Signal Processing Workshop*. 2006, pp. 79–82.

[HSGF21] S. Hassan, S. Sarkka, and A. F. Garcia-Fernandez. "Temporal Parallelization of Inference in Hidden Markov Models". In: *IEEE Trans. Signal Processing* 69 (2021), pp. 4875–4887.

[Hsu+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*. 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*. 2015.

[HT17] T. J. Hastie and R. J. Tibshirani. *Generalized additive models*. Routledge, 2017.

[HTF01] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[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: (2017). arXiv: 1705.07562 [stat.ML].

[Hua+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.

[Hua+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 (2017), pp. 1545–1554.

[Hua+18a] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck. "Music Transformer". In: (2018). arXiv: 1809.04281 [cs.LG].

[Hua+18b] C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville. "Neural Autoregressive Flows". In: *ICML*. 2018.

[Hua+18c] Y. Huang, Y. Zhang, Z. Wu, N. Li, and J. Chambers. "A Novel Adaptive Kalman Filter With Inaccurate Process and Measurement Noise Covariance Matrices". In: *IEEE Trans. Automat. Contr.* 63.2 (Feb. 2018), pp. 594–601.

[Hua+19] 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 (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. Huggins, M. Kasprzak, T. Campbell, and T. Broderick. "Validated Variational Inference via Practical Posterior Error Bounds". In: *AISTATS*. Ed. by S. Chiappa and R. Calandra. 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.

[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 (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).

[Hyv05a] A. Hyvärinen. "Estimation of Non-Normalized Statistical Models by Score Matching". In: *JMLR* 6.Apr (2005), pp. 695–709.

[Hyv05b] A. Hyvärinen. "Estimation of non-normalized statistical models by score matching". In: *JMLR* 6.Apr (2005), pp. 695–709.

[Hyv07a] A. Hyvarinen. "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.

[IBK06] E. L. Ionides, C Bretó, and A. A. King. "Inference for nonlinear dynamical systems". en. In: *PNAS* 103.49 (2006), pp. 18438–18443.

[IFF00] S. Iwata, L. Fleischer, and S. Fujishige. "A combinatorial strongly polynomial algorithm for minimizing submodular functions". In: *Journal of the ACM* (2000).

[IFW05] A. T. Ihler, J. W. Fischer III, and A. S. Willsky. "Loopy Belief Propagation: Convergence and Effects of Message Errors". In: *JMLR* 6 (2005), pp. 905–936.

[IJB13] R. Iyer, S. Jegelka, and J. Bilmes. "Curvature and Optimal Algorithms for Learning and Minimizing Submodular Functions". In: *NIPS*. Lake Tahoe, CA, 2013.

[IKB21] A. Immer, M. Korzepa, and M. Bauer. "Improving predictions of Bayesian neural nets via local linearization". In: *AISTATS*. Ed. by A. Banerjee and K. Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, 2021, pp. 703–711.

[IM17] J. Ingraham and D. Marks. "Bayesian Sparsity for Intractable Undirected Models". In: *ICML*. 2017.

[Imb03] G. Imbens. "Sensitivity to Exogeneity Assumptions in Program Evaluation". In: *The American Economic Review* (2003).

[Imb19] G. W. Imbens. "Potential Outcome and Directed Acyclic Graph Approaches to Causality: Relevance for Empirical Practice in Economics". In: (2019). arXiv: 1907.07271 [stat.ME].

[Imm+21] A. Immer, M. Bauer, V. Fortuin, G. Rätsch, and M. E. Khan. "Scalable Marginal Likelihood Estimation for Model Selection in Deep Learning". In: *ICML*. 2021.

[IN09] S. Iwata and K. Nagano. "Submodular function minimization under covering constraints". In: *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2009, pp. 671–680.

[Ing20] M. Ingram. *Deterministic ADVI in JAX (blog post)*. https://martiningram.github.io/deterministic-advi/. 2020.

[INK18] P. Izmailov, A. Novikov, and D. Kropotov. "Scalable Gaussian Processes with Billions of Inducing Inputs via Tensor Train Decomposition". In: *ICML*. 2018.

[Inn20] M. Innes. "Sense & Sensitivities: The Path to General-Purpose Algorithmic Differentiation". In: *Proceedings of Machine Learning and Systems*. Ed. by I. Dhillon, D. Papailiopoulos, and V. Sze. Vol. 2. 2020, pp. 58–69.

[IO09] S. Iwata and J. B. Orlin. "A simple combinatorial algorithm for submodular function minimization". In: *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*. SIAM. 2009, pp. 1230–1237.

[IR00] D. R. Insua and F. Ruggeri. *Robust Bayesian Analysis*. Springer, 2000.

[IR15] G. Imbens and D. Rubin. *Causal Inference in Statistics, Social and Biomedical Sciences: An Introduction*. Cambridge University Press, 2015.

[IS15] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *ICML*. 2015, pp. 448–456.

[Isa03] M. Isard. "PAMPAS: Real-Valued Graphical Models for Computer Vision". In: *CVPR*. Vol. 1. 2003, p. 613.

[Isl+19] R. Islam, R. Seraj, S. Y. Arnob, and D. Precup. "Doubly Robust Off-Policy Actor-Critic Algorithms for Reinforcement Learning". In: *NeurIPS Workshop on Safety and Robustness in Decision Making*. 2019.

[Iso+17] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. "Image-to-Image Translation with Conditional Adversarial Networks". In: *CVPR*. 2017.

[IX00] K Ito and K Xiong. "Gaussian filters for nonlinear filtering problems". In: *IEEE Trans. Automat. Contr.* 45.5 (2000), pp. 910–927.

[Iye+21] R. Iyer, N. Khargonkar, J. Bilmes, and H. Asnani. "Generalized Submodular Information Measures: Theoretical Properties, Examples, Optimization Algorithms, and Applications". In: *IEEE Transactions on Information Theory* (2021).

[Iza+15] H. Izadinia, B. C. Russell, A. Farhadi, M. D. Hoffman, and A. Hertzmann. "Deep classifiers from image tags in the wild". In: *Proceedings of the 2015 Workshop on Community-Organized Multimodal Mining: Opportunities for Novel Solutions*. 2015, pp. 13–18.

[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.

[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.

[Izm+21a] P. Izmailov, P. Nicholson, S. Lotfi, and A. G. Wilson. "Dangers of Bayesian Model Averaging under Covariate Shift". In: *NIPS*. 2021.

[Izm+21b] P. Izmailov, S. Vikram, M. D. Hoffman, and A. G. Wilson. "What Are Bayesian Neural Network Posteriors Really Like?" In: *ICML*. 2021.

[Jaa01] T. Jaakkola. "Tutorial on variational approximation methods". In: *Advanced mean field methods*. Ed. by M. Opper and D. Saad. MIT Press, 2001.

[Jac+21] M. Jacobs, M. F. Pradier, T. H. McCoy, R. H. Perlis, F. Doshi-Velez, and K. Z. Gajos. "How machine-learning recommendations influence clinician treatment selections: the example of antidepressant selection". In: *Translational psychiatry* 11.1 (2021), pp. 1–9.

[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.

[Jak21] K. Jakkala. "Deep Gaussian Processes: A Survey". In: (2021). arXiv: 2106.12135 [cs.LG].

[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.

[Jan18] E. Jang. *Normalizing Flows Tutorial*. https://blog.evjang.com/2018/01/nf1.html. 2018.

[Jan+19] M. Janner, J. Fu, M. Zhang, and S. Levine. "When to Trust Your Model: Model-Based Policy Optimization". In: *NIPS*. 2019.

[Jas] Jason Antic and Jeremy Howard and Uri Manor. *Decrappification, DeOldification, and Super Resolution (Blog post)*.

[Jay03] E. T. Jaynes. *Probability theory: the logic of science*. Cambridge university press, 2003.

[Jay+20] S. M. Jayakumar, W. M. Czarnecki, J. Menick, J. Schwarz, J. Rae, S. Osindero, Y. W. Teh, T. Harley, and R. Pascanu. "Multiplicative Interactions and Where to Find Them". In: *ICLR*. 2020.

[Jay76] E. T. Jaynes. "Confidence intervals vs Bayesian intervals". In: *Foundations of Probability Theory, Statistical Inference, and Statistical Theories of Science, vol II*. Ed. by W. L. Harper and C. A. Hooker. Reidel Publishing Co., 1976.

[JB03] A. Jakulin and I. Bratko. "Analyzing Attribute Dependencies". In: *Proc. 7th European Conf. on Principles and Practice of Knowledge Discovery in Databases*. 2003.

[JB16] S. Jegelka and J. Bilmes. "Graph cuts with interacting edge weights: examples, approximations, and algorithms". In: *Mathematical Programming* (2016), pp. 1–42.

[JB65] C. Jacobi and C. W. Borchardt. "De investigando ordine systematis aequationum differentialium vulgarium cujuscunque." In: *Journal für die reine und angewandte Mathematik* 1865.64 (1865), pp. 297–320.

[JB67] A. H. Jazwinski and A. E. Bailie. *Adaptive Filtering*. Tech. rep. 1967.

[JBB09] D. Jian, A. Barthels, and M. Beetz. "Adaptive Markov logic networks: Learning statistical relational models with dynamic parameters". In: *9th European Conf. on AI*. 2009, 937–942.

[Jef04] R. Jeffrey. *Subjective Probability: The Real Thing*. Cambridge, 2004.

[Jen+17] R. Jenatton, C. Archambeau, J. González, and M. Seeger. "Bayesian Optimization with Tree-structured Dependencies". en. In: *ICML*. 2017, pp. 1655–1664.

[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: (2019). arXiv: 1909.08471 [cs.IR].

[JG20] A. Jacovi and Y. Goldberg. "Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness?" In: *arXiv preprint arXiv:2004.03685* (2020).

[JG21] A. Jacovi and Y. Goldberg. "Aligning faithful interpretations with their social attribution". In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 294–310.

[JGH18] A. Jacot, F. Gabriel, and C. Hongler. "Neural Tangent Kernel: Convergence and Generalization in Neural Networks". In: *NIPS*. 2018.

[JGP17] E. Jang, S. Gu, and B. Poole. "Categorical Reparameterization with Gumbel-Softmax". In: *ICLR*. 2017.

[Jha+22] S. Jha, D. Gong, X. Wang, R. E. Turner, and L. Yao. "The Neural Process Family: Survey, Applications and Perspectives". In: (Sept. 2022). arXiv: 2209.00517 [cs.LG].

[Ji+22] Z. Ji et al. "Survey of Hallucination in Natural Language Generation". In: *ACM Computing Surveys* (Feb. 2022).

[Jia+19] R. Jia, A. Raghunathan, K. Göksel, and P. Liang. "Certified Robustness to Adversarial Word Substitutions". In: *EMNLP*. 2019.

[Jia+21] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. V. Le, Y. Sung, Z. Li, and T. Duerig. *Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision*. 2021. arXiv: 2102.05918 [cs.CV].

[Jia21] H. Jiang. "Minimizing convex functions with integral minimizers". In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2021, pp. 976–985.

[Jih+12] Jihong Min, J Kim, Seunghak Shin, and I. S. Kweon. "Efficient Data-Driven MCMC sampling for vision-based 6D SLAM". In: *ICRA*. 2012, pp. 3025–3032.

[Jin11] Y. Jin. "Surrogate-assisted evolutionary computation: Recent advances and future challenges". In: *Swarm and Evolutionary Computation* 1.2 (2011), pp. 61–70.

[Jit+16] W. Jitkrittum, Z. Szabó, K. P. Chwialkowski, and A. Gretton. "Interpretable Distribution Features with Maximum Testing Power". In: *NIPS*. Curran Associates, Inc., 2016, pp. 181–189.

[JJ00] T. S. Jaakkola and M. I. Jordan. "Bayesian parameter estimation via variational methods". In: *Statistics and Computing* 10 (2000), pp. 25–37.

[JKG18] H. Jiang, B. Kim, and M. Y. Guan. "To Trust Or Not To Trust A Classifier". In: *NIPS*. 2018.

[JKK95] C. S. Jensen, A. Kong, and U. Kjaerulff. "Blocking-Gibbs Sampling in Very Large Probabilistic Expert Systems". In: *Intl. J. Human-Computer Studies* (1995), pp. 647–666.

[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.

[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.

[JL16] N. Jiang and L. Li. "Doubly Robust Off-policy Evaluation for Reinforcement Learning". In: *ICML*. 2016, pp. 652–661.

[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.

[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.

[JM18] A. Jolicoeur-Martineau. "The relativistic discriminator: a key element missing from standard GAN". In: *arXiv preprint arXiv:1807.00734* (2018).

[JM70] D. H. Jacobson and D. Q. Mayne. *Differential Dynamic Programming*. Elsevier Press, 1970.

[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.

[JNJ20] C. Jin, P. Netrapalli, and M. I. Jordan. "What is local optimality in nonconvex-nonconcave minimax optimization?" In: *Proceedings of the 34th International Conference on Machine Learning-Volume 73*. 2020.

[JO18] M. Jankowiak and F. Obermeyer. "Pathwise Derivatives Beyond the Reparameterization Trick". In: *ICML*. 2018.

[JOA10] T. Jaksch, R. Ortner, and P. Auer. "Near-optimal Regret Bounds for Reinforcement Learning". In: *JMLR* 11 (2010), pp. 1563–1600.

[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).

[Joh12] M. J. Johnson. "A Simple Explanation of A Spectral Algorithm for Learning Hidden Markov Models". In: (2012). arXiv: 1204.2477 [stat.ME].

[Jon01] D. R. Jones. "A Taxonomy of Global Optimization Methods Based on Response Surfaces". In: *J. Global Optimiz.* 21.4 (2001), pp. 345–383.

[Jor07] M. I. Jordan. *An Introduction to Probabilistic Graphical Models*. In preparation. 2007.

[Jor+22] J. Jordon, L. Szpruch, F. Houssiau, M. Bottarelli, G. Cherubin, C. Maple, S. N. Cohen, and A. Weller. "Synthetic Data – what, why and how?" In: (May 2022). arXiv: 2205.03257 [cs.LG].

[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.

[Jos+17] A. Joshi, S. Ghosh, M. Betke, S. Sclaroff, and H. Pfister. "Personalizing gesture recognition using hierarchical Bayesian neural networks". In: *CVPR*. Honolulu, HI: IEEE, July 2017.

[Jos20] C. Joshi. *Transformers are Graph Neural Networks*. Tech. rep. 2020.

[Jos+20] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy. "Spanbert: Improving pre-training by representing and predicting spans". In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 64–77.

[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: (2022).

[Jou+16] A. Joulin, L. Van Der Maaten, A. Jabri, and N. Vasilache. "Learning visual features from large weakly supervised data". In: *European Conference on Computer Vision*. Springer. 2016, pp. 67–84.

[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.

[JS93] M. Jerrum and A. Sinclair. "Polynomial-time approximation algorithms for the Ising model". In: *SIAM J. on Computing* 22 (1993), pp. 1087–1116.

[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.

[JSY19] P. Jaini, K. A. Selby, and Y. Yu. "Sum-of-Squares Polynomial Flow". In: *ICML*. 2019, pp. 3009–3018.

[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.

[JUDW00] S Julier, J Uhlmann, and H. F. Durrant-Whyte. "A new method for the nonlinear transformation of means and covariances in filters and estimators". In: *IEEE Trans. Automat. Contr.* 45.3 (Mar. 2000), pp. 477–482.

[JW14] M. Johnson and A. Willsky. "Stochastic Variational Inference for Bayesian Time Series Models". en. In: *ICML*. 2014, pp. 1854–1862.

[JW19] S. Jain and B. C. Wallace. "Attention is not explanation". In: *arXiv preprint arXiv:1902.10186* (2019).

[Kaa12] Kaare Brandt Petersen and Michael Syskind Pedersen. *The Matrix Cookbook*. 2012.

[Kad+22] J. Kaddour, A. Lynch, Q. Liu, M. J. Kusner, and R. Silva. "Causal Machine Learning: A Survey and Open Problems". In: (June 2022). arXiv: 2206.15475 [cs.LG].

[KAG19] A. Kirsch, J. van Amersfoort, and Y. Gal. "BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning". In: *NIPS*. 2019.

[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.

[Kai58] H. Kaiser. "The varimax criterion for analytic rotation in factor analysis". In: *Psychometrika* 23.3 (1958).

[Kak02] S. M. Kakade. "A Natural Policy Gradient". In: *NIPS*. 2002, pp. 1531–1538.

[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.

[Kal+18a] D. Kalashnikov et al. "QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation". In: *CORL*. 2018.

[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.

[Kam16] E. Kamar. "Directions in Hybrid Intelligence: Complementing AI Systems with Human Intelligence." In: *IJCAI*. 2016, pp. 4070–4073.

[Kam+22] S. Kamthe, S. Takao, S. Mohamed, and M. P. Deisenroth. "Iterative state estimation in non-linear dynamical systems using approximate expectation propagation". In: *Trans. on Machine Learning Research* (2022).

[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).

[Kan42] L. Kantorovich. "On the transfer of masses (in Russian)". In: *Doklady Akademii Nauk* 37.2 (1942), pp. 227–229.

[Kap+22] S. Kapoor, W. J. Maddox, P. Izmailov, and A. G. Wilson. "On Uncertainty, Tempering, and Data Augmentation in Bayesian Classification". In: *arXiv preprint arXiv:2203.16481* (2022).

[Kar+18] T. Karras, T. Aila, S. Laine, and J. Lehtinen. "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *ICLR*. 2018.

[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].

[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).

[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.

[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).

[Kar+22] T. Karras, M. Aittala, T. Aila, and S. Laine. "Elucidating the Design Space of Diffusion-Based Generative Models". In: *NIPS*. June 2022.

[Kat05] T. Katayama. *Subspace Methods for Systems Identification*. Springer Verlag, 2005.

[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.

[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.

[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*. Association for Computing Machinery, 2019.

[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.

[Kau+20] H. Kaur, H. Nori, S. Jenkins, R. Caruana, H. Wallach, and J. Wortman Vaughan. "Interpreting interpretability: understanding data scientists' use of interpretability tools for machine learning". In: *Proceedings of the 2020 CHI conference on human factors in computing systems*. 2020, pp. 1–14.

[Kau+21] D. Kaushik, A. Setlur, E. Hovy, and Z. C. Lipton. "Explaining The Efficacy of Counterfactually Augmented Data". In: *ICLR*. 2021.

[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 (2000), pp. 580–585.

[KB14a] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[KB14b] K. Kirchhoff and J. Bilmes. "Submodularity for Data Selection in Machine Translation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014.

[KB16] T. Kim and Y. Bengio. "Deep directed generative models with energy-based probability estimation". In: *arXiv preprint arXiv:1606.03439* (2016).

[KBH19] F. Kunstner, L. Balles, and P. Hennig. "Limitations of the Empirical Fisher Approximation". In: (2019). arXiv: 1905.12558 [cs.LG].

[KCC20] V. Kumar, A. Choudhary, and E. Cho. "Data Augmentation using Pre-trained Transformer Models". In: *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*. Suzhou, China: Association for Computational Linguistics, Dec. 2020, pp. 18–26.

[KD18a] S. Kamthe and M. P. Deisenroth. "Data-Efficient Reinforcement Learning with Probabilistic Model Predictive Control". In: *AISTATS*. 2018.

[KD18b] D. P. Kingma and P. Dhariwal. "Glow: Generative Flow with Invertible 1×1 Convolutions". In: *NIPS*. 2018.

[Ke+19a] L. Ke, M. Barnes, W. Sun, G. Lee, S. Choudhury, and S. Srinivasa. "Imitation Learning as f-Divergence Minimization". In: *arXiv preprint arXiv:1905.12888* (2019).

[Ke+19b] L. Ke, S. Choudhury, M. Barnes, W. Sun, G. Lee, and S. Srinivasa. *Imitation Learning as f-Divergence Minimization*. arXiv:1905.12888. 2019.

[Ke+21] A. Ke, W. Ellsworth, O. Banerjee, A. Y. Ng, and P. Rajpurkar. "CheXtransfer: performance and parameter efficiency of ImageNet models for chest X-Ray interpretation". In: *Proceedings of the Conference on Health, Inference, and Learning*. 2021, pp. 116–124.

[Kei06] F. C. Keil. "Explanation and understanding". In: *Annu. Rev. Psychol.* 57 (2006), pp. 227–254.

[Kel+20] J. Kelly, J. Bettencourt, M. J. Johnson, and D. Duvenaud. "Learning Differential Equations that are Easy to Solve". In: *Neural Information Processing Systems*. 2020.

[Kel21] R. Kelter. "Bayesian model selection in the M-open setting — Approximate posterior inference and subsampling for efficient large-scale leave-one-out cross-validation via the difference estimator". In: *J. Math. Psychol.* 100 (Feb. 2021), p. 102474.

[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.

[Ken17] E. H. Kennedy. *Semiparametric theory*. 2017. arXiv: 1709.06418 [stat.ME].

[Ken20] E. H. Kennedy. *Optimal doubly robust estimation of heterogeneous causal effects*. 2020. arXiv: 2004.14497 [math.ST].

[Ken93] G. Kenji. *100 Statistical Tests*. Sage Publications, 1993.

[Kes+17] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima". In: *ICLR*. 2017.

[KF09a] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[KF09b] D. Krishnan and R. Fergus. "Fast Image Deconvolution using Hyper-Laplacian Priors". In: *NIPS*. 2009, pp. 1033–1041.

[KFL01] F. Kschischang, B. Frey, and H.-A. Loeliger. "Factor Graphs and the Sum-Product Algorithm". In: *IEEE Trans Info. Theory* (2001).

[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.

[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.

[KGO12] H. J. Kappen, V. Gómez, and M. Opper. "Optimal control as a graphical model inference problem". In: *Mach. Learn.* 87.2 (2012), pp. 159–182.

[KGV22] K. Kreis, R. Gao, and A. Vahdat. *Denoising diffusion-based generative modeling: foundations and applications*. CVPR Tutorial. 2022.

[KH22] L. Kurscheidt and M. Hein. "Lost in Translation: Modern Image Classifiers still degrade even under simple Translations". In: *ICML 2022 Shift Happens Workshop*. July 2022.

[Kha+10] M. E. Khan, B. Marlin, G. Bouchard, and K. P. Murphy. "Variational bounds for mixed-data factor analysis". In: *NIPS*. 2010.

[Kha+18] M. E. Khan, D. Nielsen, V. Tangkaratt, W. Lin, Y. Gal, and A. Srivastava. "Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam". In: *ICML*. 2018.

[Kha20] M. E. Khan. *Deep learning with Bayesian principles*. NeurIPS tutorial. 2020.

[Kha+21] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah. "Transformers in Vision: A Survey". In: *ACM Computing Surveys December* (2021).

[Khe+20] I. Khemakhem, R. Monti, D. Kingma, and A. Hyvarinen. "ICE-BeeM: Identifiable Conditional Energy-Based Deep Models Based on Nonlinear ICA". In: *NIPS*. Vol. 33. 2020, pp. 12768–12778.

[KHH20] A. Kristiadi, M. Hein, and P. Hennig. "Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks". In: *ICML*. 2020.

[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].

[Kho+20] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. "Supervised Contrastive Learning". In: *ArXiv* abs/2004.11362 (2020).

[Kil+20] K. Killamsetty, D. Sivasubramanian, G. Ramakrishnan, and R. Iyer. "GLISTER: Generalization based Data Subset Selection for Efficient and Robust Learning". In: *arXiv preprint arXiv:2012.10630* (2020).

[Kim+18a] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, and R. Sayres. "Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)". In: *ICML*. 2018.

[Kim+18b] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, et al. "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)". In: *International conference on machine learning*. PMLR. 2018, pp. 2668–2677.

[Kim+18c] Y. Kim, S. Wiseman, A. C. Miller, D. Sontag, and A. M. Rush. "Semi-Amortized Variational Autoencoders". In: *ICML*. 2018.

[Kim+19] S. Kim, S.-G. Lee, J. Song, J. Kim, and S. Yoon. "FloWaveNet : A Generative Flow for Raw Audio". In: *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 3370–3378.

[Kim+22] S. Kim, P. Y. Lu, C. Loh, J. Smith, J. Snoek, and M. Soljačić. "Deep Learning for Bayesian Optimization of Scientific Problems with High-Dimensional Structure". In: *TMLR* (2022).

[Kin+14a] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. "Semi-Supervised Learning with Deep Generative Models". In: *NIPS*. 2014.

[Kin+14b] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. *Semi-Supervised Learning with Deep Generative Models*. 2014. arXiv: 1406.5298 [cs.LG].

[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.

[Kin+19] P.-J. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim. "The (un) reliability of saliency methods". In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019, pp. 267–280.

[Kin+21] D. P. Kingma, T. Salimans, B. Poole, and J. Ho. "Variational Diffusion Models". In: *NIPS*. July 2021.

[Kir+17] J. Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". en. In: *PNAS* 114.13 (2017), pp. 3521–3526.

[Kir+21] A. Kirsch, J. M. J. van Amersfoort, P. H. S. Torr, and Y. Gal. "On pitfalls in OoD detection: Predictive entropy considered harmful". In: *ICML Workshop on Uncertainty in Deep Learning*. 2021.

[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.

[Kiv+21] B. Kivva, G. Rajendran, P. Ravikumar, and B. Aragam. "Learning latent causal graphs via mixture oracles". In: *NIPS*. June 2021.

[Kiv+22] B. Kivva, G. Rajendran, P. Ravikumar, and B. Aragam. "Identifiability of deep generative models under mixture priors without auxiliary information". In: (June 2022). arXiv: 2206.10044 [cs.LG].

[KIW20] P. Kirichenko, P. Izmailov, and A. G. Wilson. "Why Normalizing Flows Fail to Detect Out-of-Distribution Data". In: (2020). arXiv: 2006.08545 [stat.ML].

[KJ12] J. Z. Kolter and T. S. Jaakkola. "Approximate Inference in Additive Factorial HMMs with Application to Energy Disaggregation". In: *AISTATS*. 2012.

[Kja90] U. Kjaerulff. *Triangulation of graphs – algorithms giving small total state space*. Tech. rep. R-90-09. Dept. of Math. and Comp. Sci., Aalborg Univ., Denmark, 1990.

[Kja92] U. Kjaerulff. "Optimal decomposition of probabilistic networks by simulated annealing". In: *Statistics and Computing*. Vol. 2. 1992, pp. 7–17.

[KJD18] J. Knoblauch, J. Jewson, and T. Damoulas. "Doubly Robust Bayesian Inference for Non-Stationary Streaming Data with β-Divergences". In: *NIPS*. 2018.

[KJD19] J. Knoblauch, J. Jewson, and T. Damoulas. "Generalized Variational Inference: Three arguments for deriving new Posteriors". In: (2019). arXiv: 1904.02063 [stat.ML].

[KJD21] J. Knoblauch, J. Jewson, and T. Damoulas. "An Optimization-centric View on Bayes' Rule: Reviewing and Generalizing Variational Inference". In: *JMLR* (2021).

[KJM19] N. M. Kriege, F. D. Johansson, and C. Morris. "A Survey on Graph Kernels". In: (2019). arXiv: 1903.11835 [cs.LG].

[KJV83] S. Kirkpatrick, C. G. Jr., and M. Vecchi. "Optimization by simulated annealing". In: *Science* 220 (1983), pp. 671–680.

[KK11] P. Krähenbühl and V. Koltun. "Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials". In: *NIPS*. 2011.

[KKG22] A. Kirsch, J. Kossen, and Y. Gal. "Marginal and Joint Cross-Entropies & Predictives for Online Bayesian Inference, Active Learning, and Active Sampling". In: (May 2022). arXiv: 2205.08766 [cs.LG].

[KKH20] I. Khemakhem, D. P. Kingma, and A. Hyvärinen. "Variational Autoencoders and Nonlinear ICA: A Unifying Framework". In: *AISTATS*. 2020.

[KKL20] N. Kitaev, 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.

[KKR95] K. Kanazawa, D. Koller, and S. Russell. "Stochastic Simulation Algorithms for Dynamic Probabilistic Networks". In: *UAI*. 1995.

[KKS20] F. Kunstner, R. Kumar, and M. Schmidt. "Homeomorphic-Invariance of EM: Non-Asymptotic Convergence in KL Divergence for Exponential Families via Mirror Descent". In: (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. 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.

[KL17c] J. K. Kruschke and T. M. Liddell. "The Bayesian New Statistics: Hypothesis testing, estimation, meta-analysis, and power analysis from a Bayesian perspective". In: *Psychon. Bull. Rev.* (2017).

[KL21] W. M. Kouw and M. Loog. “A review of domain adaptation without target labels”. en. In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2021).

[Kla+06] M. Klaas, M. Briers, N. de Freitas, A. Doucet, S. Maskell, and D. Lang. “Fast Particle Smoothing: If I Had a Million Particles”. In: *ICML*. 2006.

[KLA19] T. Karras, S. Laine, and T. Aila. “A Style-Based Generator Architecture for Generative Adversarial Networks”. In: *CVPR*. 2019.

[KLC98] L. P. Kaelbling, M. Littman, and A. Cassandra. “Planning and acting in Partially Observable Stochastic Domains”. In: *AIJ* 101 (1998).

[Kle+11] A. Kleiner, A. Talwalkar, P. Sarkar, and M. I. Jordan. *A scalable bootstrap for massive data*. Tech. rep. UC Berkeley, 2011.

[Kle17] G. A. Klein. *Sources of power: How people make decisions*. MIT press, 2017.

[Kle18] J. Kleinberg. “Inherent Trade-Offs in Algorithmic Fairness”. In: *ACM International Conference on Measurement and Modeling of Computer Systems*. SIGMETRICS ’18. Irvine, CA, USA: Association for Computing Machinery, June 2018, p. 40.

[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.

[KM22] J. Kuan and J. Mueller. “Back to the Basics: Revisiting Out-of-Distribution Detection Baselines”. In: *ICML PODS Workshop*. July 2022.

[KMB08] N. Kriegeskorte, M. Mur, and P. Bandettini. “Representational similarity analysis - connecting the branches of systems neuroscience”. en. In: *Front. Syst. Neurosci.* 2 (Nov. 2008), p. 4.

[KML20] A. Kumar, T. Ma, and P. Liang. “Understanding Self-Training for Gradual Domain Adaptation”. In: *ICML*. Ed. by H. D. Iii and A. Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 5468–5479.

[KMR16] J. Kleinberg, S. Mullainathan, 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.

[KN98] C.-J. Kim and C. Nelson. *State-Space Models with Regime-Switching: Classical and Gibbs-Sampling Approaches with Applications*. MIT Press, 1998.

[KNP11] K. Kersting, S. Natarajan, and D. Poole. *Statistical Relational AI: Logic, Probability and Computation*. Tech. rep. UBC, 2011.

[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, 2005.

[Koh+20a] 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.

[Koh+20b] P. W. Koh et al. “WILDS: A Benchmark of in-the-Wild Distribution Shifts”. In: (Dec. 2020). arXiv: 2012.07421 [cs.LG].

[Kol+20] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby. “Big transfer (BiT): General visual representation learning”. In: *ECCV*. Springer. 2020, pp. 491–507.

[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+19] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. “Similarity of neural network representations revisited”. In: *arXiv preprint arXiv:1905.00414* (2019).

[Kor+20] A. Korotin, V. Egiazarian, A. Asadulaev, A. Safin, and E. Burnaev. “Wasserstein-2 Generative Networks”. In: *International Conference on Learning Representations*. 2020.

[Kor+21] S. Kornblith, T. Chen, H. Lee, and M. Norouzi. “Why do better loss functions lead to less transferable features?” In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021.

[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 (2010), pp. 170–180.

[Koz92] J. R. Koza. *Genetic Programming*. https://mitpress.mit.edu/books/genetic-programming. Accessed: 2017-11-26. 1992.

[KP20] A. Kumar and B. Poole. “On Implicit Regularization in β-VAEs”. In: *ICML*. 2020.

[KPB19] I. Kobyzev, S. Prince, and M. A. Brubaker. “Normalizing Flows: An Introduction and Review of Current Methods”. In: (2019). arXiv: 1908.09257 [stat.ML].

[KPHL17] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato. “Grammar Variational Autoencoder”. In: *ICML*. 2017.

[KPS98] J. T. Key, L. R. Pericchi, and A. F. Smith. “Choosing among models when none of them are true”. In: *Proc. of the Workshop on Model Selection* (1998).

[KPS99] J. T. Key, L. R. Pericchi, and A. F. Smith. “Bayesian model choice: what and why”. In: *Bayesian statistics 6* 6 (1999), pp. 343–370.

[KPT21] J. S. Kim, G. Plumb, and A. Talwalkar. “Sanity Simulations for Saliency Methods”. In: *arXiv preprint arXiv:2105.06506* (2021).

[KR19] M. Kearns and A. Roth. *The Ethical Algorithm: The Science of Socially Aware Algorithm Design*. en. Oxford University Press, 2019.

[KR21a] M. Khan and H. Rue. "The Bayesian Learning Rule". In: (2021). arXiv: 2107.04562 [stat.ME].

[KR21b] S. Kong and D. Ramanan. "OpenGAN: Open-Set Recognition via Open Data Generation". In: *ICCV*. 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).

[Kri19] N. Kriegeskorte. *What's the best measure of representational dissimilarity?* 2019.

[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.

[Kru10] J. Kruschke. *An open letter on Bayesian Data Analysis*. Tech. rep. U. Indiana, 2010.

[Kru13] J. K. Kruschke. "Bayesian estimation supersedes the t test". In: *J. Experimental Psychology: General* 142.2 (2013), pp. 573–603.

[Kru15] J. Kruschke. *Doing Bayesian Data Analysis: A Tutorial with R, JAGS and STAN*. Second edition. Academic Press, 2015.

[KS06] L. Kocsis and C. Szepesvári. "Bandit Based Monte-Carlo Planning". In: *ECML*. 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: (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.

[KSH12a] A. Krizhevsky, I. Sutskever, and G. Hinton. "Imagenet classification with deep convolutional neural networks". In: *NIPS*. 2012.

[KSH12b] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[KSL19] S. Kornblith, J. Shlens, and Q. V. Le. "Do better ImageNet models transfer better?" In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2661–2671.

[KSL21] S. Kim, Q. Song, and F. Liang. "Stochastic gradient Langevin dynamics with adaptive drifts". In: *J. Stat. Comput. Simul.* (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.

[KSW15] D. P. Kingma, T. Salimans, and M. Welling. "Variational Dropout and the Local Reparameterization Trick". In: *NIPS*. 2015.

[KT11] A. Kulesza and B. Taskar. "k-DPPs: Fixed-size determinantal point processes". In: *ICML*. 2011.

[KTB11] D. P. Kroese, T. Taimre, and Z. I. Botev. *Handbook of Monte Carlo Methods*. en. 1 edition. Wiley, 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, 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.

[Kul+13] 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.

[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.

[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.

[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).

[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).

[Kün+19] S. R. Künzel, 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.

[Kun20] J. Kuntz. "Markov chains revisited". In: (Jan. 2020). arXiv: 2001.02183 [math.PR].

[Kur+19] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel. "Model-Ensemble Trust-Region Policy Optimization". In: *ICLR*. 2019.

[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.

[Kus+18] M. J. Kusner, J. R. Loftus, C. Russell, and R. Silva. *Counterfactual Fairness*. 2018. arXiv: 1703.06856 [stat.ML].

[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 (1964), pp. 97–106.

[KVK10] A. Klami, S. Virtanen, and S. Kaski. "Bayesian exponential family projections for coupled data sources". In: *UAI*. 2010.

[KW14] D. P. Kingma and M. Welling. "Auto-encoding variational Bayes". In: *ICLR*. 2014.

[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 (2018), pp. 75–102.

[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.

[KW19b] M. J. Kochenderfer and T. A. Wheeler. *Algorithms for Optimization*. en. The MIT Press, 2019.

[KW70] G. S. Kimeldorf and G. Wahba. "A Correspondence Between Bayesian Estimation on Stochastic Processes and Smoothing by Splines". en. In: *Ann. Math. Stat.* 41.2 (1970), pp. 495–502.

[KW96] R. E. Kass and L. Wasserman. "The Selection of Prior Distributions by Formal Rules". In: *JASA* 91.435 (1996), pp. 1343–1370.

[KWW22] M. J. Kochenderfer, T. A. Wheeler, and K. Wray. *Algorithms for Decision Making*. The MIT Press, 2022.

[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.

[Kyn+19] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila. "Improved Precision and Recall Metric for Assessing Generative Models". In: *NeurIPS*. 2019.

[KZ02] V. Kolmogorov and R. Zabih. "What energy functions can be minimized via graph cuts?" In: *Computer Vision—ECCV 2002* (2002), pp. 185–208.

[KZB19] A. Kolesnikov, X. Zhai, and L. Beyer. "Revisiting self-supervised visual representation learning". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 1920–1929.

[LA87] P. J. M. Laarhoven and E. H. L. Aarts, eds. *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, 1987.

[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.

[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.

[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).

[Lam18] B. Lambert. *A Student's Guide to Bayesian Statistics*. en. 1st ed. SAGE Publications Ltd, 2018.

[Lam92] D. Lambert. "Zero-Inflated Poisson Regression, with an Application to Defects in Manufacturing". In: *Technometrics* 34.1 (1992), pp. 1–14.

[Lan95a] K. Lange. "A gradient algorithm locally equivalent to the em algorithm". en. In: *J. of Royal Stat. Soc. Series B* 57.2 (July 1995), pp. 425–437.

[Lan95b] K. Lange. "A QUASI-NEWTON ACCELERATION OF THE EM ALGORITHM". In: *Statistica Sinica* 5.1 (1995), pp. 1–18.

[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.

[Lar+22] B. W. Larsen, S. Fort, N. Becker, and S. Ganguli. "How many degrees of freedom do we need to train deep networks: a loss landscape perspective". In: *ICLR*. 2022.

[Las08] K. B. Laskey. "MEBN: A language for first-order Bayesian knowledge bases". In: *Artif. Intell.* 172.2 (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.

[Lav00] M. Lavine. *What is Bayesian statistics and why everything else is wrong*. Tech. rep. Duke, 2000.

[Law05] N. D. Lawrence. "Probabilistic non-linear principal component analysis with Gaussian process latent variable models". In: *JMLR* 6 (2005), pp. 1783–1816.

[Law+22] D. Lawson, A. Raventós, A. Warrington, and S. Linderman. "SIXO: Smoothing Inference with Twisted Objectives". In: (June 2022). arXiv: 2206.05952 [cs.LG].

[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 Asso-*

ciation 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)*. Catalina Island, USA: AUAI, 2012.

[LB19] A. Levray and V. Belle. “Learning Tractable Probabilistic Models in Open Worlds”. In: (2019). arXiv: 1901.05847 [cs.LG].

[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.

[LBH15] Y. LeCun, Y. Bengio, and G. Hinton. “Deep learning”. en. In: *Nature* 521.7553 (May 2015), pp. 436–444.

[LBH22] X. Lu, A. Boukouvalas, and J. Hensman. “Additive Gaussian Processes Revisited”. In: *ICML*. June 2022.

[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.

[LBM06] J. A. Lasserre, C. M. Bishop, and T. P. Minka. “Principled Hybrids of Generative and Discriminative Models”. In: *CVPR*. Vol. 1. June 2006, pp. 87–94.

[LBS01] T. Lefebvre, H. Bruyninckx, and J. D. Schutter. “Comment on “A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators””. In: *IEEE Trans. on Automatic Control* 47.8 (2001), pp. 1406–1409.

[LBS17] 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: *NIPS*. 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 Continual Learning Settings with Data Distribution Drift Analysis”. In: (2021). arXiv: 2104.01678 [cs.LG].

[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.

[Le+18] T. A. Le, M. Igl, T. Rainforth, T. Jin, and F. Wood. “Auto-Encoding Sequential Monte Carlo”. In: *ICLR*. 2018.

[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. 29–52.

[Le+19] T. A. Le, A. R. Kosiorek, N Siddharth, Y. W. Teh, and F. Wood. “Revisiting Reweighted Wake-Sleep for Models with Stochastic Control Flow”. In: *UAI*. 2019.

[LeC+89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. “Backpropagation Applied to Handwritten Zip Code Recognition”. In: *Neural Computation* 1.4 (1989), pp. 541–551.

[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.

[Lee04] M. D. Lee. “Models, parameters and priors in Bayesian inference”. 2004.

[Lee06] J. de Leeuw. “Principal Component Analysis of Binary Data by Iterated Singular Value Decomposition”. In: *Comput. Stat. Data Anal.* 50.1 (2006), pp. 21–39.

[Lee+09] H. Lee, R. B. Grosse, R. Ranganath, and A. Ng. “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations”. In: *ICML '09*. 2009.

[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.

[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.

[Lei+18] J. Lei, M. G'Sell, A. Rinaldo, R. J. Tibshirani, and L. Wasserman. “Distribution-Free Predictive Inference For Regression”. In: *JASA* (2018).

[Lei+20] F. Leibfried, V. Dutordoir, S. T. John, and N. Durrande. “A Tutorial on Sparse Gaussian Processes and Variational Inference”. In: (2020). arXiv: 2012.13962 [cs.LG].

[Lem09] C. Lemieux. *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer, New York, NY, 2009.

[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.

[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).

[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.

[Lev18] S. Levine. "Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review". In: (2018). arXiv: 1805.00909 [cs.LG].

[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.

[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: (2021). arXiv: 2110.04286 [cs.LG].

[LFG21] F. Lin, X. Fang, and Z. Gao. "Distributionally Robust Optimization: A review on theory and applications". In: *Numer. Algebra Control Optim.* 12.1 (2021), pp. 159–212.

[LG94] D. D. Lewis and W. A. Gale. "A sequential algorithm for training text classifiers". In: *SIGIR94*. Springer. 1994, pp. 3–12.

[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.

[LHF17] R. M. Levy, A. Haldane, and W. F. Flynn. "Potts Hamiltonian models of protein co-variation, free energy landscapes, and evolutionary fitness". en. In: *Curr. Opin. Struct. Biol.* 43 (2017), pp. 55–62.

[LHF20] J. Liang, D. Hu, and J. Feng. "Do We Really Need to Access the Source Data? Source Hypothesis Transfer for Unsupervised Domain Adaptation". In: *ICML*. Ed. by H. D. Iii and A. Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 6028–6039.

[LHLT15] Y. Li, J. M. Hernandez-Lobato, and R. E. Turner. "Stochastic Expectation Propagation". In: *NIPS*. 2015.

[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.

[Li+10] L. Li, W. Chu, J. Langford, and R. E. Schapire. "A contextual-bandit approach to personalized news article recommendation". In: *WWW*. 2010.

[Li+16] C. Li, C. Chen, D. Carlson, and L. Carin. "Preconditioned Stochastic Gradient Langevin Dynamics for Deep Neural Networks". In: *AAAI*. 2016.

[Li+17a] A. Li, A. Jabri, A. Joulin, and L. van der Maaten. "Learning visual n-grams from web data". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 4183–4192.

[Li+17b] 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.

[Li+17c] L. Li, K. Jamieson, G. De Salvo, A. Rostamizadeh, and A. Talwalkar. "Hyperband: bandit-based configuration evaluation for hyperparameter optimization". In: *ICLR*. 2017.

[Li+17d] 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].

[Li+17e] 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 (2017), pp. 1913–1939.

[Li18] Y. Li. "Deep Reinforcement Learning". In: (2018). arXiv: 1810.06339 [cs.LG].

[Li+18a] C. Li, H. Farkhoor, R. Liu, and J. Yosinski. "Measuring the Intrinsic Dimension of Objective Landscapes". In: *ICLR*. 2018.

[Li+18b] C. Li, H. Farkhoor, R. Liu, and J. Yosinski. "Measuring the Intrinsic Dimension of Objective Landscapes". In: *ICLR*. 2018.

[Li+18c] 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, 2018, pp. 603–611.

[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.

[Li+20] 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*. 2020.

[Li+21] Y. Li, R. Pogodin, D. J. Sutherland, and A. Gretton. "Self-Supervised Learning with Kernel Dependence Maximization". In: *NeurIPS*. 2021.

[Li+22] R. Li, W. Li, Y. Yang, H. Wei, J. Jiang, and Q. Bai. "Swinv2-Imagen: Hierarchical Vision Transformer Diffusion Models for Text-to-Image Generation". In: (Oct. 2022). arXiv: 2210.09549 [cs.CV].

[Lia+] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, and Y. Wu. "Helm: Holistic Evaluation of Language Models (HELM), a framework to increase the transparency of language models (https://arxiv.org/abs/2211.09110)". en.

[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.

[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.

[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.

[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.

[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.

[Lin13b] D. A. Linzer. "Dynamic Bayesian Forecasting of Presidential Elections in the States". In: *JASA* 108.501 (2013), pp. 124–134.

[Lin+17a] 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.

[Lin+17b] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. “Feature pyramid networks for object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[Lin19] J. K. Lindelov. *Common statistical tests are linear models (or: how to teach stats)*. Blog post. 2019.

[Lin+20] H. Lin, H. Chen, T. Zhang, C. Laroche, and K. Choromanski. “Demystifying Orthogonal Monte Carlo and Beyond”. In: (2020). arXiv: 2005.13590 [cs.LG].

[Lin+21a] T. Lin, Y. Wang, X. Liu, and X. Qiu. “A Survey of Transformers”. In: (2021). arXiv: 2106.04554 [cs.LG].

[Lin+21b] A. Lindholm, N. Wahlström, F. Lindsten, and T. B. Schön. *Machine Learning - A First Course for Engineers and Scientists*. 2021.

[Lin+21c] J. Lindqvist, S. Särkkä, Á. F. García-Fernández, M. Raitoharju, and L. Svensson. “Posterior linearisation smoothing with robust iterations”. In: (Dec. 2021). arXiv: 2112.03969 [math.OC].

[Lin56] D. Lindley. “On a measure of the information provided by an experiment”. In: *The Annals of Math. Stat.* (1956), 986–1005.

[Lin88a] B. Lindsay. “Composite Likelihood Methods”. In: *Contemporary Mathematics* 80.1 (1988), pp. 221–239.

[Lin88b] R Linsker. “Self-organization in a perceptual network”. In: *Computer* 21.3 (1988), pp. 105–117.

[Lin88c] R. Linsker. “Self-organization in a perceptual network”. In: *Computer* 21.3 (1988), pp. 105–117.

[Lin92] L.-J. Lin. “Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching”. In: *Mach. Learn.* 8.3-4 (1992), pp. 293–321.

[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.

[LIS20] Z. Lu, E. Ie, and F. Sha. “Mean-Field Approximation to Gaussian-Softmax Integral with Application to Uncertainty Estimation”. In: (June 2020). arXiv: 2006.07584 [cs.LG].

[Liu01] J. Liu. *Monte Carlo Strategies in Scientific Computation*. Springer, 2001.

[Liu+15] Z. Liu, P. Luo, X. Wang, and X. Tang. “Deep Learning Face Attributes in the Wild”. In: *ICCV*. 2015.

[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: (2018). arXiv: 1809.09853 [math.OC].

[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.

[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: (2019). arXiv: 1909.06541 [stat.ML].

[Liu+19b] R. Liu, J. Regier, N. Tripuraneni, M. I. Jordan, and J. McAuliffe. “Rao-Blackwellized Stochastic Gradients for Discrete Distributions”. In: *ICML*. 2019.

[Liu+19c] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692* (2019).

[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*. 2020.

[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.

[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).

[Liu+20d] J. Z. Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax-Weiss, and B. Lakshminarayanan. “Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness”. In: *NIPS*. 2020.

[Liu+21a] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. “Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing”. In: (2021). arXiv: 2107.13586 [cs.CL].

[Liu+21b] W. Liu, X. Wang, J. D. Owens, and Y. Li. “Energy-based Out-of-distribution Detection”. In: *NIPS*. 2021.

[Liu+22a] J. Z. Liu, S. Padhy, J. Ren, Z. Lin, Y. Wen, G. Jerfel, Z. Nado, J. Snoek, D. Tran, and B. Lakshminarayanan. “A Simple Approach to Improve Single-Model Deep Uncertainty via Distance-Awareness”. In: *JMLR* 23 (May 2022), pp. 1–62.

[Liu+22b] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. “A ConvNet for the 2020s”. In: (2022). arXiv: 2201.03545 [cs.CV].

[LJ08] P. Liang and M. I. Jordan. “An Asymptotic Analysis of Generative, Discriminative, and Pseudolikelihood Estimators”. In: *International Conference on Machine Learning (ICML)*. 2008.

[Lju87] L. Ljung. *System Identificiation: Theory for the User*. Prentice Hall, 1987.

[LJY19] S. Liu, Y. Jiang, and T. Yu. “Modelling RNA-Seq data with a zero-inflated mixture Poisson linear model”. en. In: *Genet. Epidemiol.* 43.7 (2019), pp. 786–799.

[LK07] P. Liang and D. Klein. *Structured Bayesian Nonparametric Models with Variational Inference*. ACL Tutorial. 2007.

[LK09] P. Liang and D. Klein. “Online EM for Unsupervised Models”. In: *NAACL*. 2009.

[LKJ09] D. Lewandowski, D. Kurowicka, and H. Joe. “Generating random correlation matrices based on vines and extended onion method”. In: *J. Multivar. Anal.* 100.9 (2009), pp. 1989–2001.

[LL02] P. Larranaga and J. A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.

[LL17] S. M. Lundberg and S.-I. Lee. “A unified approach to interpreting model predictions”. In: *NIPS*. 2017, pp. 4765–4774.

[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).

[LLJ16] Q. Liu, J. Lee, and M. Jordan. "A kernelized Stein discrepancy for goodness-of-fit tests". In: *International conference on machine learning*. 2016, pp. 276–284.

[LLN06] B. Lehmann, D. Lehmann, and N. Nisan. "Combinatorial auctions with decreasing marginal utilities". In: *Games and Economic Behavior* 55.2 (2006), pp. 270–296.

[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.

[LM03] T. S. Lee and D. Mumford. "Hierarchical Bayesian inference in the visual cortex." In: *Journal of the Optical Society of America. A, Optics, image science, and vision* 20 7 (2003), pp. 1434–48.

[LM11] H. Larochelle and I. Murray. "The neural autoregressive distribution estimator". In: *AISTATS*. Vol. 15. 2011, pp. 29–37.

[LM20] M. L. Leavitt and A. Morcos. "Towards falsifiable interpretability research". In: *arXiv preprint arXiv:2010.12016* (2020).

[LMP01] J. Lafferty, A. McCallum, and F. Pereira. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *ICML*. 2001.

[LMS16] B. Leimkuhler, C. Matthews, and G. Stoltz. "The computation of averages from equilibrium and nonequilibrium Langevin molecular dynamics". In: *IMA J. Numer. Anal.* 36.1 (2016), pp. 13–79.

[LN01] S. Lauritzen and D. Nilsson. "Representing and solving decision problems with limited information". In: *Management Science* 47 (2001), pp. 1238–1251.

[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.

[LN96] o. Lee and J. A. Nelder. "Hierarchical Generalized Linear Models". In: *J. of Royal Stat. Soc. Series B* 58.4 (1996), pp. 619–678.

[Loc+18] F. Locatello, S. Bauer, M. Lucic, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem. "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations". In: (2018). arXiv: 1811.12359 [cs.LG].

[Loc+20a] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem. "A Sober Look at the Unsupervised Learning of Disentangled Representations and their Evaluation". In: *Journal of Machine Learning Research* 21.209 (2020), pp. 1–62.

[Loc+20b] F. Locatello, B. Poole, G. Rätsch, B. Schölkopf, O. Bachem, and M. Tschannen. "Weakly-supervised disentanglement without compromises". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 6348–6359.

[Lod+02] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. "Text classification using string kernels". en. In: *J. Mach. Learn. Res.* (2002).

[Loe04] H Loeliger. "An introduction to factor graphs". In: *IEEE Signal Process. Magazine* 21.1 (2004), pp. 28–41.

[Loe+07] H Loeliger, J Dauwels, J Hu, S Korl, L Ping, and F. R. Kschischang. "The Factor Graph Approach to Model-Based Signal Processing". In: *Proc. IEEE* 95.6 (2007), pp. 1295–1322.

[Loe+16] H.-A. Loeliger, L. Bruderer, H. Malmberg, F. Wadehn, and N. Zalmai. "On Sparsity by NUV-EM, Gaussian Message Passing, and Kalman Smoothing". In: *ITA Workshop*. 2016.

[Loi+21] N. Loizou, S. Vaswani, I. Laradji, and S. Lacoste-Julien. "Stochastic Polyak step-size for SGD: An adaptive learning rate for fast convergence". In: *AISTATS*. 2021.

[Lon+18] M. Long, Z. Cao, J. Wang, and M. I. Jordan. "Conditional adversarial domain adaptation". In: *Neural Information Processing Systems* (2018).

[Lot+22] S. Lotfi, P. Izmailov, G. Benton, M. Goldblum, and A. G. Wilson. "Bayesian Model Selection, the Marginal Likelihood, and Generalization". In: *ICML*. 2022.

[Lov+20] T. Lovett, M. Briers, M. Charalambides, R. Jersakova, J. Lomax, and C. Holmes. "Inferring proximity from Bluetooth Low Energy RSSI with Unscented Kalman Smoothers". In: (2020). arXiv: 2007.05057 [eess.SP].

[Lov83] L. Lovász. "Submodular functions and convexity". In: *Mathematical programming the state of the art*. Springer, 1983, pp. 235–257.

[LÖW21] T. van de Laar, A. Özçelikkale, and H. Wymeersch. "Application of the Free Energy Principle to Estimation and Control". In: *IEEE Trans. Signal Process.* 69 (2021), pp. 4234–4244.

[LP01] U. Lerner and R. Parr. "Inference in Hybrid Networks: Theoretical Limits and Practical Algorithms". In: *UAI*. 2001.

[LP03] M. G. Lagoudakis and R. Parr. "Least-Squares Policy Iteration". In: *JMLR* 4 (2003), pp. 1107–1149.

[LP06] N. Lartillot and H. Philippe. "Computing Bayes factors using thermodynamic integration". en. In: *Systematic Biology* 55.2 (2006), pp. 195–207.

[LP20] P. Lemberger and I. Panico. "A Primer on Domain Adaptation". In: (Jan. 2020). arXiv: 2001.09994 [cs.LG].

[LPB17] B. Lakshminarayanan, A. Pritzel, and C. Blundell. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In: *NIPS*. 2017.

[LPO17] D. Lopez-Paz and M. Oquab. "Revisiting classifier two-sample tests". In: *International Conference on Learning Representations*. 2017.

[LPR17] D. Lopez-Paz and M. Ranzato. "Gradient Episodic Memory for Continual Learning". In: *NIPS*. 2017.

[LR19] F. Lattimore and D. Rohde. "Replacing the do-calculus with Bayes rule". In: (2019). arXiv: 1906.07125 [stat.ML].

[LR85] T. L. Lai and H. Robbins. "Asymptotically efficient adaptive allocation rules". en. In: *Adv. Appl. Math.* (1985).

[LR87] R. J. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. Wiley and Son, 1987.

[LR95] C. Liu and D. Rubin. "ML Estimation of the T distribution using EM and its extensions, ECM and ECME". In: *Statistica Sinica* 5 (1995), pp. 19–39.

[LRC19] Y. Li, B. I. P. Rubinstein, and T. Cohn. "Truth Inference at Scale: A Bayesian Model for Adjudicating Highly Redundant Crowd Annotations". In: *WWW*. 2019.

[LS01] D. Lee and S. Seung. "Algorithms for non-negative matrix factorization". In: *NIPS*. 2001.

[LS19] T. Lattimore and C. Szepesvari. *Bandit Algorithms*. Cambridge, 2019.

[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.

[LST15] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. "Human-level concept learning through probabilistic program induction". In: *Science* 350 (2015), pp. 1332 –1338.

[LST21] N. Loo, S. Swaroop, and R. E. Turner. "Generalized Variational Continual Learning". In: *ICLR*. 2021.

[LST90] J. K. Lenstra, D. B. Shmoys, and É. Tardos. "Approximation algorithms for scheduling unrelated parallel machines". In: *Mathematical programming*. 1990.

[LSV09] J. Lee, M. Sviridenko, and J. Vondrák. "Submodular maximization over multiple matroids via generalized exchange properties". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques* (2009), pp. 244–257.

[LSW15] Y. T. Lee, A. Sidford, and S. C.-w. Wong. "A faster cutting 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.

[LSZ15] Y. Li, K. Swersky, and R. Zemel. "Generative Moment Matching Networks". In: *ICML*. 2015.

[LT16] M.-Y. Liu and O. Tuzel. "Coupled Generative Adversarial Networks". In: *NIPS*. 2016, pp. 469–477.

[LTW15] Q. Li, C. Tai, and E Weinan. "Stochastic modified equations and adaptive stochastic gradient algorithms". In: *ICML*. 2015.

[Lu+21a] X. Lu, I. Osband, B. Van Roy, and Z. Wen. "Evaluating Probabilistic Inference in Deep Learning: Beyond Marginal Predictions". In: (2021). arXiv: 2107.09224 [cs.LG].

[Lu+21b] X. Lu, B. Van Roy, V. Dwaracherla, M. Ibrahimi, I. Osband, and Z. Wen. "Reinforcement Learning, Bit by Bit". In: (Mar. 2021). arXiv: 2103.04047 [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: (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 (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 Luttinen. "BayesPy: variational Bayesian inference in Python". In: *JMLR* (2016).

[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: *Statisica 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).

[Ly+20] A. Ly et al. "The Bayesian Methodology of Sir Harold Jeffreys as a Practical Alternative to the P Value Hypothesis Test". In: *Computational Brain & Behavior* 3.2 (June 2020), pp. 153–161.

[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).

[Lyu+20] X.-K. Lyu, Y. Xu, X.-F. Zhao, X.-N. Zuo, and C.-P. Hu. "Beyond psychology: prevalence of p value and confidence interval misinterpretation across different fields". In: *Journal of Pacific Rim Psychology* 14 (2020).

[LZ20] B. Lim and S. Zohren. "Time Series Forecasting With Deep Learning: A Survey". In: (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. Liévin, and O. Winther. "BIVA: A Very Deep Hierarchy of 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).

[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 (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.

[Mac99] D. MacKay. "Comparision 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^m-Bayes: Narrowing the Empirical Risk Gap in the Misspecified Bayesian Regime". In: (2020). arXiv: 2010.09629 [cs.LG].

[Mah07] R. P. S. Mahler. *Statistical Multisource-Multitarget Information Fusion*. Artech House, Inc., 2007.

[Mah13] R Mahler. "Statistics 102 for Multisource-Multitarget Detection and Tracking". In: *IEEE J. Sel. Top. Signal Process.* 7.3 (2013), pp. 376–389.

[Mah+18] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten. "Exploring the limits of weakly supervised pretraining". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 181–196.

[Mah+19] N. Maheswaranathan, A. H. Williams, M. D. Golub, S. Ganguli, and D. Sussillo. "Universality and individuality in neural dynamics across large populations of recurrent networks". In: *Advances in Neural Information Processing Systems* 2019 (2019), p. 15629.

[Mah+23] K. Mahowald, A. A. Ivanova, I. A. Blank, N. Kanwisher, J. B. Tenenbaum, and E. Fedorenko. "Dissociating language and thought in large language models: a cognitive perspective". In: (Jan. 2023). arXiv: 2301.06627 [cs.CL].

[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 (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 (2022), pp. 28–51.

[Mak+15a] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. "Adversarial Autoencoders". In: (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. Makkuva, 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+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: (2019). arXiv: 1906.07516 [cs.LG].

[Man22a] V. Manokhin. *Awesome Conformal Prediction*. Version v1.0.0. Apr. 2022.

[Man22b] V. Manokhin. "Machine Learning for Probabilistic Prediction". PhD thesis. Royal Holloway, University of London, June 2022.

[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.

[MAP17] H. Martínez Alonso and B. Plank. "When is multitask learning effective? Semantic sequence prediction under varying data conditions". In: *Proc. European ACL*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 44–53.

[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 Bionformatics* 7 (2006).

[Mar+10] B. M. Marlin, K. Swersky, B. Chen, and N. de Freitas. "Inductive Principles for Restricted Boltzmann Machine Learning". In: *AISTATS*. 2010.

[Mar10a] J Martens. "Deep learning via Hessian-free optimization". In: *ICML*. 2010.

[Mar10b] J. Martens. "Learning the Linear Dynamical System with ASOS". In: *ICML*. ICML'10. Omnipress, 2010, pp. 743–750.

[Mar16] J. Martens. "Second-order optimization for neural networks". PhD thesis. Toronto, 2016.

[Mar18] O. Martin. *Bayesian analysis with Python*. Packt, 2018.

[Mar20] J. Martens. "New insights and perspectives on the natural gradient method". In: *JMLR* (2020).

[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: (2020). arXiv: 2010.15277 [cs.LG].

[Mat14] C. Mattfeld. "Implementing spectral methods for hidden Markov models with real-valued emissions". MA thesis. ETH Zurich, 2014.

[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*. 2016, pp. 231–239.

[Mav16] Mavrogonatou, L And Vyshemirsky, "Sequential Importance Sampling for Online Bayesian Changepoint Detection". In: *22nd International Conference on Computational Statistics*. 2016.

[May+19] A. May, J. Zhang, T. Dao, and C. Ré. "On the downstream performance of compressed word embeddings". In: *Advances in neural information processing systems* 32 (2019), p. 11782.

[May79] P. Maybeck. *Stochastic models, estimation, and control*. Academic Press, 1979.

[Maz+22] P. Mazzaglia, T. Verbelen, O. Çatal, and B. Dhoedt. "The Free Energy Principle for Perception and Action: A Deep Learning Perspective". en. In: *Entropy* 24.2 (2022).

[MAZA18] X. B. P. amd Marcin Andrychowicz, W. Zaremba, and P. Abbeel. "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization". In: *ICRA*. 2018, pp. 1–8.

[MB16] Y. Miao and P. Blunsom. "Language as a Latent Variable: Discrete Generative Models for Sentence Compression". In: *EMNLP*. 2016.

[MB18] A. Mensch and M. Blondel. "Differentiable Dynamic Programming for Structured Prediction and Attention". In: *ICML*. 2018.

[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*. 2021.

[MB88] T. Mitchell and J. Beauchamp. "Bayesian Variable Selection in Linear Regression". In: *JASA* 83 (1988), pp. 1023–1036.

[MBJ20] T. M. Moerland, J. Broekens, and C. M. Jonker. "Model-based Reinforcement Learning: A Survey". In: (2020). arXiv: 2006.16712 [cs.LG].

[MBL20] B. Mirzasoleiman, J. Bilmes, and J. Leskovec. "Coresets for data-efficient training of machine learning models". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 6950–6960.

[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).

[MC03] P. Moscato and C. Cotta. "A Gentle Introduction to Memetic Algorithms". en. In: *Handbook of Metaheuristics*. International Series in Operations Research & Management Science. Springer, Boston, MA, 2003, pp. 105–144.

[MC19] P. Moreno Comellas. "Vision as inverse graphics for detailed scene understanding". en. PhD thesis. 2019.

[McA23] D. McAllester. "On the Mathematics of Diffusion Models". In: (Jan. 2023). arXiv: 2301.11108 [cs.LG].

[McA99] D. A. McAllester. "PAC-Bayesian model averaging". In: *Proceedings of the twelfth annual conference on computational learning theory*. 1999.

[McE20] R. McElreath. *Statistical Rethinking: A Bayesian Course with Examples in R and Stan (2nd edition)*. en. Chapman and Hall/CRC, 2020.

[McG54] W. McGill. "Multivariate information transmission". In: *Psychometrika* 19 (1954), pp. 97–116.

[McM+13] H. B. McMahan, G. Holt, D Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, et al. "Ad click prediction: a view from the trenches". In: *KDD*. 2013, pp. 1222–1230.

[Md+19] C. de Masson d'Autume, M. Rosca, J. Rae, and S. Mohamed. "Training language GANs from Scratch". In: (2019). arXiv: 1905.09922 [cs.CL].

[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.

[MDA15] D. Maclaurin, D. Duvenaud, and R. P. Adams. "Gradient-based Hyperparameter Optimization through Reversible Learning". In: *ICML*. 2015.

[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.

[MDR94] S. Muggleton and L. De Raedt. "Inductive logic programming: Theory and methods". In: *The Journal of Logic Programming* 19 (1994), pp. 629–679.

[Med+21] M. A. Medina, J. L. M. Olea, C. Rush, and A. Velez. "On the Robustness to Misspecification of α-Posteriors and Their Variational Approximations". In: (2021). arXiv: 2104.08324 [stat.ML].

[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.

[Meh72] R Mehra. "Approaches to adaptive filtering". In: *IEEE Trans. Automat. Contr.* 17.5 (Oct. 1972), pp. 693–698.

[Mei18a] N. Meinshausen. "Causality from a distributional robustness point of view". In: *IEEE Data Science Workshop (DSW)*. 2018, pp. 6–10.

[Mei18b] N. Meinshausen. "CAUSALITY FROM A DISTRIBUTIONAL ROBUSTNESS POINT OF VIEW". In: *2018 IEEE Data Science Workshop (DSW)*. 2018, pp. 6–10.

[Men+21] A. K. Menon, S. Jayasumana, A. S. Rawat, H. Jain, A. Veit, and S. Kumar. "Long-tail learning via logit adjustment". In: *ICLR*. 2021.

[Mer] *Definition of interpret*. 2022. URL: https://www.merriam-webster.com/dictionary/interpret.

[Mer+00] R. van der Merwe, A. Doucet, N. de Freitas, and E. Wan. "The Unscented Particle Filter". In: *NIPS-13*. 2000.

[Met16] C. Metz. *In Two Moves, AlphaGo and Lee Sedol Redefined the Future*. 2016. URL: https://www.wired.com/2016/03/two-moves-alphago-lee-sedol-redefined-future/ (visited on 01/07/2022).

[Met+16] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. "Unrolled Generative Adversarial Networks". In: (2016).

[Met+17] L. Metz, J. Ibarz, N. Jaitly, and J. Davidson. "Discrete Sequential Prediction of Continuous Actions for Deep RL". In: (2017). arXiv: 1705.05035 [cs.LG].

[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.

[Mey+18] F. Meyer, T. Kropfreiter, J. Williams, R. Lau, F. Hlawatsch, P. Braca, and M. Win. "Message passing algorithms for scalable multitarget tracking". In: *Proc. IEEE* 106.2 (2018).

[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.

[Mey22] S. Meyn. *Control Systems and Reinforcement Learning*. Cambridge, 2022.

[MFP00] A. McCallum, D. Freitag, and F. Pereira. "Maximum Entropy Markov Models for Information Extraction and Segmentation". In: *ICML*. 2000.

[MFR20] G. M. Martin, D. T. Frazier, and C. P. Robert. "Computing Bayes: Bayesian Computation from 1763 to the 21st Century". In: (2020). arXiv: 2004.06425 [stat.CO].

[MG05] I. Murray and Z. Ghahramani. *A note on the evidence and Bayesian Occam's razor*. Tech. rep. Gatsby, 2005.

[MG15] J. Martens and R. Grosse. "Optimizing Neural Networks with Kronecker-factored Approximate Curvature". In: *ICML*. 2015.

[MG18] A. Malinin and M. Gales. "Predictive Uncertainty Estimation via Prior Networks". In: (2018). arXiv: 1802.10501 [stat.ML].

[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.

[MGN18a] L. Mescheder, A. Geiger, and S. Nowozin. "Which Training Methods for GANs do actually Converge?" In: *ICML*. 2018.

[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.

[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, K Grauman, N Cesa-Bianchi, and R Garnett. Curran Associates, Inc., 2018, pp. 1800–1809.

[MH12] R. Mazumder and T. Hastie. *The Graphical Lasso: New Insights and Alternatives*. Tech. rep. Stanford Dept. Statistics, 2012.

[MH20] I. Mordatch and J. Hamrick. *ICML tutorial on model-based methods in reinforcement learning*. https://sites.google.com/corp/view/mbrl-tutorial. 2020.

[MHB17] S. Mandt, M. D. Hoffman, and D. M. Blei. "Stochastic Gradient Descent As Approximate Bayesian Inference". In: *JMLR* 18.1 (2017), pp. 4873–4907.

[MHH14] F. Meyer, O. Hlinka, and F. Hlawatsch. "Sigma point belief propagation". In: *IEEE Signal Processing Letters* 21.2 (2014), pp. 145–149.

[MHN13] A. L. Maas, A. Y. Hannun, and A. Y. Ng. "Rectifier Nonlinearities Improve Neural Network Acoustic Models". In: *ICML*. Vol. 28. 2013.

[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.

[Mil+05] B. Milch, B. Marthi, S. Russell, D. Sontag, D. Ong, and A. Kolobov. "BLOG: Probabilistic Models with Unknown Objects". In: *IJCAI*. 2005.

[Mil19] T. Miller. "Explanation in artificial intelligence: Insights from the social sciences". In: *Artificial intelligence* 267 (2019), pp. 1–38.

[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*. 2020.

[Mil+21] 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.

[Min00a] T. Minka. *Bayesian linear regression*. Tech. rep. MIT, 2000.

[Min00b] T. Minka. *Bayesian model averaging is not model combination*. Tech. rep. MIT Media Lab, 2000.

[Min00c] T. Minka. *Estimating a Dirichlet distribution*. Tech. rep. MIT, 2000.

[Min01a] T. Minka. "A family of algorithms for approximate Bayesian inference". PhD thesis. MIT, 2001.

[Min01b] T. Minka. "Expectation Propagation for approximate Bayesian inference". In: *UAI*. 2001.

[Min04] T. Minka. *Power EP*. Tech. rep. MSR-TR-2004-149. 2004.

[Min05] T. Minka. *Divergence measures and message passing*. Tech. rep. MSR Cambridge, 2005.

[Min+18] T. Minka, J. Winn, J. Guiver, Y. Zaykov, D. Fabian, and J. Bronskill. *Infer.NET 0.3*. Microsoft Research Cambridge. 2018.

[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.

[Min99] T. Minka. *Pathologies of Orthodox Statisics*. Tech. rep. MIT Media Lab, 1999.

[Mis+16] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. "Cross-stitch Networks for Multi-task Learning". In: *CVPR*. 2016.

[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.

[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.

[Mit+20] J. Mitrovic, B. McWilliams, J. Walker, L. Buesing, and C. Blundell. *Representation Learning via Invariant Causal Mechanisms*. 2020. arXiv: 2010.07922 [cs.LG].

[Mit97] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[Miy+18a] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. "Spectral Normalization for Generative Adversarial Networks". In: *ICLR*. 2018.

[Miy+18b] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. "Spectral Normalization for Generative Adversarial Networks". In: *ICLR*. 2018.

[Miy+18c] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. "Spectral Normalization for Generative Adversarial Networks". In: *International Conference on Learning Representations*. 2018.

[MJ97] M. Meila and M. Jordan. *Triangulation by continuous embedding*. Tech. rep. 1605. MIT AI Lab, 1997.

[MK05] J. Mooij and H. Kappen. "Sufficient conditions for convergence of loopy belief propagation". In: *UAI*. 2005.

[MK07] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions (Second Edition)*. Wiley, 2007.

[MK18] T. Miyato and M. Koyama. "cGANs with Projection Discriminator". In: *International Conference on Learning Representations*. 2018.

[MK19] J. Menick and N. Kalchbrenner. "Generating high fidelity images with subscale pixel networks and multidimensional upscaling". In: *ICLR*. 2019.

[MK97] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, 1997.

[MKH19] R. Müller, S. Kornblith, and G. E. Hinton. "When does label smoothing help?" In: *NIPS*. 2019, pp. 4694–4703.

[MKL11] O. Martin, R. Kumar, and J. Lao. *Bayesian Modeling and Computation in Python*. CRC Press, 2011.

[MKL21] O. A. Martin, R. Kumar, and J. Lao. *Bayesian Modeling and Computation in Python*. CRC Press, 2021.

[MKS21] K. Murphy, A. Kumar, and S. Serghiou. "Risk score learning for COVID-19 contact tracing apps". In: *Machine Learning for Healthcare*. 2021.

[ML02] T. Minka and J. Lafferty. "Expectation-propagation for the Generative Aspect Model". In: *UAI*. Morgan Kaufmann Publishers Inc., 2002, pp. 352–359.

[ML16] S. Mohamed and B. Lakshminarayanan. "Learning in Implicit Generative Models". In: (2016). arXiv: 1610.03483 [stat.ML].

[MLN19] P. Michel, O. Levy, and G. Neubig. "Are Sixteen Heads Really Better than One?" In: *NIPS*. 2019.

[MLW19] V. Masrani, T. A. Le, and F. Wood. "The Thermodynamic Variational Objective". In: *NIPS*. Curran Associates, Inc., 2019, pp. 11521–11530.

[MM01] T. K. Marks and J. R. Movellan. *Diffusion networks, products of experts, and factor analysis*. Tech. rep. University of California San Diego, 2001.

[MM90] D. Q. Mayne and H Michalska. "Receding horizon control of nonlinear systems". In: *IEEE Trans. Automat. Contr.* 35.7 (1990), pp. 814–824.

[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.

[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.

[MMT17] C. J. Maddison, A. Mnih, and Y. W. Teh. "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables". In: *ICLR*. 2017.

[MN89] P. McCullagh and J. Nelder. *Generalized linear models*. 2nd edition. Chapman and Hall, 1989.

[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.

[MNG17b] L. Mescheder, S. Nowozin, and A. Geiger. "The numerics of gans". In: *Advances in Neural Information Processing Systems*. 2017, pp. 1825–1835.

[Mni+15] V. Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), pp. 529–533.

[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.

[MO14] M. Mirza and S. Osindero. "Conditional generative adversarial nets". In: *arXiv preprint arXiv:1411.1784* (2014).

[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.

[Moh+20] S. Mohamed, M. Rosca, M. Figurnov, and A. Mnih. "Monte Carlo Gradient Estimation in Machine Learning". In: *JMLR* 21.132 (2020), pp. 1–62.

[Mon81] G. Monge. "Mémoire sur la théorie des déblais et des remblais". In: *Histoire de l'Académie Royale des Sciences* (1781), pp. 666–704.

[Mor+11] F. Morcos, A. Pagnani, B. Lunt, A. Bertolino, D. S. Marks, C. Sander, R. Zecchina, J. N. Onuchic, T. Hwa, and M. Weigt. "Direct-coupling analysis of residue coevolution captures native contacts across many protein families". en. In: *Proc. Natl. Acad. Sci. U. S. A.* 108.49 (2011), E1293–301.

[Mor+16] R. D. Morey, R. Hoekstra, J. N. Rouder, M. D. Lee, and E.-J. Wagenmakers. "The fallacy of placing confidence in confidence intervals". en. In: *Psychon. Bull. Rev.* 23.1 (2016), pp. 103–123.

[Mor+21a] 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.

[Mor+21b] W. Morningstar, S. Vikram, C. Ham, A. Gallagher, and J. Dillon. "Automatic Differentiation Variational Inference with Mixtures". In: *AISTATS*. Ed. by A. Banerjee and K. Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, 2021, pp. 3250–3258.

[Mor63] T. Morimoto. "Markov Processes and the H-Theorem". In: *J. Phys. Soc. Jpn.* 18.3 (1963), pp. 328–331.

[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.

[Mov08] J. R. Movellan. "A minimum velocity approach to learning". In: *unpublished draft, Jan* (2008).

[MP01] K. Murphy and M. Paskin. "Linear time inference in hierarchical HMMs". In: *NIPS*. 2001.

[MP21] D. Mazza and M. Pagani. "Automatic Differentiation in PCF". In: *Proc. ACM Program. Lang.* 5.POPL (2021).

[MP95] D. MacKay and L. Peto. "A hierarchical dirichlet language model". In: *Natural Language Engineering* 1.3 (1995), pp. 289–307.

[MPS18] O. Mangoubi, N. S. Pillai, and A. Smith. "Does Hamiltonian Monte Carlo mix faster than a random walk on multimodal densities?" In: (2018). arXiv: 1808.03230 [math.PR].

[MPT13] K. Mohan, J. Pearl, and J. Tian. "Graphical models for inference with missing data". In: *NIPS*. 2013.

[MR09] A. Melkumyan and F. Ramos. "A Sparse Covariance Function for Exact Gaussian Process Inference in Large Datasets". In: *IJCAI*. 2009, pp. 1936–1942.

[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. Halper. College Publications, 2010.

[MRB18] A. S. Morcos, M. Raghu, and S. Bengio. "Insights on representational similarity in neural networks with canonical correlation". In: *Advances in neural information processing systems* (2018).

[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.

[MS67] J McNamee and F Stenger. "Construction of fully symmetric numerical integration formulas of fully symmetric numerical integration formulas". In: *Numer. Math.* 10.4 (Nov. 1967), pp. 327–344.

[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 (1996), pp. 6174–6185.

[MS99] C. Manning and H. Schuetze. *Foundations of statistical natural language processing*. MIT Press, 1999.

[MSA18] S. Makridakis, E. Spiliotis, and V. Assimakopoulos. "The M4 Competition: Results, findings, conclusion and way forward". In: *Int. J. Forecast.* 34.4 (2018), pp. 802–808.

[MT12] A. Mnih and Y. W. Teh. "A fast and simple algorithm for training neural probabilistic language models". In: *ICML*. 2012, pp. 419–426.

[MTM14] C. J. Maddison, D. Tarlow, and T. Minka. "A* Sampling". In: *NIPS*. 2014.

[MTS22] Y. Ma, D. Tsao, and H.-Y. Shum. "On the Principles of Parsimony and Self-Consistency for the Emergence of Intelligence". In: (July 2022). arXiv: 2207.04630 [cs.AI].

[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.

[Mua+20] K. Muandet, A. Mehrjou, S. K. Lee, and A. Raj. *Dual Instrumental Variable Regression*. 2020.

[Muk+18] S. Mukherjee, D. Shankar, A. Ghosh, N. Tathawadekar, P. Kompalli, S. Sarawagi, and K. Chaudhury. "ARMDN: Associative and Recurrent Mixture Density Networks for eRetail Demand Forecasting". In: (2018). arXiv: 1803.03800 [cs.LG].

[Mül+19a] T. Müller, B. McWilliams, F. Rousselle, M. Gross, and J. Novák. "Neural Importance Sampling". In: *SIGGRAPH*. 2019.

[Mül+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.

[Mun14] R. Munos. "From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning". In: *Foundations and Trends in Machine Learning* 7.1 (2014), pp. 1–129.

[Mun+16] R. Munos, T. Stepleton, A. Harutyunyan, and M. G. Bellemare. "Safe and Efficient Off-Policy Reinforcement Learning". In: *NIPS*. 2016, pp. 1046–1054.

[Mun57] J. Munkres. "Algorithms for the assignment and transportation problems". In: *Journal of the society for industrial and applied mathematics* 5.1 (1957), pp. 32–38.

[Mur00] K. Murphy. "Bayesian Map Learning in Dynamic Environments". In: *NIPS*. Vol. 12. 2000.

[Mur02] K. Murphy. "Dynamic Bayesian Networks: Representation, Inference and Learning". PhD thesis. Dept. Computer Science, UC Berkeley, 2002.

[Mur+19] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. "Definitions, methods, and applications in interpretable machine learning". In: *Proceedings of the National Academy of Sciences* 116.44 (2019), pp. 22071–22080.

[Mur22] K. P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[MW15] S. Morgan and C. Winship. *Counterfactuals and Causal Inference*. 2nd. Cambridge University Press, 2015.

[MWJ99] K. Murphy, Y. Weiss, and M. Jordan. "Loopy Belief Propagation for Approximate Inference: an Empirical Study". In: *UAI*. 1999.

[MYM18] J. Marino, Y. Yue, and S. Mandt. "Iterative Amortized Inference". In: *ICML*. 2018.

[Nac+17] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans. "Bridging the Gap Between Value and Policy Based Reinforcement Learning". In: *NIPS*. 2017, pp. 2772–2782.

[Nac+19a] O. Nachum, Y. Chow, B. Dai, and L. Li. "DualDICE: Behavior-agnostic Estimation of Discounted Stationary Distribution Corrections". In: *NeurIPS*. 2019, pp. 2315–2325.

[Nac+19b] O. Nachum, B. Dai, I. Kostrikov, Y. Chow, L. Li, and D. Schuurmans. *Algae: Policy Gradient from Arbitrary Experience*. CoRR abs/1912.02074. 2019.

[Nad+19] S. Naderi, K. He, R. Aghajani, S. Sclaroff, and P. Felzenszwalb. "Generalized Majorization-Minimization". In: *ICML*. 2019.

[Nae+18] C. A. Naesseth, S. W. Linderman, R. Ranganath, and D. M. Blei. "Variational Sequential Monte Carlo". In: *AISTATS*. 2018.

[Nal18] E. T. Nalisnick. "On Priors for Bayesian Neural Networks". PhD thesis. UC Irvine, 2018.

[Nal+19a] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan. "Do Deep Generative Models Know What They Don't Know?" In: *ICLR*. 2019.

[Nal+19b] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan. "Hybrid Models with Deep and Invertible Features". In: *ICML*. 2019, pp. 4723–4732.

[Nau04] J. Naudts. "Estimators, escort probabilities and ϕ-exponential families in statistical physics". In: *J. of Inequalities in Pure and Applied Mathematics* 5.4 (2004).

[NB05] M. Narasimhan and J. Bilmes. "A Submodular-Supermodular Procedure with Applications to Discriminative Structure Learning". In: *Uncertainty in Artificial Intelligence: Proceedings of the Twentieth Conference (UAI-2004)*. Edinburgh, Scotland: Morgan Kaufmann Publishers, 2005.

[NB06] M. Narasimhan and J. Bilmes. *Learning Graphical Models over partial k-trees*. Tech. rep. UWEETR-2006-0001. https://vannevar.ece.uw.edu/techsite/papers/refer/UWEETR-2006-0001.html. University of Washington, Department of Electrical Engineering, 2006.

[NBS18] B. Neyshabur, S. Bhojanapalli, and N. Srebro. "A PAC-Bayesian Approach to Spectrally-Normalized Margin Bounds for Neural Networks". In: *ICLR*. 2018.

[NCH15] M. Naeini, G. Cooper, and M. Hauskrecht. "Obtaining well calibrated probabilities using Bayesian binning". In: *AAAI*. 2015.

[NCL20] T. Nguyen, Z. Chen, and J. Lee. "Dataset Meta-Learning from Kernel Ridge-Regression". In: *International Conference on Learning Representations*. 2020.

[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: (2020). arXiv: 2001.01866 [cs.LG].

[ND21] A. Nichol and P. Dhariwal. "Improved denoising diffusion probabilistic models". In: *ICML*. 2021.

[NDL20] A. Nishimura, D. Dunson, and J. Lu. "Discontinuous Hamiltonian Monte Carlo for discrete parameters and discontinuous likelihoods". In: *Biometrika* (2020).

[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. 7–5–767.

[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.

[Nea12] R. C. Neath. "On Convergence Properties of the Monte Carlo EM Algorithm". In: *arXiv [math.ST]* (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.

[Nea96] R. Neal. *Bayesian learning for neural networks*. Springer, 1996.

[Nef+02] A. Nefian, 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". 2021.

[Nei+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.

[NF16] M. Noroozi and P. Favaro. "Unsupervised learning of visual representations by solving jigsaw puzzles". In: *European conference on computer vision*. Springer. 2016, pp. 69–84.

[NG01] D. Nilsson and J. Goldberger. "Sequentially finding the N-Best List in Hidden Markov Models". In: *IJCAI*. 2001, pp. 1280–1285.

[Ng+11] A. Ng et al. "Sparse autoencoder". In: *CS294A Lecture notes* 72.2011 (2011), pp. 1–19.

[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.

[Ngi+18] J. Ngiam, D. Peng, V. Vasudevan, S. Kornblith, Q. V. Le, and R. Pang. "Domain adaptive transfer learning with specialist models". In: *arXiv preprint arXiv:1811.07056* (2018).

[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+18] 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: (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.

[NH98a] 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.

[NH98b] 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. Springer Netherlands, 1998, pp. 355–368.

[NHLS19] E. Nalisnick, J. M. Hernández-Lobato, and P. Smyth. "Dropout as a Structured Shrinkage Prior". In: *ICML*. 2019.

[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.

[Nig+00] K. Nigam, A. K. Mccallum, S. Thrun, and T. Mitchell. "Text Classification from Labeled and Unlabeled Documents using EM". In: *MLJ* 39.2 (May 2000), pp. 103–134.

[Nij+19] E. Nijkamp, M. Hill, S.-C. Zhu, and Y. N. Wu. "Learning non-convergent non-persistent short-run MCMC toward energy-based model". In: *NIPS*. 2019, pp. 5232–5242.

[Nix+19] J. Nixon, M. Dusenberry, L. Zhang, G. Jerfel, and D. Tran. "Measuring Calibration in Deep Learning". In: (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.

[NLS15] C. A. Naesseth, F. Lindsten, and T. B. Schön. "Nested Sequential Monte Carlo Methods". In: *ICML*. 2015.

[NLS19] 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.

[NNP19] 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*. 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 (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.

[NP03] W. K. Newey and J. L. Powell. "Instrumental variable estimation of nonparametric models". In: *Econometrica* 71.5 (2003), pp. 1565–1578.

[NR00a] A. Ng and S. Russell. "Algorithms for inverse reinforcement learning". In: *ICML*. 2000.

[NR00b] A. Y. Ng and S. Russell. "Algorithms for Inverse Reinforcement Learning". In: *in Proc. 17th International Conf. on Machine Learning*. Citeseer. 2000.

[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.

[NS17] E. Nalisnick and P. Smyth. "Variational Reference Priors". In: *ICLR Workshop*. 2017.

[NS18] E. Nalisnick and P. Smyth. "Learning Priors for Invariance". In: *AISTATS*. 2018.

[NW06] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2006.

[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/108/2/299/37938939/asaa076.pdf.

[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.

[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.

[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.

[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 (2010), pp. 5847–5861.

[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.

[NYC15] A. Nguyen, J. Yosinski, and J. Clune. "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images". In: *CVPR*. 2015.

[OA09] M. Opper and C. Archambeau. "The variational Gaussian approximation revisited". en. In: *Neural Comput.* 21.3 (2009), pp. 786–792.

[OAC18] I. Osband, J. Aslanides, and A. Cassirer. "Randomized prior functions for deep reinforcement learning". In: *NIPS*. 2018.

[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*. 2019.

[OCM21] L. A. Ortega, R. Cabañas, and A. R. Masegosa. "Diversity and Generalization in Neural Network Ensembles". In: (2021). arXiv: 2110.13786 [cs.LG].

[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.

[OED21] J. Ortiz, T. Evans, and A. J. Davison. "A visual introduction to Gaussian Belief Propagation". In: *arXiv preprint arXiv:2107.02308* (2021).

[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.

[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.

[OKK16] A. Van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. "Pixel Recurrent Neural Networks". In: *ICML*. 2016.

[Oll+17] Y. Ollivier, L. Arnold, A. Auger, and N. Hansen. "Information-Geometric Optimization Algorithms: A Unifying Picture via Invariance Principles". In: *JMLR* 18 (2017), pp. 1–65.

[Oll18] Y. Ollivier. "Online natural gradient as a Kalman filter". en. In: *Electron. J. Stat.* 12.2 (2018), pp. 2930–2961.

[OLV18a] A. van den Oord, Y. Li, and O. Vinyals. "Representation Learning with Contrastive Predictive Coding". In: (2018). arXiv: 1807.03748 [cs.LG].

[OLV18b] A. v. d. Oord, Y. Li, and O. Vinyals. "Representation learning with contrastive predictive coding". In: *arXiv preprint arXiv:1807.03748* (2018).

[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 (2017).

[O'N09] 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 (2018), pp. 465–478.

[Oor+16a] 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: (2016). arXiv: 1609.03499 [cs.SD].

[Oor+16b] 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].

[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. PMLR, 2018, pp. 3918–3926.

[Oor+19] A. van den Oord, B. Poole, O. Vinyals, and A. Razavi. "Fixing Posterior Collapse 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.

[Ope] OpenAI. *ChatGPT: Optimizing Language Models for Dialogue*. Blog.

[Oqu+14] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. "Learning and transferring mid-level image representations using convolutional neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1717–1724.

[OR20] A. Owen and D. Rudolf. "A strong law of large numbers for scrambled net integration". In: (2020). arXiv: 2002.07859 [math.NA].

[Ora19] F. Orabona. "A Modern Introduction to Online Learning". In: *arXiv* (Dec. 2019). arXiv: 1912.13213 [cs.LG].

[Ore+20] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting". In: *ICLR*. 2020.

[ORW21] S. W. Ober, C. E. Rasmussen, and M. van der Wilk. "The Promises and Pitfalls of Deep Kernel Learning". In: *ICML*. Feb. 2021.

[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. Matsuoka. "Large-Scale Distributed Second-Order Optimization Using Kronecker-Factored Approximate Curvature for Deep Convolutional Neural Networks". In: *CVPR*. 2019.

[Osb16] I. Osband. "Risk versus Uncertainty in Deep Learning: Bayes, Bootstrap and the Dangers of Dropout". In: *NIPS workshop on Bayesian deep learning*. 2016.

[Osb+21] I. Osband, Z. Wen, S. M. Asghari, V. Dwaracherla, B. Hao, M. Ibrahimi, D. Lawson, X. Lu, B. O'Donoghue, and B. Van Roy. "The Neural Testbed: Evaluating Predictive Distributions". In: (2021). arXiv: 2110.04629 [cs.LG].

[Ose11] I. V. Oseledets. "Tensor-Train Decomposition". In: *SIAM J. Sci. Comput.* 33.5 (2011), pp. 2295–2317.

[OT05] A. B. Owen and S. D. Tribble. "A quasi-Monte Carlo Metropolis algorithm". en. In: *PNAS* 102.25 (2005), pp. 8844–8849.

[Ouy+22] L. Ouyang et al. "Training language models to follow instructions with human feedback". In: (Mar. 2022). arXiv: 2203.02155 [cs.CL].

[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]* (2017).

[Oxl11] 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". en. In: *ICML*. 2014, pp. 1152–1160.

[Pai] *Explainable AI in Practice Falls Short of Transparency Goals*. https://partnershiponai.org/xai-in-practice/. Accessed: 2021-11-23.

[Pai+14] T. L. Paine, P. Khorrami, W. Han, and T. S. Huang. "An analysis of unsupervised pre-training in light of recent advances". In: *arXiv preprint arXiv:1412.6597* (2014).

[Pan+10] L. Paninski, 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". en. In: *J. Comput. Neurosci.* 29.1-2 (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 (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: *Transactions on Machine Learning Research* (Jan. 2022).

[Pap16] P. Papastamoulis. "label.switching: An R Package for Dealing with the Label Switching Problem in MCMC Outputs". en. In: *J. Stat. Softw.* 69 (Feb. 2016), pp. 1–24.

[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: (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 (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.

[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 (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. Association for Computing Machinery, 2021, 13–23.

[Pat+16] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. "Context encoders: Feature learning by inpainting". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2536–2544.

[Pat+22] Z. B. Patel, P. Purohit, H. M. Patel, S. Sahni, and N. Batra. "Accurate and Scalable Gaussian Processes for Fine-Grained Air Quality Inference". en. In: *AAAI* 36.11 (June 2022), pp. 12080–12088.

[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 (2012), pp. 159–168.

[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". en. In: *IEEE Trans. Image Process.* 25.1 (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 (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*. 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 (1994), pp. 147–160.

[Peh+20] R. Peharz, S. Lang, A. Vergari, K. Stelzner, A. Molina, M. Trapp, G. Van den Broeck, K. Kersting, and Z. Ghahramani. "Einsum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits". In: (2020). arXiv: 2004.06231 [cs.LG].

[Pel05] M. Pelikan. *Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms*. en. Softcover reprint of hardcover 1st ed. 2005 edition. Springer, 2005.

[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: (2021). arXiv: 2112.12841 [stat.AP].

[Pey20] G. Peyre. "Course notes on Optimization for Machine Learning". 2020.

[Pez+21] M. Pezeshki, S.-O. Kaba, Y. Bengio, A. Courville, D. Precup, and G. Lajoie. "Gradient Starvation: A Learning Proclivity in Neural Networks". In: *NIPS*. 2021.

[PF03] G. V. Puskorius and L. A. Feldkamp. "Parameter-based Kalman filter training: Theory and implementation". In: *Kalman Filtering and Neural Networks*. Ed. by S. Haykin. 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. 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. M. Yeung, B.-L. Yeo, and C. A. Bouman. Vol. 3656. International Society for Optics and Photonics. SPIE, 1998, pp. 552 –560.

[PGCP00] M Pelikan, D. E. Goldberg, and E Cantú-Paz. "Linkage problem, distribution estimation, and Bayesian networks". en. In: *Evol. Comput.* 8.3 (2000), pp. 311–340.

[PGJ16] J. Pearl, M. Glymour, and N. Jewell. *Causal inference in statistics: a primer*. Wiley, 2016.

[PH22] M. Phuong and M. Hutter. "Formal Algorithms for Transformers". In: (July 2022). arXiv: 2207.09238 [cs.LG].

[PHL12] M. Pelikan, M. Hausschild, and F. Lobo. *Introduction to estimation of distribution algorithms*. Tech. rep. U. Missouri, 2012.

[PHR18] E. Petersen, C. Hoffmann, and P. Rostalski. "On Approximate Nonlinear Gaussian Message Passing On Factor Graphs". In: *IEEE Statistical Signal Processing Workshop (SSP)*. 2018.

[Phu+18] M. Phuong, M. Welling, N. Kushman, R. Tomioka, and S. 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.

[PJD21] Y. Petetin, Y. Janati, and F. Desbouvries. "Structured Variational Bayesian Inference for Gaussian State-Space Models With Regime Switching". In: *IEEE Signal Process. Lett.* 28 (2021), pp. 1953–1957.

[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, 2017.

[PKP21] A. Plaat, W. Kosters, and M. Preuss. "High-Accuracy Model-Based Reinforcement Learning, a Survey". In: (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.

[Pla18] E. Plaut. "From Principal Subspaces to Principal Components with Linear Autoencoders". In: *ArXiv* abs/1804.10253 (2018).

[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.

[Plu+20] G. Plumb, 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).

[PM18a] N. Papernot and P. McDaniel. *Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning*. 2018. arXiv: 1803.04765 [cs.LG].

[PM18b] J. Pearl and D. Mackenzie. *The book of why: the new science of cause and effect*. 2018.

[PMT18] G. Plumb, D. Molitor, and A. Talwalkar. "Supervised Local Modeling for Interpretability". In: *CoRR* abs/1807.02910 (2018). arXiv: 1807.02910.

[Pol+19] A. A. Pol, V. Berger, G. Cerminara, C. Germain, and M. Pierini. "Anomaly Detection With Conditional Variational Autoencoders". In: *IEEE International Conference on Machine Learning and Applications*. 2019.

[Pom89] D. Pomerleau. "ALVINN: An Autonomous Land Vehicle in a Neural Network". In: *NIPS*. 1989, pp. 305–313.

[Poo+12] D. Poole, D. Buchman, S. Natarajan, and K. Kersting. "Aggregation and Population Growth: The Relational Logistic Regression and Markov Logic Cases". In: *Statistical Relational AI workshop*. 2012.

[Poo+19a] B. Poole, S. Ozair, A. van den Oord, A. A. Alemi, and G. Tucker. "On Variational Bounds of Mutual Information". In: *ICML*. 2019.

[Poo+19b] B. Poole, S. Ozair, A. van den Oord, A. A. Alemi, and G. Tucker. "On variational lower bounds of mutual information". In: *ICML*. 2019.

[Pou04] M. Pourahmadi. *Simultaneous Modelling of Covariance Matrices: GLM, Bayesian and Nonparametric Perspectives*. Tech. rep. Northern Illinois University, 2004.

[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.

[PPC09] G. Petris, S. Petrone, and P. Campagnoli. *Dynamic linear models with R*. Springer, 2009.

[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.

[PPM17] G. Papamakarios, T. Pavlakou, and I. Murray. "Masked Autoregressive Flow for Density Estimation". In: *NIPS*. 2017.

[PPR22] B. Paria, B. Pòczos, and P. Ravikumar. "Be greedy – a simple algorithm for blackbox optimization using neural networks". In: *ICML Workshop on Adaptive Experimental Design and Active Learning in the Real World*. 2022.

[PPS18] T. Pierrot, N. Perrin, and O. Sigaud. "First-order and second-order variants of the gradient descent in a unified framework". In: (2018). arXiv: 1810.08102 [cs.LG].

[PR03] O. Papaspiliopoulos and G. O. Roberts. "Non-Centered Parameterisations for Hierarchical Models and Data Augmentation". In: *Bayesian Statistics* 7 (2003), pp. 307–326.

[Pra+18] S. Prabhumoye, 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.

[Pre05] S. J. Press. *Applied multivariate analysis, using Bayesian and frequentist methods of inference*. Second edition. Dover, 2005.

[Pre+17a] V. Premachandran, D. Tarlow, A. L. Yuille, and D. Batra. "Empirical Minimum Bayes Risk Prediction". en. In: *IEEE PAMI* 39.1 (Jan. 2017), pp. 75–86.

[Pre+17b] 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).

[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.

[PRG17] M. Probst, F. Rothlauf, and J. Grahl. "Scalability of using Restricted Boltzmann Machines for combinatorial optimization". In: *Eur. J. Oper. Res.* 256.2 (2017), pp. 368–383.

[Pri58] R. Price. "A useful theorem for nonlinear devices having Gaussian inputs". In: *IRE Trans. Info. Theory* 4.2 (1958), pp. 69–72.

[PS07] J. Peters and S. Schaal. "Reinforcement Learning by Reward-Weighted Regression for Operational Space Control". In: *ICML*. 2007, pp. 745–750.

[PS08a] B. A. Pearlmutter and J. M. Siskind. "Reverse-Mode AD in a Functional Framework: Lambda the Ultimate Backpropagator". In: *ACM Trans. Program. Lang. Syst.* 30.2 (2008).

[PS08b] J. Peters and S. Schaal. "Reinforcement Learning of Motor Skills with Policy Gradients". In: *Neural Networks* 21.4 (2008), pp. 682–697.

[PS12] N. G. Polson and J. G. Scott. "On the Half-Cauchy Prior for a Global Scale Parameter". en. In: *Bayesian Anal.* 7.4 (2012), pp. 887–902.

[PS17] N. G. Polson and V. Sokolov. "Deep Learning: A Bayesian Perspective". en. In: *Bayesian Anal.* 12.4 (2017), pp. 1275–1304.

[PSCP06] M. Pelikan, K. Sastry, and E. Cantú-Paz. *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications (Studies in Computational Intelligence)*. Springer-Verlag New York, Inc., 2006.

[PSD00] J. K. Pritchard, M Stephens, and P Donnelly. "Inference of population structure using multilocus genotype data". In: *Genetics* 155.2 (2000), pp. 945–959.

[PSDG14] B. Poole, J. Sohl-Dickstein, and S. Ganguli. "Analyzing noise in autoencoders and deep networks". In: *arXiv preprint arXiv:1406.1831* (2014).

[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. 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. 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.

[PW05] S. Parise and M. Welling. "Learning in Markov Random Fields: An Empirical Study". In: *Joint Statistical Meeting*. 2005.

[PX22] W. Peebles and S. Xie. "Scalable Diffusion Models with Transformers". In: (Dec. 2022). arXiv: 2212.09748 [cs.CV].

[PY10] G. Papandreou and A. L. Yuille. "Gaussian sampling by local perturbations". In: *NIPS*. 2010.

[PY11] G Papandreou and A. L. Yuille. "Perturb-and-MAP random fields: Using discrete optimization to learn and sample from energy models". In: *ICCV*. Nov. 2011, pp. 193–200.

[PY14] G. Papandreou and A. Yuille. "Perturb-and-MAP Random Fields: Reducing Random Sampling to Optimization, with Applications in Computer Vision". In: *Advanced Structured Prediction*. Ed. by S. Nowozin, P. Gehler, J. Jancsary, C. Lampert. MIT Press, 2014.

[QC+06] J. Quiñonero-Candela, C. E. Rasmussen, F. Sinz, O. Bousquet, 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. Quinonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, eds. *Dataset Shift in Machine Learning*. en. The MIT Press, 2008.

[QCR05] J. Quinonero-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).

[QRJN18] J. Qiu, S Rao Jammalamadaka, and N. Ning. "Multivariate Bayesian Structural Time Series Model". In: *JMLR* 19.68 (2018), pp. 1–33.

[Qu+21] H. Qu, H. Rahmani, L. Xu, B. Williams, and J. Liu. "Recent Advances of Continual Learning in Computer Vision: An Overview". In: (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: *ICLR*. 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.

[Rad+21] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. "Learning transferable visual models from natural language supervision". In: *arXiv preprint arXiv:2103.00020* (2021).

[Raf+20a] 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: *JMLR* (2020).

[Raf+20b] C. Raffel, N. M. 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: *ArXiv* abs/1910.10683 (2020).

[Raf22] E. Raff. *Inside Deep Learning: Math, Algorithms, Models*. en. Annotated edition. Manning, May 2022.

[RAG04] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House Radar Library, 2004.

[Rag+17] M. Raghu, J. Gilmer, J. Yosinski, and J. Sohl-Dickstein. "Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6076–6085.

[Rag+19] M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio. "Transfusion: Understanding transfer learning for medical imaging". In: *NIPS*. 2019, pp. 3347–3357.

[Rag+21] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy. "Do Vision Transformers See Like Convolutional Neural Networks?" In: *NIPS*. 2021.

[Rai+18a] 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.

[Rai+18b] M. Raitoharju, L. Svensson, Á. F. García-Fernández, and R. Piché. "Damped Posterior Linearization Filter". In: *IEEE Signal Process. Lett.* 25.4 (2018).

[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.

[Rai68] H. Raiffa. *Decision Analysis*. Addison Wesley, 1968.

[Rak+08] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. "SimpleMKL". In: *JMLR* 9 (2008), pp. 2491–2521.

[Ram+21a] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. "Zero-Shot Text-to-Image Generation". In: (2021). arXiv: 2102.12092 [cs.CV].

[Ram+21b] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. "Zero-shot text-to-image generation". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8821–8831.

[Ram+22] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. "Hierarchical Text-Conditional Image Generation with CLIP Latents". In: (Apr. 2022). arXiv: 2204.06125 [cs.CV].

[Ran+06] M. Ranzato, C. S. Poultney, S. Chopra, and Y. LeCun. "Efficient Learning of Sparse Representations with an Energy-Based Model". In: *NIPS*. 2006.

[Ran16] R. Ranganath. "Hierarchical Variational Models". In: *ICML*. 2016.

[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.

[Rao10] A. V. Rao. "A Survey of Numerical Methods for Optimal Control". In: *Adv. Astronaut. Sci.* 135.1 (2010).

[Rao99] R. P. Rao. "An optimal estimation approach to visual perception and learning". en. In: *Vision Res.* 39.11 (1999), pp. 1963–1989.

[Ras+15] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko. *Semi-Supervised Learning with Ladder Networks*. 2015. arXiv: 1507.02672 [cs.NE].

[Rat+09] M. Rattray, 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.

[Rav+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.

[RB16] G. P. Rigby BR. "The Efficacy of Equine-Assisted Activities and Therapies on Improving Physical Function." In: *J Altern Complement Med.* (2016).

[RBB18a] H. Ritter, A. Botev, and D. Barber. "A Scalable Laplace Approximation for Neural Networks". In: *ICLR*. 2018.

[RBB18b] H. Ritter, A. Botev, and D. Barber. "Online Structured Laplace Approximations for Overcoming Catastrophic Forgetting". In: *NIPS*. 2018, pp. 3738–3748.

[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.

[RBS84] J. Ramsay, J. ten Berge, and G. Styan. "Matrix correlation". In: *Psychometrika* 49.3 (1984), pp. 403–423.

[RC04] C. Robert and G. Casella. *Monte Carlo Statisical Methods*. 2nd edition. Springer, 2004.

[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.

[RD06] M. Richardson and P. Domingos. "Markov logic networks". In: *Machine Learning* 62 (2006), pp. 107–136.

[RDV18] 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.

[RE76] P Robert and Y Escoufier. "A unifying tool for linear multivariate statistical methods: The RV- coefficient". In: *J. R. Stat. Soc. Ser. C Appl. Stat.* 25.3 (1976), p. 257.

[Rea+19] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. "Regularized Evolution for Image Classifier Architecture Search". In: *AAAI*. 2019.

[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.

[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].

[Rei+10] J. Reisinger, A. Waters, B. Silverthorn, and R. Mooney. "Spherical topic models". In: *ICML*. 2010.

[Rei13] S. Reich. "A Nonparametric Ensemble Transform Method for Bayesian Inference". In: *SIAM J. Sci. Comput.* 35.4 (2013), A2013–A2024.

[Rei16] P. C. Reiss. "Just How Sensitive are Instrumental Variable Estimates?" In: *Foundations and Trends in Accounting* 10.2-4 (2016).

[Rei+22] P. Reizinger, L. Gresele, J. Brady, J. von Kügelgen, D. Zietlow, B. Scholkopf, G. Martius, W. Brendel, and M. Besserve. "Embrace the Gap: VAEs Perform Independent Mechanism Analysis". In: 2022.

[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.

[Ren+21] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang. "A Survey of

Deep Active Learning". In: *ACM Comput. Surv.* 54.9 (Oct. 2021), pp. 1–40.

[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.

[Ren+94] S Renals, N Morgan, H Bourlard, M Cohen, and H Franco. "Connectionist probability estimators in HMM speech recognition". In: *IEEE Trans. Audio Speech Lang. Processing* 2.1 (Jan. 1994), pp. 161–174.

[RFB15] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *MICCAI (Intl. Conf. on Medical Image Computing and Computer Assisted Interventions).* 2015.

[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*. 2017, pp. 4686–4690.

[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.

[RGB14] R. Ranganath, S. Gerrish, and D. M. Blei. "Black Box Variational Inference". In: *AISTATS*. 2014.

[RGL19] S. Rabanser, S. Günnemann, and Z. C. Lipton. "Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift". In: *NIPS*. 2019.

[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.

[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).

[RHG16] D. Ritchie, P. Horsfall, and N. D. Goodman. "Deep Amortized Inference for Probabilistic Programs". In: (2016). arXiv: 1610.05735 [cs.AI].

[RHK17] S. Remes, M. Heinonen, and S. Kaski. "Non-Stationary Spectral Kernels". In: *NIPS*. May 2017.

[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.

[RHW86b] D. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536.

[Ric03] T. Richardson. "Markov properties for acyclic directed mixed graphs". In: *Scandinavian J. of Statistics* 30 (2003), pp. 145–157.

[Ric95] J. Rice. *Mathematical statistics and data analysis*. 2nd edition. Duxbury, 1995.

[Rif+11] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. "Contractive auto-encoders: Explicit invariance during feature extraction". In: *Icml*. 2011.

[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.

[Riv87] R. L. Rivest. "Learning decision lists". In: *Machine learning* 2.3 (1987), pp. 229–246.

[RK04] R. Rubinstein and D. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. Springer-Verlag, 2004.

[RL17] S. Ravi and H. Larochelle. "Optimization as a Model for Few-Shot Learning". In: *ICLR*. 2017.

[RM15] D. J. Rezende and S. Mohamed. "Variational Inference with Normalizing Flows". In: *ICML*. 2015.

[RM22] H. Rahimian and S. Mehrotra. "Distributionally Robust Optimization: A Review". In: *Open Journal of Mathematical Optimization* 3.4 (2022).

[RMB08] N. L. Roux, P.-A. Manzagol, and Y. Bengio. "Topmoumoute Online Natural Gradient Algorithm". In: *NIPS*. 2008, pp. 849–856.

[RMC09] 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.

[RMC15] A. Radford, L. Metz, and S. Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *arXiv* (2015).

[RMC16a] A. Radford, L. Metz, and S. Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *ICLR*. 2016.

[RMC16b] A. Radford, L. Metz, and S. Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *CoRR* abs/1511.06434 (2016).

[RMK21] G. Roeder, L. Metz, and D. P. Kingma. "On Linear Identifiability of Learned Representations". In: *ICML*. 2021.

[RMW14a] D. Rezende, S. Mohamed, and D. Wierstra. "Stochastic backpropagation and approximate inference in deep generative models". In: *ICML*. 2014.

[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. PMLR, 2014, pp. 1278–1286.

[RN02] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. 2nd edition. Prentice Hall, 2002.

[RN10] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. 3rd edition. Prentice Hall, 2010.

[RN19] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. 4th edition. Prentice Hall, 2019.

[RN94] G. A. Rummery and M Niranjan. *On-Line Q-Learning Using Connectionist Systems*. Tech. rep. Cambridge Univ. Engineering Dept., 1994.

[RN95] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[RNA22] L. Regenwetter, A. H. Nobari, and F. Ahmed. "Deep Generative Models in Engineering Design: A Review". In: *J. Mech. Des.* (2022).

[Rob07] C. P. Robert. *The Bayesian Choice: From Decision-Theoretic Foundations to Computational

Implementation. en. 2nd edition. Springer Verlag, New York, 2007.

[Rob+13] S Roberts, M Osborne, M Ebden, S Reece, N Gibson, and S Aigrain. “Gaussian processes for time-series modelling”. en. In: *Philos. Trans. A Math. Phys. Eng. Sci.* 371.1984 (2013), p. 20110550.

[Rob+18] C. P. Robert, V. Elvira, N. Tawn, and C. Wu. “Accelerating MCMC Algorithms”. In: (2018). arXiv: 1804.02719 [stat.CO].

[Rob+21] J. Robinson, C.-Y. Chuang, S. Sra, and S. Jegelka. “Contrastive Learning with Hard Negative Samples”. In: *ArXiv* abs/2010.04592 (2021).

[Rob63] L. G. Roberts. “Machine Perception of Three-Dimensional Solids”. In: *Outstanding Dissertations in the Computer Sciences*. 1963.

[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.

[Rob95a] C. Robert. “Simulation of truncated normal distributions”. In: *Statistics and computing* 5 (1995), pp. 121–125.

[Rob95b] A. Robins. “Catastrophic Forgetting, Rehearsal and Pseudorehearsal”. In: *Conn. Sci.* 7.2 (1995), pp. 123–146.

[Rod14] J. Rodu. “Spectral estimation of hidden Markov models”. PhD thesis. U. Penn, 2014.

[RÖG13] M. Roth, E. Özkan, and F. Gustafsson. “A Student’s t filter for heavy tailed process and measurement noise”. In: *ICASSP*. 2013, pp. 5770–5774.

[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*. 2021.

[Rom+22] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. “High-Resolution Image Synthesis with Latent Diffusion Models”. In: *CVPR*. 2022.

[Ros10] P. Rosenbaum. *Design of Observational Studies*. 2010.

[Ros+21] M. Rosca, Y. Wu, B. Dherin, and D. G. Barrett. “Discretization Drift in Two-Player Games”. In: (2021).

[Ros+22] C. Rosato, L. Devlin, V. Beraud, P. Horridge, T. B. Schön, and S. Maskell. “Efficient Learning of the Parameters of Non-Linear Models Using Differentiable Resampling in Particle Filters”. In: *IEEE Trans. Signal Process.* 70 (2022), pp. 3676–3692.

[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. 2022.

[Rot+17] M. Roth, G. Hendeby, C. Fritsche, and F. Gustafsson. “The Ensemble Kalman filter: a signal processing perspective”. In: *EURASIP J. Adv. Signal Processing* 2017.1 (2017), p. 56.

[Rot+18] W. Roth, R. Peharz, S. Tschiatschek, and F. Pernkopf. “Hybrid generative-discriminative training of Gaussian mixture models”. In: *Pattern Recognit. Lett.* 112 (Sept. 2018), pp. 131–137.

[Rot96] D. Roth. “On the hardness of approximate reasoning”. In: *Artificial Intelligence* 82.1-2 (1996), pp. 273–302.

[ROV19] A. Razavi, A. van den Oord, and O. Vinyals. “Generating diverse high resolution images with VA-VAE-2”. In: *NIPS*. 2019.

[Row97] S. Roweis. “EM algorithms for PCA and SPCA”. In: *NIPS*. 1997.

[Roy+21] N. Roy et al. “From Machine Learning to Robotics: Challenges and Opportunities for Embodied Intelligence”. In: (Oct. 2021). arXiv: 2110.15245 [cs.RO].

[RPC19] Y. Romano, E. Patterson, and E. J. Candès. “Conformalized Quantile Regression”. In: *NIPS*. 2019.

[RPH21] A. Robey, G. J. Pappas, and H. Hassani. *Model-Based Domain Generalization*. 2021. arXiv: 2102.11436 [stat.ML].

[RR01a] 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.

[RR01b] G. Roberts and J. Rosenthal. “Optimal scaling for various Metropolis-Hastings algorithms”. In: *Statistical Science* 16 (2001), pp. 351–367.

[RR08] A. Rahimi and B. Recht. “Random Features for Large-Scale Kernel Machines”. In: *NIPS*. Curran Associates, Inc., 2008, pp. 1177–1184.

[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.

[RR11] T. S. Richardson and J. M. Robins. “Single World Intervention Graphs: A Primer”. In: *Second UAI workshop on causal structure learning*. 2011.

[RR13] T. S. Richardson and J. M. Robins. “Single World Intervention Graphs (SWIGs): A Unification of the Counterfactual and Graphical Approaches to Causality”. 2013.

[RR14] D. Russo and B. V. Roy. “Learning to Optimize via Posterior Sampling”. In: *Math. Oper. Res.* 39.4 (2014), pp. 1221–1243.

[RR83] P. R. Rosenbaum and D. B. Rubin. “Assessing Sensitivity to an Unobserved Binary Covariate in an Observational Study with Binary Outcome”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 45.2 (1983), pp. 212–218.

[RRR21] E. Rosenfeld, P. Ravikumar, and A. Risteski. “The Risks of Invariant Risk Minimization”. In: *ICML*. 2021.

[RRS00] J. M. Robins, A. Rotnitzky, and D. O. Scharfstein. “Sensitivity analysis for selection bias and unmeasured confounding in missing data and causal inference models”. In: *Statistical models in epidemiology, the environment, and clinical trials*. Springer, 2000, pp. 1–94.

[RS07] M. Raphan and E. P. Simoncelli. “Learning to be Bayesian without supervision”. In: *Advances in neural information processing systems*. 2007, pp. 1145–1152.

[RS11] M. Raphan and E. P. Simoncelli. “Least squares estimation without priors or supervision”. In: *Neural computation* 23.2 (2011), pp. 374–420.

[RS20] A. Rotnitzky and E. Smucler. “Efficient Adjustment Sets for Population Average Causal Treatment

Effect Estimation in Graphical Models." In: *J. Mach. Learn. Res.* 21 (2020), pp. 188–1.

[RS97a] G. O. Roberts and S. K. Sahu. "Updating Schemes, Correlation Structure, Blocking and Parameterization for the Gibbs Sampler". In: *J. of Royal Stat. Soc. Series B* 59.2 (1997), pp. 291–317.

[RS97b] G. O. Roberts and S. K. Sahu. "Updating schemes, correlation structure, blocking and parameterization for the Gibbs sampler". In: *J. of Royal Stat. Soc. Series B* 59.2 (1997), pp. 291–317.

[RSC20] Y. Romano, M. Sesia, and E. J. Candès. "Classification with Valid and Adaptive Coverage". In: *NIPS*. 2020.

[RSG16a] M. T. Ribeiro, S. Singh, and C. Guestrin. "" Why should i trust you?" Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[RSG16b] M. T. Ribeiro, S. Singh, and C. Guestrin. "Model-agnostic interpretability of machine learning". In: *arXiv preprint arXiv:1606.05386* (2016).

[RSG17] S. Rabanser, O. Shchur, and S. Günnemann. "Introduction to Tensor Decompositions and their Applications in Machine Learning". In: (2017). arXiv: 1711.10781 [stat.ML].

[RT16] S. Reid and R. Tibshirani. "Sparse regression and marginal testing using cluster prototypes". In: *Biostatistics* 17.2 (2016), pp. 364–376.

[RT82] D. B. Rubin and D. T. Thayer. "EM algorithms for ML factor analysis". In: *Psychometrika* 47.1 (1982), pp. 69–76.

[RT96] G. O. Roberts and R. L. Tweedie. "Exponential convergence of Langevin distributions and their discrete approximations". en. In: *Bernoulli* 2.4 (1996), pp. 341–363.

[RTS18] C. Riquelme, G. Tucker, and J. Snoek. "Deep Bayesian Bandits Showdown: An Empirical Comparison of Bayesian Deep Networks for Thompson Sampling". In: *ICLR*. 2018.

[RTS65] H. E. Rauch, F. Tung, and C. T. Striebel. "Maximum likelihood estimates of linear dynamic systems". In: *AIAA Journal* 3.8 (1965), pp. 1445–1450.

[Rub+20] Y. Rubanova, D. Dohan, K. Swersky, and K. Murphy. "Amortized Bayesian Optimization over Discrete Spaces". In: *UAI*. 2020.

[Rub74] D. B. Rubin. "Estimating causal effects of treatments in randomized and nonrandomized studies". In: *J. Educ. Psychol.* 66.5 (1974), pp. 688–701.

[Rub76] D. B. Rubin. "Inference and Missing Data". In: *Biometrika* 63.3 (1976), pp. 581–592.

[Rub84] D. B. Rubin. "Bayesianly Justifiable and Relevant Frequency Calculations for the Applied Statistician". In: *Ann. Stat.* 12.4 (1984), pp. 1151–1172.

[Rub97] R. Y. Rubinstein. "Optimization of computer simulation models with rare events". In: *Eur. J. Oper. Res.* 99.1 (1997), pp. 89–112.

[Rud19] C. Rudin. *Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead*. 2019. arXiv: 1811.10154 [stat.ML].

[Ruf+21] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller. "A Unifying Review of Deep and Shallow Anomaly Detection". In: *Proc. IEEE* 109.5 (2021), pp. 756–795.

[Rus15] S. Russell. "Unifying Logic and Probability". In: *Commun. ACM* 58.7 (2015), pp. 88–97.

[Rus+16] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. "Progressive Neural Networks". In: (2016). arXiv: 1606.04671 [cs.LG].

[Rus+18] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen. "A Tutorial on Thompson Sampling". In: *Foundations and Trends in Machine Learning* 11.1 (2018), pp. 1–96.

[Rus+95] S. Russell, J. Binder, D. Koller, and K. Kanazawa. "Local learning in probabilistic networks with hidden variables". In: *IJCAI*. 1995.

[RV19] S. Ravuri and O. Vinyals. "Classification accuracy score for conditional generative models". In: *Advances in Neural Information Processing Systems*. 2019, pp. 12268–12279.

[RW06] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[RW11] M. D. Reid and R. C. Williamson. "Information, Divergence and Risk for Binary Experiments". In: *Journal of Machine Learning Research* 12.3 (2011).

[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.

[RW18] E. Richardson and Y. Weiss. "On GANs and GMMs". In: *NIPS*. 2018.

[RWD17] G. Roeder, Y. Wu, and D. Duvenaud. "Sticking the Landing: An Asymptotically Zero-Variance Gradient Estimator for Variational Inference". In: *NIPS*. 2017.

[RY21] D. Roberts and S. Yaida. *The Principles of Deep Learning Theory: An Effective Theory Approach to Understanding Neural Network*. 2021.

[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.

[Ryu+20] M. Ryu, Y. Chow, R. Anderson, C. Tjandraatmadja, and C. Boutilier. "CAQL: Continuous Action Q-Learning". In: *ICLR*. 2020.

[RZL17] P. Ramachandran, B. Zoph, and Q. V. Le. "Searching for Activation Functions". In: (2017). arXiv: 1710.05941 [cs.NE].

[SA19] F. Schafer and A. Anandkumar. "Competitive gradient descent". In: *NIPS*. 2019, pp. 7625–7635.

[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).

[SAC17] J. Schulman, P. Abbeel, and X. Chen. *Equivalence Between Policy Gradients and Soft Q-Learning*. arXiv:1704.06440. 2017.

[Sag+20] S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang. "Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization". In: *ICLR*. 2020.

[Sah+22a] C. Saharia, W. Chan, H. Chang, C. A. Lee, J. Ho, T. Salimans, D. J. Fleet, and M. Norouzi. "Palette: Image-to-Image Diffusion Models". In: *SIGGRAPH* (Nov. 2022). arXiv: 2111.05826 [cs.CV].

[Sah+22b] C. Saharia et al. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding". In: (May 2022). arXiv: 2205.11487 [cs.CV].

[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.

[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.

[Sal16] T. Salimans. "A Structured Variational Autoencoder for Learning Deep Hierarchies of Sparse Features". In: (2016). arXiv: 1602.08734 [stat.ML].

[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].

[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.

[Sal+17b] T. Salimans, J. Ho, X. Chen, and I. Sutskever. "Evolution Strategies as a Scalable Alternative to Reinforcement Learning". In: (2017). arXiv: 1703.03864 [stat.ML].

[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.

[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.

[Sal+19b] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski. "DeepAR: Probabilistic forecasting with autoregressive recurrent networks". In: *International Journal of Forecasting* (2019).

[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).

[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: (2021). arXiv: 2110.14051 [cs.CV].

[Sal+22] 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: *Transaction on Machine Learning* (2022).

[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.

[San17] R. Santana. "Gray-box optimization and factorized distribution algorithms: where two worlds collide". In: (2017). arXiv: 1707.03093 [cs.NE].

[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.

[Sar08] S. Sarkka. "Unscented Rauch–Tung–Striebel Smoother". In: *IEEE Trans. Automat. Contr.* 53.3 (Apr. 2008), pp. 845–849.

[Sar13] S. Sarkka. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

[Sar18] H. Sarin. "Playing a game of GANstruction". In: *The Gradient* (2018).

[Say+19] R. Sayres, S. Xu, T Saensuksopa, 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.

[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.

[SB18] R. Sutton and A. Barto. *Reinforcement learning: an introduction (2nd edn)*. MIT Press, 2018.

[SBG07] S. Siddiqi, B. Boots, and G. Gordon. "A constraint generation approach to learning stable linear dynamical systems". In: *NIPS*. 2007.

[SBP17] 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 (2017), pp. 794–816.

[SC13] C. Schäfer and N. Chopin. "Sequential Monte Carlo on large binary sampling spaces". In: *Stat. Comput.* 23.2 (2013), pp. 163–184.

[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.

[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.

[Sca21] S. Scardapane. *Lecture 8: Beyond single-task supervised learning*. 2021.

[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.

[Sch02] N. N. Schraudolph. "Fast Curvature Matrix-Vector Products for Second-Order Gradient Descent". In: *Neural Computation* 14 (2002).

[Sch04] A. Schrijver. *Combinatorial Optimization*. Springer, 2004.

[Sch+12a] B. Schoelkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. Mooij. "On Causal and Anticausal Learning". In: *ICML*. 2012.

[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 Coference on International Conference on Machine Learning*. 2012, pp. 459–466.

[Sch14] J. Schmidhuber. *Deep Learning in Neural Networks: An Overview*. Tech. rep. 2014.

[Sch+15a] J. Schulman, N. Heess, T. Weber, and P. Abbeel. "Gradient Estimation Using Stochastic Computation Graphs". In: *NIPS*. 2015.

[Sch+15b] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. "Trust Region Policy Optimization". In: *ICML*. 2015.

[Sch+16a] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. "Prioritized Experience Replay". In: *ICLR*. 2016.

[Sch+16b] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In: *ICLR*. 2016.

[Sch+17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. "Proximal Policy Optimization Algorithms". In: (2017). arXiv: 1707.06347 [cs.LG].

[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*. 2018.

[Sch19] B. Schölkopf. "Causality for Machine Learning". In: (2019). arXiv: 1911.10500 [cs.LG].

[Sch20] J. Schmidhuber. *Planning & Reinforcement Learning with Recurrent World Models and Artificial Curiosity*. 2020.

[Sch+20] J. Schrittwieser et al. "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model". In: *Nature* (2020).

[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].

[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 (2021), pp. 612–634.

[Sch+21c] B. Schölkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio. "Towards Causal Representation Learning". In: *CoRR* abs/2102.11107 (2021). arXiv: 2102.11107.

[Sch78] G. Schwarz. "Estimating the dimension of a model". In: *Annals of Statistics* 6.2 (1978), pp. 461–464.

[Sco02] S Scott. "Bayesian methods for hidden Markov models: Recursive computing in the 21st century." In: *JASA* (2002).

[Sco09] S. Scott. "Data augmentation, frequentist estimation, and the Bayesian analysis of multinomial logit models". In: *Statistical Papers* (2009).

[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.

[SCPD22] R. Sanchez-Cauce, I. Paris, and F. J. Diez. "Sum-Product Networks: A Survey". en. In: *IEEE PAMI* 44.7 (July 2022), pp. 3821–3839.

[SCS19] A. Subbaswamy, B. Chen, and S. Saria. *A Universal Hierarchy of Shift-Stable Distributions and the Tradeoff Between Stability and Performance*. 2019. arXiv: 1905.11374 [stat.ML].

[SCS22] A. Subbaswamy, B. Chen, and S. Saria. "A unifying causal framework for analyzing dataset shift-stable learning algorithms". en. In: *Journal of Causal Inference* 10.1 (Jan. 2022), pp. 64–89.

[SD12] J. Sohl-Dickstein. "The Natural Gradient by Analogy to Signal Whitening, and Recipes and Tricks for its Use". In: (2012). arXiv: 1205.1828 [cs.LG].

[SD+15a] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. "Deep Unsupervised Learning using Nonequilibrium Thermodynamics". In: *ICML*. 2015, pp. 2256–2265.

[SD+15b] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli. "Deep Unsupervised Learning using Nonequilibrium Thermodynamics". In: *ICML*. 2015.

[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.

[SE19] Y. Song and S. Ermon. "Generative Modeling by Estimating Gradients of the Data Distribution". In: *NIPS*. 2019, pp. 11895–11907.

[SE20a] J. Song and S. Ermon. "Multi-label Contrastive Predictive Coding". In: *NIPS*. 2020.

[SE20b] Y. Song and S. Ermon. "Improved Techniques for Training Score-Based Generative Models". In: *NIPS*. 2020.

[See+17] M. Seeger, S. Rangapuram, Y. Wang, D. Salinas, J. Gasthaus, T. Januschowski, and V. Flunkert. "Approximate Bayesian Inference in Linear State Space Models for Intermittent Demand Forecasting at Scale". In: (2017). arXiv: 1709.07638 [stat.ML].

[Sej20] T. J. Sejnowski. "The unreasonable effectiveness of deep learning in artificial intelligence". en. In: *PNAS* 117.48 (Dec. 2020), pp. 30033–30038.

[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.

[Sel+19] A. D. Selbst, D. Boyd, S. A. Friedler, S. Venkatasubramanian, and J. Vertesi. "Fairness and Abstraction in Sociotechnical Systems". In: *Proceedings of the Conference on Fairness, Accountability, and Transparency*. FAT* '19. Atlanta, GA, USA: Association for Computing Machinery, 2019, 59–68.

[Sen+08] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. "Collective Classification in Network Data". en. In: *AI Magazine* 29.3 (2008), pp. 93–93.

[Ser+20] J. Serrà, 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.

[Set12] B. Settles. "Active learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6 (2012), 1–114.

[SF08] Z. Svitkina and L. Fleischer. "Submodular approximation: Sampling-based algorithms and lower bounds". In: *FOCS*. 2008.

[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.

[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.

[SG02] J. L. Schafer and J. W. Graham. "Missing data: our view of the state of the art". en. In: *Psychol. Methods* 7.2 (June 2002), pp. 147–177.

[SG05] E. Snelson and Z. Ghahramani. "Compact Approximations to Bayesian Predictive Distributions". In: *ICML*. 2005.

[SG06a] E. Snelson and Z. Ghahramani. "Sparse Gaussian processes using pseudo-inputs". In: *NIPS*. 2006.

[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.

[SG07] M. Steyvers and T. Griffiths. "Probabilistic topic models". In: *Latent Semantic Analysis: A Road to Meaning*. Ed. by T. Landauer, D McNamara, S. Dennis, and W. Kintsch. Laurence Erlbaum, 2007.

[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.

[SGF21] S. Särkkä and Á. F. García-Fernández. "Temporal Parallelization of Bayesian Filters and Smoothers". In: *IEEE Trans. Automat. Contr.* 66.1 (2021).

[SGS16] A. Sharghi, B. Gong, and M. Shah. "Query-focused extractive video summarization". In: *European Conference on Computer Vision*. Springer. 2016, pp. 3–19.

[SH07] R. Salakhutdinov and G. Hinton. "Using Deep Belief Nets to Learn Covariance Kernels for Gaussian Processes". In: *NIPS*. 2007.

[SH09] R. Salakhutdinov and G. Hinton. "Deep Boltzmann Machines". In: *AISTATS*. Vol. 5. 2009, pp. 448–455.

[SH10] R. Salakhutdinov and G. Hinton. "Replicated Softmax: an Undirected Topic Model". In: *NIPS*. 2010.

[SH21] T. Salimans and J. Ho. "Should EBMs model the energy or the score?" In: *ICLR Energy Based Models Workshop*. Apr. 2021.

[SH22] T. Salimans and J. Ho. "Progressive Distillation for Fast Sampling of Diffusion Models". In: *ICLR*. Feb. 2022.

[SHA15] M. A. Skoglund, G. Hendeby, and D. Axehill. "Extended Kalman filter modifications based on an optimization view point". In: *2015 18th International Conference on Information Fusion (Fusion)*. July 2015, pp. 1856–1861.

[Sha+16] B Shahriari, 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 (2016), pp. 148–175.

[Sha16] L. S. Shapley. *17. A value for n-person games*. Princeton University Press, 2016.

[Sha+16] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. "Accessorize 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.

[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.

[Sha+20] H. Shah, K. Tamuly, A. Raghunathan, P. Jain, and P. Netrapalli. "The Pitfalls of Simplicity Bias in Neural Networks". In: *NIPS*. 2020.

[Sha22] C. Shalizi. *Advanced Data Analysis from an Elementary Point of View*. Cambridge University Press, 2022.

[Sha48] C. Shannon. "A mathematical theory of communication". In: *Bell Systems Tech. Journal* 27 (1948), pp. 379–423.

[Sha98] R. Shachter. "Bayes-Ball: The Rational Pastime (for determining Irrelevance and Requisite Information in Belief Networks and Influence Diagrams)". In: *UAI*. 1998.

[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/pdf/10.1002/bimj.201100042`.

[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.

[She+20] T. Shen, J. Mueller, R. Barzilay, and T. Jaakkola. "Educating Text Autoencoders: Latent Representation Guidance via Denoising". In: *ICML*. 2020.

[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: (2021). arXiv: `2108.13624 [cs.LG]`.

[SHF15] R. Steorts, R. Hall, and S. Fienberg. "A Bayesian Approach to Graphical Record Linkage and De-duplication". In: *JASA* (2015).

[Shi00a] H. Shimodaira. "Improving predictive inference under covariate shift by weighting the log-likelihood function". In: *J. Stat. Plan. Inference* 90.2 (2000), pp. 227–244.

[Shi00b] B. Shipley. *Cause and Correlation in Biology: A User's Guide to Path Analysis, Structural Equations and Causal Inference*. Cambridge, 2000.

[Shi+21] C. Shi, D. Sridhar, V. Misra, and D. M. Blei. "On the Assumptions of Synthetic Control Methods". In: (2021). arXiv: `2112.05671 [stat.ME]`.

[SHM14] D. Soudry, I. Hubara, and R. Meir. "Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights". In: *NIPS*. 2014.

[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).

[SHS] D. Stutz, M. Hein, and B. Schiele. "Confidence-calibrated adversarial training: Generalizing to unseen attacks". In: ().

[SHS01] B. Schölkopf, R. Herbrich, and A. J. Smola. "A Generalized Representer Theorem". In: *COLT*. COLT '01/EuroCOLT '01. Springer-Verlag, 2001, pp. 416–426.

[Shu+19a] 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.

[Shu+19b] R. Shu, Y. Chen, A. Kumar, S. Ermon, and B. Poole. "Weakly supervised disentanglement with guarantees". In: *arXiv preprint arXiv:1910.09772* (2019).

[SI00] M. Sato and S. Ishii. "On-line EM algorithm for the normalized Gaussian network". In: *Neural Computation* 12 (2000), pp. 407–432.

[Sil+14] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. "Deterministic Policy Gradient Algorithms". In: *ICML*. ICML'14. JMLR.org, 2014, pp. I–387–I–395.

[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.

[Sil18] D. Silver. *Lecture 9L Exploration and Exploitation*. 2018.

[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 (2018), pp. 1140–1144.

[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.

[Sim02] D. Simon. "Training radial basis neural networks with the extended Kalman Filter". In: *Neurocomputing* (2002).

[Sim06] D. Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley, 2006.

[Sin+00] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári. "Convergence Results for Single-Step On-PolicyReinforcement-Learning Algorithms". In: *MLJ* 38.3 (2000), pp. 287–308.

[Sin67] R. Sinkhorn. "Diagonal Equivalence to Matrices with Prescribed Row and Column Sums". In: *The American Mathematical Monthly* 74.4 (1967), pp. 402–405.

[Sit+20] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein. "Implicit Neural Representations with Periodic Activation Functions". In: *NIPS*. https://www.vincentsitzmann.com/siren/. June 2020.

[Siv+20] T. Sivula, M. Magnusson, A. A. Matamoros, and A. Vehtari. "Uncertainty in Bayesian Leave-One-Out Cross-Validation Based Model Comparison". In: (Aug. 2020). arXiv: `2008.10296 [stat.ME]`.

[SJ08] S. Shirdhonkar 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.

[SJ15] A. Swaminathan and T. Joachims. "Batch Learning from Logged Bandit Feedback through Counterfactual Risk Minimization". In: *JMLR* 16.1 (2015), pp. 1731–1755.

[SJ95] L. Saul and M. Jordan. "Exploiting tractable substructures in intractable networks". In: *NIPS*. Vol. 8. 1995.

[SJ99] L. Saul and M. Jordan. "Mixed memory Markov models: Decomposing complex stochastic processes as mixture of simpler ones". In: *Machine Learning* 37.1 (1999), pp. 75–87.

[SJJ96] L. Saul, T. Jaakkola, and M. Jordan. "Mean Field Theory for Sigmoid Belief Networks". In: *JAIR* 4 (1996), pp. 61–76.

[SJR04] S. Singh, M. James, and M. Rudary. "Predictive state representations: A new theory for modeling dynamical systems". In: *UAI*. 2004.

[SK19] C. Shorten and T. M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning". en. In: *Journal of Big Data* 6.1 (2019), pp. 1–48.

[SK20] S. Singh and S. Krishnan. "Filter Response Normalization Layer: Eliminating Batch Dependence in the Training of Deep Neural Networks". In: *CVPR*. 2020.

[SK89] R. Shachter and C. R. Kenley. "Gaussian Influence Diagrams". In: *Managment Science* 35.5 (1989), pp. 527–550.

[Ski06] J. Skilling. "Nested sampling for general Bayesian computation". In: *Bayesian Analysis* 1.4 (2006), pp. 833–860.

[Ski89] J. Skilling. "The eigenvalues of mega-dimensional matrices". In: *Maximum Entropy and Bayesian Methods*. Springer, 1989, pp. 455–466.

[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.

[SKM18] S. Schwöbel, S. Kiebel, and D. Marković. "Active Inference, Belief Propagation, and the Bethe Approximation". en. In: *Neural Comput.* 30.9 (2018), pp. 2530–2567.

[SKM21] M. Shanahan, C. Kaplanis, and J. Mitrović. "Encoders and Ensembles for Task-Free Continual Learning". In: (2021). arXiv: `2105.13327 [cs.LG]`.

[SKP15] F. Schroff, D. Kalenichenko, and J. Philbin. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

[SKTF18] H. Shao, A. Kumar, and P Thomas Fletcher. "The Riemannian Geometry of Deep Generative Models". In: *CVPR*. 2018, pp. 315–323.

[SKW15] T. Salimans, D. Kingma, and M. Welling. "Markov Chain Monte Carlo and Variational Inference: Bridging the Gap". In: *ICML*. 2015, pp. 1218–1226.

[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.

[SL18] S. L. Smith and Q. V. Le. "A Bayesian Perspective on Generalization and Stochastic Gradient Descent". In: *ICLR*. 2018.

[SL90] D. J. Spiegelhalter and S. L. Lauritzen. "Sequential updating of conditional probabilities on directed graphical structures". In: *Networks* 20 (1990).

[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.

[SLG17] A. Sharghi, J. S. Laurel, and B. Gong. "Query-focused video summarization: Dataset, evaluation, and a memory network based approach". In: *Proceed-*

ings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, pp. 4788–4797.

[Sli19] A. Slivkins. "Introduction to Multi-Armed Bandits". In: *Foundations and Trends in Machine Learning* (2019).

[SLL09] A. L. Strehl, L. Li, and M. L. Littman. "Reinforcement Learning in Finite MDPs: PAC Analysis". In: *JMLR* 10 (2009), pp. 2413–2444.

[SLM92] B. Selman, H. Levesque, and D. Mitchell. "A New Method for Solving Hard Satisfiability Problems". In: *Proceedings of the Tenth National Conference on Artificial Intelligence*. AAAI'92. AAAI Press, 1992, pp. 440–446.

[SLW19] M. Sadinle, J. Lei, and L. Wasserman. "Least Ambiguous Set-Valued Classifiers With Bounded Error Levels". In: *JASA* 114.525 (2019), pp. 223–234.

[SM07] C. Sutton and A. McCallum. "Improved Dynamic Schedules for Belief Propagation". In: *UAI*. 2007.

[SM12] Y Saika and K Morimoto. "Generalized MAP estimation via parameter scheduling and maximizer of the posterior marginal estimate for image reconstruction using multiple halftone images". In: *12th International Conference on Control, Automation and Systems*. 2012, pp. 1285–1289.

[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.

[SME21] J. Song, C. Meng, and S. Ermon. "Denoising Diffusion Implicit Models". In: *ICLR*. 2021.

[SMH07] R. R. Salakhutdinov, A. Mnih, and G. E. Hinton. "Restricted Boltzmann machines for collaborative filtering". In: *ICML*. Vol. 24. 2007, pp. 791–798.

[Smi+00] G. Smith, J. F. G. de Freitas, T. Robinson, and M. Niranjan. "Speech Modelling Using Subspace and EM Techniques". In: *NIPS*. MIT Press, 2000, pp. 796–802.

[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.

[Smi11] N. Smith. *Linguistic structure prediction*. Morgan Claypool, 2011.

[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].

[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. Rumehart and J. McClelland. McGraw-Hill, 1986.

[SMS17] M. Saito, E. Matsumoto, and S. Saito. "Temporal Generative Adversarial Nets with Singular Value Clipping". In: *ICCV*. 2017.

[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 (2018), pp. 1–36.

[Smy20] S. Smyl. "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting". In: *Int. J. Forecast.* 36.1 (2020), pp. 75–85.

[Sn+16] C. K. Sø nderby, T. Raiko, L. Maalø e, S. R. K. Sø nderby, and O. Winther. "Ladder Variational Autoencoders". In: *NIPS*. Curran Associates, Inc., 2016, pp. 3738–3746.

[SNM16] M. Suzuki, K. Nakayama, and Y. Matsuo. "Joint Multimodal Learning with Deep Generative Models". In: (2016). arXiv: 1611.01891 [stat.ML].

[SOB12] R. Snyder, J. K. Ord, and A. Beaumont. "Forecasting the intermittent demand for slow-moving inventories: A modelling approach". In: *Int. J. Forecast.* 28.2 (2012), pp. 485–496.

[Soh16] K. Sohn. "Improved deep metric learning with multi-class n-pair loss objective". In: *Advances in Neural Information Processing Systems*. 2016, pp. 1857–1865.

[Søn+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.

[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.

[Son+21a] Y. Song, C. Durkan, I. Murray, and S. Ermon. "Maximum Likelihood Training of Score-Based Diffusion Models". In: *NIPS*. 2021.

[Son+21b] 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.

[Son98] E. D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. 2nd. Vol. 6. Texts in Applied Mathematics. Springer, 1998.

[SOS92] H. Seung, M. Opper, and H. Sompolinsky. "Query by committee". In: *5th Annual Workshop on Computational Learning Theory*. 1992, 287–294.

[SPD92] S. Shah, F. Palmieri, and M. Datum. "Optimal filtering algorithms for fast learning in feedforward neural networks". In: *Neural Netw.* 5.5 (1992), pp. 779–787.

[Spi71] M. Spivak. *Calculus On Manifolds: A Modern Approach To Classical Theorems Of Advanced Calculus*. Westview Press; 5th edition, 1971.

[SPL20] M. B. Sariyildiz, J. Perez, and D. Larlus. "Learning visual representations with caption annotations". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*. Springer. 2020, pp. 153–170.

[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).

[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.

[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 (2018), pp. 604–610.

[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).

[SQ05] V. Smidl and A. Quinn. *The Variational Bayes Method in Signal Processing*. Springer, 2005.

[SR+14] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. "CNN features off-the-shelf: an astounding baseline for recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2014, pp. 806–813.

[SRG03] R. Salakhutdinov, S. T. Roweis, and Z. Ghahramani. "Optimization with EM and Expectation-Conjugate-Gradient". In: *ICML*. 2003.

[Sri+09] B. K. Sriperumbudur, K. Fukumizu, A. Gretton, B. Schölkopf, and G. R. G. Lanckriet. "On integral probability metrics, φ-divergences and binary classification". In: (2009). arXiv: 0901.2698 [cs.IT].

[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.

[Sri+14a] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *JMLR* (2014).

[Sri+14b] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *J. Mach. Learn. Res.* 15 (2014), pp. 1929–1958.

[Sri+17] A. Srivastava, L. Valkov, 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.

[SRS10] P. Schnitzspan, S. Roth, and B. Schiele. "Automatic discovery of meaningful object parts with latent CRFs". In: *CVPR*. 2010.

[SS15] D. J. Sutherland and J. Schneider. "On the Error of Random Fourier Features". In: *UAI*. June 2015.

[SS17a] A. Srivastava and C. Sutton. "Autoencoding Variational Inference For Topic Models". In: *ICLR*. 2017.

[SS17b] A. Srivastava and C. Sutton. "Autoencoding Variational Inference For Topic Models". In: *ICLR*. 2017.

[SS18a] O. Sener and S. Savarese. "Active Learning for Convolutional Neural Networks: A Core-Set Approach". In: *International Conference on Learning Representations*. 2018.

[SS18b] 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.

[SS19] S. Sarkka and A. Solin. *Applied stochastic differential equations*. en. Cambridge University Press, 2019.

[SS20a] S. Sarkka and L. Svensson. "Levenberg-Marquardt and Line-Search Extended Kalman Smoothers". In: *ICASSP*. Barcelona, Spain: IEEE, May 2020.

[SS20b] K. E. Smith and A. O. Smith. "Conditional GAN for timeseries generation". In: *arXiv preprint arXiv:2006.16477* (2020).

[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.

[SS23] S. Sarkka and L. Svensson. *Bayesian Filtering and Smoothing (2nd edition)*. Cambridge University Press, 2023.

[SS82] R. H. Shumway and D. S. Stoffer. "An approach to time series smoothing and forecasting using the em algorithm". en. In: *J. Time Ser. Anal.* 3.4 (July 1982), pp. 253–264.

[SSA14] K. Swersky, J. Snoek, and R. P. Adams. "Freeze-Thaw Bayesian Optimization". In: (2014). arXiv: 1406.3896 [stat.ML].

[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.

[SSB17] S. Semeniuta, A. Severyn, and E. Barth. "A Hybrid Convolutional Variational Autoencoder for Text Generation". In: (2017). arXiv: 1702.02390 [cs.CL].

[SSE18] Y. Song, J. Song, and S. Ermon. "Accelerating Natural Gradient with Higher-Order Invariance". In: *ICML*. 2018.

[SSF16] M. W. Seeger, D. Salinas, and V. Flunkert. "Bayesian Intermittent Demand Forecasting for Large Inventories". In: *NIPS*. 2016, pp. 4646–4654.

[SSG18a] S. Semeniuta, A. Severyn, and S. Gelly. "On Accurate Evaluation of GANs for Language Generation". In: *arXiv preprint arXiv:1806.04936* (2018).

[SSG18b] S. Semeniuta, A. Severyn, and S. Gelly. "On Accurate Evaluation of GANs for Language Generation". In: (2018). arXiv: 1806.04936 [cs.CL].

[SSG19] R. Singh, M. Sahani, and A. Gretton. "Kernel Instrumental Variable Regression". In: *Advances in Neural Information Processing Systems*. 2019, pp. 4593–4605.

[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).

[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.

[SSK12] M. Sugiyama, T. Suzuki, and T. Kanamori. *Density Ratio Estimation in Machine Learning*. en. Cambridge University Press, 2012.

[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.

[SSZ17] J. Snell, K. Swersky, and R. Zemel. "Prototypical networks for few-shot learning". In: *NIPS*. 2017, pp. 4077–4087.

[Sta] *Scientific Explanation*. https://plato.stanford.edu/entries/scientific-explanation/#ConcOpenIssuFutuDire. Accessed: 2021-11-23.

[Sta07] K. O. Stanley. "Compositional pattern producing networks: A novel abstraction of development". In: *Genet. Program. Evolvable Mach.* 8.2 (2007), pp. 131–162.

[Sta+17] N. Stallard, F. Miller, S. Day, S. W. Hee, J. Madan, S. Zohar, and M. Posch. "Determination of the optimal sample size for a clinical trial accounting for the population size". en. In: *Biom. J.* 59.4 (2017), pp. 609–625.

[Sta+19] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen. "Designing neural networks through neuroevolution". In: *Nature Machine Intelligence* 1.1 (2019).

[Sta+20] T. Standley, A. R. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese. "Which Tasks Should Be Learned Together in Multi-task Learning?" In: *ICML*. 2020.

[Ste00] M. Stephens. "Dealing with label-switching in mixture models". In: *J. Royal Statistical Society, Series B* 62 (2000), pp. 795–809.

[Ste81] C. M. Stein. "Estimation of the mean of a multivariate normal distribution". In: *The annals of Statistics* (1981), pp. 1135–1151.

[Sto09] A. J. Storkey. "When Training and Test Sets are Different: Characterising Learning Transfer". In: *Dataset Shift in Machine Learning*. 2009.

[Sto17] J. Stoehr. "A review on statistical inference methods for discrete Markov random fields". In: (2017). arXiv: 1704.03331 [stat.ME].

[STR10] Y. Saatchi, R. Turner, and C. E. Rasmussen. "Gaussian Process Change Point Models". In: *ICML*. unknown, 2010, pp. 927–934.

[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.

[Str19] M. Streeter. "Bayes Optimal Early Stopping Policies for Black-Box Optimization". In: (2019). arXiv: 1902.08285 [cs.LG].

[Stu+22] D. Stutz, Krishnamurthy, Dvijotham, A. T. Cemgil, and A. Doucet. "Learning Optimal Conformal Classifiers". In: *ICLR*. 2022.

[STY17] M. Sundararajan, A. Taly, and Q. Yan. *Axiomatic Attribution for Deep Networks*. 2017. arXiv: 1703.01365 [cs.LG].

[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.

[Sud+03] E. Sudderth, A. Ihler, W. Freeman, and A. Willsky. "Nonparametric Belief Propagation". In: *CVPR*. 2003.

[Sud06] E. Sudderth. "Graphical Models for Visual Object Recognition and Tracking". PhD thesis. MIT, 2006.

[Sud+10] E. Sudderth, A. Ihler, M. Isard, W. Freeman, and A. Willsky. "Nonparametric Belief Propagation". In: *Comm. of the ACM* 53.10 (2010).

[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 (2013), pp. 2734–2775.

[Sun+09] L. Sun, S. Ji, S. Yu, and J. Ye. "On the Equivalence Between Canonical Correlation Analysis and Orthonormalized Partial Least Squares". In: *IJCAI*. 2009.

[Sun+17] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. "Revisiting unreasonable effectiveness of data in deep learning era". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 843–852.

[Sun+18] S. Sun, G. Zhang, C. Wang, W. Zeng, J. Li, and R. Grosse. "Differentiable Compositional Kernel Learning for Gaussian Processes". In: *ICML*. 2018.

[Sun+19a] S. Sun, G. Zhang, J. Shi, and R. Grosse. "Functional variational bayesian neural networks". In: *arXiv preprint arXiv:1903.05779* (2019).

[Sun+19b] S. Sun, Z. Cao, H. Zhu, and J. Zhao. "A Survey of Optimization Methods from a Machine Learning Perspective". In: (2019). arXiv: 1906.06821 [cs.LG].

[Sun+19c] M. Sundararajan, J. Xu, A. Taly, R. Sayres, and A. Najmi. "Exploring Principled Visualizations for Deep Network Attributions." In: *IUI Workshops*. Vol. 4. 2019.

[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.

[Sun+22] T. Sun, M. Segu, J. Postels, Y. Wang, L. Van Gool, B. Schiele, F. Tombari, and F. Yu. "SHIFT: A Synthetic Driving Dataset for Continuous Multi-Task Domain Adaptation". In: *CVPR*. June 2022.

[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.

[Sut19] R. Sutton. *The Bitter Lesson*. 2019.

[Sut88] R. Sutton. "Learning to predict by the methods of temporal differences". In: *Machine Learning* 3.1 (1988), pp. 9–44.

[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. Morgan Kaufmann, 1990, pp. 216–224.

[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.

[Sut+99] R. Sutton, D. McAllester, S. Singh, and Y. Mansour. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *NIPS*. 1999.

[SV08] G. Shafer and V. Vovk. "A Tutorial on Conformal Prediction". In: *JMLR* 9.Mar (2008), pp. 371–421.

[SV14] S. L. Scott and H. R. Varian. "Predicting the present with Bayesian structural time series". In: *International Journal of Mathematical Modelling and Numerical Optimisation* 5.1-2 (2014), pp. 4–23.

[SV98] M. Studenty 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.

[SVE04] A. Smola, S. V. N. Vishwanathan, and E. Eskin. "Laplace Propagation". In: *NIPS*. MIT Press, 2004, pp. 441–448.

[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.

[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).

[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).

[SW06] J. E. Smith and R. L. Winkler. "The Optimizer's Curse: Skepticism and Postdecision Surprise in Decision Analysis". In: *Manage. Sci.* 52.3 (2006), pp. 311–322.

[SW13] G. J. Sussman and J. Wisdom. *Functional Differential Geometry*. Functional Differential Geometry. MIT Press, 2013.

[SW20] V. G. Satorras and M. Welling. "Neural Enhanced Belief Propagation on Factor Graphs". In: (2020). arXiv: 2003.01998 [cs.LG].

[SW87a] M. Shewry and H. Wynn. "Maximum entropy sampling". In: *J. Applied Statistics* 14 (1987), 165–170.

[SW87b] R. Swendsen and J.-S. Wang. "Nonuniversal critical dynamics in Monte Carlo simulations". In: *Physical Review Letters* 58 (1987), pp. 86–88.

[SW89] S. Singhal and L. Wu. "Training Multilayer Perceptrons with the Extended Kalman Algorithm". In: *NIPS*. Vol. 1. 1989.

[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.

[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.

[SWL03] M. Seeger, C. K. I. Williams, and N. D. Lawrence. "Fast Forward Selection to Speed Up Sparse Gaussian Process Regression". In: *AISTATS*. 2003.

[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.

[SZ22] R. Shwartz-Ziv. "Information Flow in Deep Neural Networks". PhD thesis. 2022.

[SZ+22] R. Shwartz-Ziv, M. Goldblum, H. Souri, S. Kapoor, C. Zhu, Y. LeCun, and A. G. Wilson. "Pre-Train Your Loss: Easy Bayesian Transfer Learning with Informative Priors". In: (May 2022). arXiv: 2205.10279 [cs.LG].

[Sze10] C. Szepesvari. *Algorithms for Reinforcement Learning*. Morgan Claypool, 2010.

[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.

[Sze+15a] 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.

[Sze+15b] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the Inception Architecture for Computer Vision". In: (2015). arXiv: 1512.00567 [cs.CV].

[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.

[TB16] P. S. Thomas and E. Brunskill. "Data-Efficient Off-Policy Policy Evaluation for Reinforcement Learning". In: *ICML*. 2016, pp. 2139–2148.

[TB22] A. Tiulpin and M. B. Blaschko. "Greedy Bayesian Posterior Approximation with Deep Ensembles". In: *Trans. on Machine Learning Research* (2022).

[TB99] M. Tipping and C. Bishop. "Probabilistic principal component analysis". In: *J. of Royal Stat. Soc. Series B* 21.3 (1999), pp. 611–622.

[TBA19] N. Tremblay, S. Barthelmé, and P.-O. Amblard. "Determinantal Point Processes for Coresets." In: *J. Mach. Learn. Res.* 20 (2019), pp. 168–1.

[TBB19] J. Townsend, T. Bird, and D. Barber. "Practical Lossless Compression with Latent Variables using Bits Back Coding". In: *ICLR*. 2019.

[TBF06] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2006.

[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.

[TCG21] Y. Tian, X. Chen, and S. Ganguli. "Understanding self-supervised learning dynamics without contrastive pairs". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 10268–10278.

[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.

[Teh+06] Y.-W. Teh, M. Jordan, M. Beal, and D. Blei. "Hierarchical Dirichlet processes". In: *JASA* 101.476 (2006), pp. 1566–1581.

[Teh+20] N. Tehrani, N. S. Arora, Y. L. Li, K. D. Shah, D. Noursi, M. Tingley, N. Torabi, S. Masouleh, E. Lippert, and E. Meijer. "Bean 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.

[Ten+20] I. Tenney, J. Wexler, J. Bastings, T. Bolukbasi, A. Coenen, S. Gehrmann, E. Jiang, M. Pushkarna, C. Radebaugh, E. Reif, et al. "The language interpretability tool: Extensible, interactive visualizations and analysis for NLP models". In: *arXiv preprint arXiv:2008.05122* (2020).

[TF03] M. Tipping and A. Faul. "Fast marginal likelihood maximisation for sparse Bayesian models". In: *AI/Stats*. 2003.

[TG18] L. Tu and K. Gimpel. "Learning Approximate Inference Networks for Structured Prediction". In: *ICLR*. 2018.

[TGFS18] F. Tronarp, Á. F. García-Fernández, and S. Särkkä. "Iterative Filtering and Smoothing in Nonlinear and Non-Gaussian Systems Using Conditional Moments". In: *IEEE Signal Process. Lett.* 25.3 (2018), pp. 408–412.

[TGK03] B. Taskar, C. Guestrin, and D. Koller. "Max-Margin Markov Networks". In: *NIPS*. 2003.

[TH09] T. Tieleman and G. Hinton. “Using Fast Weights to Improve Persistent Contrastive Divergence”. In: *ICML*. 2009, pp. 1033–1040.

[Tho+19] V. Thomas, F. Pedregosa, B. van Merriënboer, P.-A. Mangazol, Y. Bengio, and N. Le Roux. “Information matrices and generalization”. In: (2019). arXiv: 1906.07774 [cs.LG].

[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.

[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).

[Thr98] S. Thrun. “Lifelong learning algorithms”. In: *Learning to learn*. Ed. by S. Thrun and L. Pratt. Kluwer, 1998, pp. 181–209.

[Thu+21] S. Thulasidasan, S. Thapa, S. Dhaubhadel, G. Chennupati, T. Bhattacharya, and J. Bilmes. “An Effective Baseline for Robustness to Distributional Shift”. In: (2021). arXiv: 2105.07107 [cs.LG].

[Tia+20] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola. “What makes for good views for contrastive learning”. In: *ArXiv* abs/2005.10243 (2020).

[Tib+19] R. J. Tibshirani, R. F. Barber, E. J. Candes, and A. Ramdas. “Conformal Prediction Under Covariate Shift”. In: *NIPS*. 2019.

[Tib96] R. Tibshirani. “Regression shrinkage and selection via the lasso”. In: *J. Royal. Statist. Soc B* 58.1 (1996), pp. 267–288.

[Tie08] T. Tieleman. “Training restricted Boltzmann machines using approximations to the likelihood gradient”. In: *ICML*. ACM New York, NY, USA. 2008, pp. 1064–1071.

[Tip01] M. Tipping. “Sparse Bayesian learning and the relevance vector machine”. In: *JMLR* 1 (2001), pp. 211–244.

[Tip+03] M. K. Tippett, J. L. Anderson, C. H. Bishop, T. M. Hamill, and J. S. Whitaker. “Ensemble Square Root Filters”. en. In: *Mon. Weather Rev.* 131.7 (July 2003), pp. 1485–1490.

[Tip98] M. Tipping. “Probabilistic visualization of high-dimensional binary data”. In: *NIPS*. 1998.

[Tit09] M. K. Titsias. “Variational Learning of Inducing Variables in Sparse Gaussian Processes”. In: *AISTATS*. 2009.

[TK86] L. Tierney and J. Kadane. “Accurate approximations for posterior moments and marginal densities”. In: *JASA* 81.393 (1986).

[TKI20] Y. Tian, D. Krishnan, and P. Isola. “Contrastive Multiview Coding”. In: *ECCV*. 2020.

[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.

[TL18a] S. J. Taylor and B. Letham. “Forecasting at scale”. en. In: *The American Statistician* 72.1 (2018), pp. 37–45.

[TL18b] G. Tucker and D. Lawson. “Doubly Reparameterized Gradient Estimators for Monte Carlo Objectives”. In: *1st Symposium on Advances in Approximate Bayesian Inference*. 2018.

[TL19] M. Tan and Q. Le. “Efficientnet: Rethinking model scaling for convolutional neural networks”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6105–6114.

[TLG08] D. G. Tzikas, A. C. Likas, and N. P. Galatsanos. “The variational approximation for Bayesian inference”. In: *IEEE Signal Process. Mag.* 25.6 (Nov. 2008), pp. 131–146.

[TLG14] M. Titsias and M. Lázaro-Gredilla. “Doubly Stochastic Variational Bayes for non-Conjugate Inference”. In: *ICML*. 2014, pp. 1971–1979.

[TM15] D. Trafimow and M. Marks. “Editorial”. In: *Basic Appl. Soc. Psych.* 37.1 (2015), pp. 1–2.

[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.

[TMK04] G. Theocharous, K. Murphy, and L. Kaelbling. “Representing hierarchical POMDPs as DBNs for multi-scale robot localization”. In: *ICRA*. 2004.

[TN13] L. S. L. Tan and D. J. Nott. “Variational Inference for Generalized Linear Mixed Models Using Partially Noncentered Parametrizations”. In: *Stat. Sci.* (2013).

[TND21] M.-N. Tran, T.-N. Nguyen, and V.-H. Dao. “A practical tutorial on Variational Bayes”. In: (2021). arXiv: 2103.01327 [stat.CO].

[TOB16] L. Theis, A. van den Oord, and M. Bethge. “A note on the evaluation of generative models”. In: *ICLR*. 2016.

[Tol22] S. Toledo. “UltimateKalman: Flexible Kalman Filtering and Smoothing Using Orthogonal Transformations”. In: (July 2022). arXiv: 2207.13526 [math.NA].

[Tom+20] R. Tomsett, D. Harborne, S. Chakraborty, P. Gurram, and A. Preece. “Sanity checks for saliency metrics”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 04. 2020, pp. 6021–6029.

[Tom22] J. M. Tomczak. *Deep Generative Modeling*. en. 1st ed. Springer, 2022.

[Tou09] M. Toussaint. “Robot Rrajectory Optimization using Approximate Inference”. In: *ICML*. 2009, pp. 1049–1056.

[Tou14] M. Toussaint. *Bandits, Global Optimization, Active Learning, and Bayesian RL – understanding the common ground*. Autonomous Learning Summer School. 2014.

[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 (2019), pp. 1203–1215.

[TOV18] C. Truong, L. Oudre, and N. Vayatis. “Selective review of offline change point detection methods”. In: (2018). arXiv: 1801.00718 [cs.CE].

[TP97] S. Thrun and L. Pratt, eds. *Learning to learn*. Kluwer, 1997.

[TPB00] N. Tishby, F. C. Pereira, and W. Bialek. “The information bottleneck method”. In: *ArXiv* physics/0004057 (2000).

[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.

[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.

[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.

[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.

[Tra+20a] L. Tran, B. S. Veeling, K. Roth, J. Swiatkowski, J. V. Dillon, J. Snoek, S. Mandt, T. Salimans, S. Nowozin, and R. Jenatton. "Hydra: Preserving Ensemble Diversity for Model Distillation". In: (2020). arXiv: 2001.04694 [cs.LG].

[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 (2020), pp. 97–113.

[TRB16] D. Tran, R. Ranganath, and D. M. Blei. "The Variational Gaussian Process". In: *ICLR*. 2016.

[Tri21] K. Triantafyllopoulos. *Bayesian Inference of State Space Models: Kalman Filtering and Beyond*. en. 1st ed. Springer, 2021.

[TS06] M. Toussaint and A. Storkey. "Probabilistic inference for solving discrete and continuous state Markov Decision Processes". In: *ICML*. 2006, pp. 945–952.

[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.

[Tsa+19] Y.-H. H. Tsai, S. Bai, M. Yamada, L.-P. Morency, and R. Salakhutdinov. "Transformer Dissection: An Unified Understanding for Transformer's Attention via the Lens of Kernel". In: *EMNLP*. Association for Computational Linguistics, 2019, pp. 4344–4353.

[Tsa+21] Y.-H. H. Tsai, S. Bai, L.-P. Morency, and R. Salakhutdinov. "A Note on Connecting Barlow Twins with Negative-Sample-Free Contrastive Learning". In: *ArXiv* abs/2104.13712 (2021).

[Tsa88] C. Tsallis. "Possible generalization of Boltzmann-Gibbs statistics". In: *J. of Statistical Physics* 52 (1988), pp. 479–487.

[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.

[Tsc+19] M. Tschannen, J. Djolonga, P. K. Rubenstein, S. Gelly, and M. Lucic. "On Mutual Information Maximization for Representation Learning". In: *arXiv preprint arXiv:1907.13625* (2019).

[Tsi+17] P. A. Tsividis, T. Pouncy, J. L. Xu, J. B. Tenenbaum, and S. J. Gershman. "Human Learning in Atari". en. In: *AAAI Spring Symposium Series*. 2017.

[Tso+05] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. "Large Margin Methods for Structured and Interdependent Output Variables". In: *JMLR* 6 (2005), pp. 1453–1484.

[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.

[TT17] B. Trippe and R. Turner. "Overpruning in Variational Bayesian Neural Networks". In: *NIPS Workshop on Advances in Approximate Bayesian Inference*. 2017.

[Tuc+19] G. Tucker, D. Lawson, B. Dai, and R. Ranganath. "Revisiting auxiliary latent variables in generative models". In: (2019).

[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 (2017), p. 16.

[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.

[Tur+08] R. Turner, P. Berkes, M. Sahani, and D. Mackay. *Counterexamples to variational free energy compactness folk theorems*. Tech. rep. U. Cambridge, 2008.

[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.

[TW16] J. M. Tomczak and M. Welling. "Improving variational auto-encoders using Householder flow". In: *NeurIPS Workshop on Bayesian Deep Learning* (2016).

[TX00] J. B. Tenenbaum and F. Xu. "Word learning as Bayesian inference". In: *Proc. 22nd Annual Conf.of the Cognitive Science Society*. 2000.

[TZ02] Z. Tu and S. Zhu. "Image Segmentation by Data-Driven Markov Chain Monte Carlo". In: *IEEE PAMI* 24.5 (2002), pp. 657–673.

[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.

[UAP22] A. Ulhaq, N. Akhtar, and G. Pogrebna. "Efficient Diffusion Models for Vision: A Survey". In: (Oct. 2022). arXiv: 2210.09292 [cs.CV].

[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 (2017), pp. 1075–1099.

[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.

[UHJ20] M. Uehara, J. Huang, and N. Jiang. "Minimax Weight and Q-Function Learning for Off-Policy Evaluation". In: *ICML*. 2020.

[UML13] B. Uria, I. Murray, and H. Larochelle. "RNADE: The real-valued neural autoregressive density-estimator". In: *NIPS*. 2013.

[UML14] B. Uria, I. Murray, and H. Larochelle. "A Deep and Tractable Density Estimator". In: *ICML*. 2014.

[UN98] N. Ueda and R. Nakano. "Deterministic annealing EM algorithm". In: *Neural Networks* 11 (1998), pp. 271–282.

[UR16] B. Ustun and C. Rudin. "Supersparse linear integer models for optimized medical scoring systems". In: *Machine Learning* 102.3 (2016), pp. 349–391.

[Uri+16] B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle. "Neural Autoregressive Distribution Estimation". In: *JMLR* (2016).

[UTR14] B. Ustun, S. Tracà, and C. Rudin. *Supersparse Linear Integer Models for Interpretable Classification*. 2014. arXiv: 1306.6677 [stat.ML].

[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).

[UVL16] D. Ulyanov, A. Vedaldi, and V. Lempitsky. "Instance Normalization: The Missing Ingredient for Fast Stylization". In: (2016). arXiv: 1607.08022 [cs.CV].

[UVL18] D. Ulyanov, A. Vedaldi, and V. Lempitsky. "Deep Image Prior". In: *CVPR*. 2018.

[Vaa00] A. W. Van der Vaart. *Asymptotic statistics*. Vol. 3. Cambridge university press, 2000.

[Val00] H. Valpola. "Bayesian Ensemble Learning for Nonlinear Factor Analysis". PhD thesis. Helsinki University of Technology, 2000.

[Van10] J. Vanhatalo. "Speeding up the inference in Gaussian process models". PhD thesis. Helsinki Univ. Technology, 2010.

[Van14] J. VanderPlas. *Frequentism and Bayesianism III: Confidence, Credibility, and why Frequentism and Science do not Mix*. Blog post. 2014.

[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.

[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.

[Vas+17b] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *NIPS*. 2017, pp. 5998–6008.

[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.

[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.

[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.

[VCB22] M. J. Vowels, N. C. Camgoz, and R. Bowden. "D'ya Like DAGs? A Survey on Structure Learning and Causal Discovery". In: *ACM Comput. Surv.* 55.4 (Nov. 2022), pp. 1–36.

[VDMW03] R. Van Der Merwe and E. Wan. "Sigma-Point Kalman Filters for probabilistic inference in dynamic state-space models". In: *Workshop on Advances in ML*. 2003.

[Ved+18] R. Vedantam, I. Fischer, J. Huang, and K. Murphy. "Generative Models of Visually Grounded Imagination". In: *ICLR*. 2018.

[Veh+15] A. Vehtari, D. Simpson, A. Gelman, Y. Yao, and J. Gabry. "Pareto Smoothed Importance Sampling". In: (July 2015). arXiv: 1507.02646 [stat.CO].

[Veh+19] A. Vehtari, A. Gelman, D. Simpson, B. Carpenter, and P.-C. Bürkner. "Rank-normalization, folding, and localization: An improved $\widehat{R}$ for assessing convergence of MCMC". In: (2019). arXiv: 1903.08008 [stat.CO].

[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*. 2021.

[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).

[Ver18] R. Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. en. 1 edition. Cambridge University Press, 2018.

[Ver+19] A. Vergari, A. Molina, R. Peharz, Z. Ghahramani, K. Kersting, and I. Valera. "Automatic Bayesian Density Analysis". In: *AAAI*. 2019.

[VF+80] B. C. Van Fraassen et al. *The scientific image*. Oxford University Press, 1980.

[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 (2017), pp. 1413–1432.

[VGS05] V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic Learning in a Random World*. en. Springer, 2005.

[VGS22] V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic Learning in a Random World*. en. Second. Springer, 2022.

[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.

[Vil08] C. Villani. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2008.

[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+08] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 1096–1103.

[Vin+10a] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". In: *Journal of machine learning research* 11.Dec (2010), pp. 3371–3408.

[Vin+10b] M. Vinyals, J. Cerquides, 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*. ACM Press, 2006.

[Vis+10] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgward. "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.

[VJV09] J. Vanhatalo, P. Jylänki, and A. Vehtari. "Gaussian process regression with Student-t likelihood". In: *NIPS*. Vol. 22. 2009.

[VK20a] A. Vahdat and J. Kautz. "NVAE: A Deep Hierarchical Variational Autoencoder". In: *NIPS*. 2020.

[VK20b] A. Vahdat and J. Kautz. "Nvae: A deep hierarchical variational autoencoder". In: *arXiv preprint arXiv:2007.03898* (2020).

[VKK21] A. Vahdat, K. Kreis, and J. Kautz. "Score-based Generative Modeling in Latent Space". In: *NIPS*. June 2021.

[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 (CLVision)*. 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 nonnegative 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. Luo, and V. Vovk. Springer Berlin Heidelberg, 2013, pp. 105–116.

[VPV10] J. Vanhatalo, V. Pietilainen, 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.

[VR18] S. Verma and J. Rubin. "Fairness Definitions Explained". In: *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)*. May 2018, pp. 1–7.

[VRF11] C. Varin, N. Reid, and D. Firth. "An overview of composite likelihood methods". In: *Stat. Sin.* 21.1 (2011), pp. 5–42.

[VS11] T. J. VanderWeele and I. Shpitser. "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.

[VW21] J. de Vilmarest and O. Wintenberger. "Viking: Variational Bayesian Variance Tracking". In: (Apr. 2021). arXiv: 2104.10777 [cs.LG].

[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.

[Wal+09] H. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno. "Evaluation Methods for Topic Models". In: *ICML*. 2009.

[Wal+20] M. Walmsley et al. "Galaxy Zoo: probabilistic morphology through Bayesian CNNs and active learning". In: *Monthly Notices Royal Astronomical Society* 491.2 (2020), pp. 1554–1574.

[Wal47] A. Wald. "An Essentially Complete Class of Admissible Decision Functions". en. In: *Ann. Math. Stat.* 18.4 (1947), pp. 549–555.

[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: *JASA* 112.517 (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 (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.

[Wan+19b] S. Wang, W. Bai, C. Lavania, and J. Bilmes. "Fixing Mini-batch Sequences with Hierarchical Robust Partitioning". 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.

[Wan+19c] Y. Wang, A. Smola, D. C. Maddix, J. Gasthaus, D. Foster, and T. Januschowski. "Deep Factors for Forecasting". In: *ICML*. 2019.

[Wan+20a] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell. "Tent: Fully Test-Time Adaptation by Entropy Minimization". In: *ICLR*. 2020.

[Wan+20b] H. Wang, X. Wu, Z. Huang, and E. P. Xing. "High-frequency component helps explain the generalization of convolutional neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8684–8694.

[Wan+20c] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros. *Dataset Distillation*. 2020. arXiv: 1811.10959 [cs.LG].

[Wan+20d] Z. Wang, S. Cheng, L. Yueru, J. Zhu, and B. Zhang. "A Wasserstein Minimum Velocity Approach to Learning Unnormalized Models". 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. 3728–3738.

[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*. 2021.

[Wan+22] H. Wang, Y. Yang, D. Pati, and A. Bhattacharya. "Structured Variational Inference in Bayesian State-Space Models". In: (2022).

[Wan+23] L. Wang, X. Zhang, H. Su, and J. Zhu. "A Comprehensive Survey of Continual Learning: Theory, Method and Application". In: (Jan. 2023). arXiv: 2302.00487 [cs.LG].

[Was04] L. Wasserman. *All of statistics. A concise course in statistical inference*. Springer, 2004.

[Was06] L. Wasserman. *All of Nonparametric Statistics*. Springer, 2006.

[Wat10] S. Watanabe. "Asymptotic Equivalence of Bayes Cross Validation and Widely Applicable Information Criterion in Singular Learning Theory". In: *JMLR* 11 (2010), pp. 3571–3594.

[Wat13] S. Watanabe. "A Widely Applicable Bayesian Information Criterion". In: *JMLR* 14 (2013), pp. 867–897.

[Wat60] S. Watanabe. "Information theoretical analysis of multivariate correlation". In: *IBM J. of Research and Development* 4 (1960), pp. 66–82.

[WB05] J. Winn and C. Bishop. "Variational Message Passing". In: *JMLR* 6 (2005), pp. 661–694.

[WB20] T. Wang and J. Ba. "Exploring Model-based Planning with Policy Networks". In: *ICLR*. 2020.

[WBC21] Y. Wang, D. M. Blei, and J. P. Cunningham. "Posterior Collapse and Latent Variable Non-identifiability". In: *NIPS*. 2021.

[WC19] A. Wang and K. Cho. "BERT has a Mouth, and It Must Speak: BERT as a Markov Random Field Language Model". In: *Proc. Workshop on Methods for Optimizing and Evaluating Neural Language Generation*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 30–36.

[WCS08] M. Welling, C. Chemudugunta, and N. Sutter. "Deterministic Latent Variable Models and their Pitfalls". In: *ICDM*. 2008.

[WD92] C. Watkins and P. Dayan. "Q-learning". In: *Machine Learning* 8.3 (1992), pp. 279–292.

[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.

[Web+17] T. Weber et al. "Imagination-Augmented Agents for Deep Reinforcement Learning". In: *NIPS*. 2017.

[Wei00] Y. Weiss. "Correctness of local probability propagation in graphical models with loops". In: *Neural Computation* 12 (2000), pp. 1–41.

[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.

[Wei+14] K. Wei, Y. Liu, K. Kirchhoff, and J. Bilmes. "Unsupervised Submodular Subset Selection for Speech Data". In: *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*. Florence, Italy, 2014.

[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 Submodular Partitioning". In: *Neural Information Processing Society (NeurIPS, formerly NIPS) Workshop*. LearningSys Workshop, http://learningsys.org. Montreal, Canada, 2015.

[Wei+15b] K. Wei, R. Iyer, S. Wang, W. Bai, and J. Bilmes. "Mixed Robust/Average Submodular Partitioning: Fast Algorithms, Guarantees, and Applications". In: *Neural Information Processing Society (NeurIPS, formerly NIPS)*. Montreal, Canada, 2015.

[Wei+22] J. Wei et al. "Emergent Abilities of Large Language Models". In: *TMLR* (June 2022).

[Wel11] M. Welling. "Bayesian Learning via Stochastic Gradient Langevin Dynamics". In: *ICML*. 2011.

[Wen+17] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka. "A Multi-Horizon Quantile Recurrent Forecaster". In: *NIPS Time Series Workshop*. 2017.

[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.

[Wen+19b] F. Wenzel, T. Galy-Fajou, C. Donner, M. Kloft, and M. Opper. "Efficient Gaussian Process Classification Using Polya-Gamma Data Augmentation". In: *AAAI*. 2019.

[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).

[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.

[Wen+20c] F. Wenzel, J. Snoek, D. Tran, and R. Jenatton. "Hyperparameter Ensembles for Robustness and Uncertainty Quantification". In: *NIPS*. 2020.

[Wen21] L. Weng. *What are diffusion models?* 2021.

[Wes03] M. West. "Bayesian Factor Regression Models in the "Large p, Small n" Paradigm". In: *Bayesian Statistics 7* (2003).

[Wes87] M. West. "On scale mixtures of normal distributions". In: *Biometrika* 74 (1987), pp. 646–648.

[WF01a] Y. Weiss and W. T. Freeman. "Correctness of belief propagation in Gaussian graphical models of arbitrary topology". In: *Neural Computation* 13.10 (2001), pp. 2173–2200.

[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, Special Issue on Codes on Graphs and Iterative Algorithms* 47.2 (2001), pp. 723–735.

[WF14] Z. Wang and N. de Freitas. "Theoretical Analysis of Bayesian Optimisation with Unknown Gaussian Process Hyper-Parameters". In: (2014). arXiv: 1406.7758 [stat.ML].

[WF16] Z. Wang and N. de Freitas. "Theoretical Analysis of Bayesian Optimisation with Unknown Gaussian Process Hyper-Parameters". In: *BayesOpt Workshop*. 2016.

[WG17] P. White and S. Gorard. "Against Inferential Statistics: How and Why Current Statistics Teaching Gets It Wrong". en. In: *J. Educ. Behav. Stat.* 16.1 (2017), pp. 55–65.

[WG18] M. Wu and N. Goodman. "Multimodal Generative Models for Scalable Weakly-Supervised Learning". In: *NIPS*. 2018.

[WGY21] G. Weiss, Y. Goldberg, and E. Yahav. "Thinking Like Transformers". In: *ICML*. 2021.

[WH02] M. Welling and G. E. Hinton. "A new learning algorithm for mean field Boltzmann machines". In: *International Conference on Artificial Neural Networks*. Springer. 2002, pp. 351–357.

[WH18] Y. Wu and K. He. "Group Normalization". In: *ECCV*. 2018.

[WH97] M. West and J. Harrison. *Bayesian forecasting and dynamic models*. Springer, 1997.

[WHD18] J. T. Wilson, F. Hutter, and M. P. Deisenroth. "Maximizing acquisition functions for Bayesian optimization". In: *NIPS*. 2018.

[Whi16] T. White. "Sampling Generative Networks". In: *arXiv* (2016).

[Whi88] P Whittle. "Restless bandits: activity allocation in a changing world". In: *J. Appl. Probab.* 25.A (1988), pp. 287–298.

[WHT19] Y. Wang, H. He, and X. Tan. "Truly Proximal Policy Optimization". In: *UAI*. 2019.

[WI20] A. G. Wilson and P. Izmailov. "Bayesian Deep Learning and a Probabilistic Perspective of Generalization". In: *NIPS*. 2020.

[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.

[Wie+14] D Wierstra, T Schaul, J Peters, and J Schmidhuber. "Natural Evolution Strategies". In: *JMLR* 15.1 (2014), pp. 949–980.

[Wik21] Wikipedia contributors. *CliffsNotes — Wikipedia, The Free Encyclopedia*. [Online; accessed 29-December-2021]. 2021.

[Wil+14] A. G. Wilson, E. Gilboa, A. Nehorai, and J. P. Cunningham. "Fast kernel learning for multidimensional pattern extrapolation". In: *Advances in Neural Information Processing Systems*. 2014, pp. 3626–3634.

[Wil14] A. G. Wilson. "Covariance kernels for fast automatic pattern discovery and extrapolation with Gaussian processes". PhD thesis. University of Cambridge, 2014.

[Wil+16] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. "Deep Kernel Learning". en. In: *AISTATS*. 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: (2017). arXiv: 1705.05495 [stat.ML].

[Wil+20] H. Wilde, J. Jewson, S. Vollmer, and C. Holmes. "Foundations of Bayesian Learning from Synthetic Data". In: (Nov. 2020). arXiv: 2011.08299 [cs.LG].

[Wil20] A. G. Wilson. "The Case for Bayesian Deep Learning". In: (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 Posteriors". 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*.

[Win+19] J. Winn, C. Bishop, T. Diethe, J. Guiver, and Y. Zaykov. *Model-based Machine Learning*. 2019.

[WIP20] J. Watson, A. Imohiosen, and J. Peters. "Active Inference or Control as Inference? A Unifying View". In: *International Workshop on Active Inference*. 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: (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.

[WJ21] Y. Wang and M. I. Jordan. *Desiderata for Representation Learning: A Causal Perspective*. 2021. arXiv: 2109.03795 [stat.ML].

[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 (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: (2021). arXiv: 2106.01121 [stat.ML].

[WL14a] N. Whiteley and A. Lee. "Twisted particle filters". en. In: *Annals of Statistics* 42.1 (Feb. 2014), pp. 115–141.

[WL14b] 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* (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.

[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].

[WN01] E. A. Wan and A. T. Nelson. "Dual EKF Methods". In: *Kalman Filtering and Neural Networks*. Ed. by S. Haykin. Wiley, 2001.

[WN07] D. Wipf and S. Nagarajan. "A new view of automatic relevancy determination". In: *NIPS*. 2007.

[WN10] D. Wipf and S. Nagarajan. "Iterative Reweighted ℓ_1 and ℓ_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*. ICML'15. 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: (2018). arXiv: 1801.03851 [cs.LG].

[WO12] M. Wiering and M. van Otterlo, eds. *Reinforcement learning: State-of-the-art*. Springer, 2012.

[Wol+21] M. Wołczyk, M. Zając, R. Pascanu, Ł. Kuciński, and P. Miłoś. "Continual World: A Robotic Benchmark For Continual Reinforcement Learning". In: *NIPS*. 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+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: (2019). arXiv: 1906.05827 [cs.LG].

[WP18] H. Wang and H. Poon. "Deep Probabilistic Logic: A Unifying Framework for Indirect Supervision". In: *EMNLP*. 2018.

[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.

[WSL19] R. L. Wasserstein, A. L. Schirm, and N. A. Lazar. "Moving to a World Beyond "$p < 0.05$"". In: *The American Statistician* 73.sup1 (2019), pp. 1–19.

[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.

[WSS21] W. J. Wilkinson, S. Särkkä, and A. Solin. "Bayes-Newton Methods for Approximate Bayesian Inference with PSD Guarantees". In: (2021). arXiv: 2111.01721 [stat.ML].

[WT01] M. Welling and Y.-W. Teh. "Belief Optimization for Binary Networks: a 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 (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.

[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).

[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.

[Wüt+16] M. Wüthrich, S. Trimpe, C. Garcia Cifuentes, D. Kappler, and S. Schaal. "A new perspective and extension of the Gaussian Filter". en. In: *The International Journal of Robotics Research* 35.14 (2016), pp. 1731–1749.

[WW12] Y. Wu and D. P. Wipf. "Dual-Space Analysis of the Sparse Linear Model". In: *NIPS*. 2012.

[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.

[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.

[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.

[XC19] Z. Xia and A. Chakrabarti. "Training Image Estimators without Image Ground-Truth". In: *NIPS*. 2019.

[XD18] J. Xu and G. Durrett. "Spherical Latent Spaces for Stable Variational Autoencoders". In: *EMNLP*. 2018.

[Xia+21a] K. Xia, K.-Z. Lee, Y. Bengio, and E. Bareinboim. "The Causal-Neural Connection: Expressiveness, Learnability, and Inference". In: *NIPS*. July 2021.

[Xia+21b] K. Xiao, L. Engstrom, A. Ilyas, and A. Madry. "Noise or Signal: The Role of Image Backgrounds in Object Recognition". In: *ICLR*. 2021.

[Xie+16] J. Xie, Y. Lu, S.-C. Zhu, and Y. N. Wu. "A Theory of Generative ConvNet". In: *ICML*. 2016.

[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.

[Xie+22] S. M. Xie, A. Raghunathan, P. Liang, and T. Ma. "An Explanation of In-context Learning as Implicit Bayesian Inference". In: *ICLR*. 2022.

[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.

[XKV22] Z. Xiao, K. Kreis, and A. Vahdat. "Tackling the Generative Learning Trilemma with Denoising Diffusion GANs". In: *ICLR*. 2022.

[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.

[Xu18] J. Xu. "Distance-based Protein Folding Powered by Deep Learning". In: (2018). arXiv: `1811.03481 [q-bio.BM]`.

[Yad+18] S. Yadlowsky, H. Namkoong, S. Basu, J. Duchi, and L. 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]`.

[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]`.

[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)*. 2017, pp. 685–691.

[Yan19] G. Yang. "Scaling Limits of Wide Neural Networks with Weight Sharing: Gaussian Process Behavior, Gradient Independence, and Neural Tangent Kernel Derivation". In: (2019). arXiv: `1902.04760 [cs.NE]`.

[Yan+21] J. Yang, K. Zhou, Y. Li, and Z. Liu. "Generalized OOD Detection: A Survey". In: (2021).

[Yan+22] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, Y. Shao, W. Zhang, B. Cui, and M.-H. Yang. "Diffusion Models: A Comprehensive Survey of Methods and Applications". In: (Sept. 2022). arXiv: `2209.00796 [cs.LG]`.

[Yan74] H Yanai. "Unification of various techniques of multivariate analysis by means of generalized coefficient of determination (GCD)". In: *J. Behaviormetrics* 1.1 (1974), pp. 45–54.

[Yan81] M. Yannakakis. "Computing the minimum fill-in is NP-complete". In: *SIAM J. Alg. Discrete Methods* 2 (1981), pp. 77–79.

[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 (2018), pp. 917–1007.

[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.

[YBM20] Y. Yang, R. Bamler, and S. Mandt. *Improving Inference for Neural Image Compression*. 2020. arXiv: 2006.04240 [eess.IV].

[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].

[Yed11] J. S. Yedidia. "Message-Passing Algorithms for Inference and Optimization". In: *J. Stat. Phys.* 145.4 (2011), pp. 860–890.

[Yeh+18] C.-K. Yeh, J. S. Kim, I. E. H. Yen, and P. Ravikumar. *Representer Point Selection for Explaining Deep Neural Networks*. 2018. arXiv: 1811.09720 [cs.LG].

[Yeh+19a] C.-K. Yeh, C.-Y. Hsieh, A. Suggala, D. I. Inouye, and P. K. Ravikumar. "On the (in) fidelity and sensitivity of explanations". In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 10967–10978.

[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).

[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"*. 2017.

[Yeu91a] R. W. Yeung. "A new outlook on Shannon's information measures". In: *IEEE Trans. Inf. Theory* 37.3 (1991), pp. 466–474.

[Yeu91b] R. W. Yeung. "A new outlook on Shannon's information measures". In: *IEEE Trans. on Information Theory* 37 (1991), pp. 466–474.

[YFW00] J. Yedidia, W. T. Freeman, and Y. Weiss. "Generalized Belief Propagation". In: *NIPS*. 2000.

[YH21] G. Yang and E. J. Hu. "Feature Learning in Infinite-Width Neural Networks". In: *ICML*. 2021.

[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.

[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.

[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.

[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: (2020). arXiv: 2006.10974 [cs.LG].

[YK06] A. Yuille and D. Kersten. "Vision as Bayesian inference: analysis by synthesis?" en. In: *Trends Cogn. Sci.* 10.7 (2006), pp. 301–308.

[YK19] M. Yang and B. Kim. *Benchmarking Attribution Methods with Relative Feature Importance*. 2019. arXiv: 1907.09701 [cs.LG].

[YMT22] Y. Yang, S. Mandt, and L. Theis. "An Introduction to Neural Data Compression". In: (2022). arXiv: 2202.06533 [cs.LG].

[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*. 2018.

[You19] A. Young. "Consistency without inference: Instrumental variables in practical application". In: (2019).

[You89] L. Younes. "Parameter estimation for imperfectly observed Gibbsian fields". In: *Probab. Theory and Related Fields* 82 (1989), pp. 625–645.

[You99] 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.

[YS10] G. A. Young and R. L. Smith. *Essentials of Statistical Inference*. en. Illustrated edition. Cambridge University Press, Mar. 2010.

[Yu+06] S. Yu, K. Yu, V. Tresp, K. H-P., and M. Wu. "Supervised probabilistic principal component analysis". In: *KDD*. 2006.

[Yu10] S.-Z. Yu. "Hidden Semi-Markov Models". In: *Artificial Intelligence J.* 174.2 (2010).

[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. 1975–1983.

[Yu+17] L. Yu, W. Zhang, J. Wang, and Y. Yu. "Seqgan: Sequence generative adversarial nets with policy gradient". In: *Thirty-first AAAI conference on artificial intelligence*. 2017.

[Yu+18] Y. Yu et al. "Dynamic Control Flow in Large-Scale Machine Learning". In: *Proceedings of the Thirteenth EuroSys Conference*. EuroSys '18. Association for Computing Machinery, 2018.

[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).

[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: (2021). arXiv: 2110.04627 [cs.CV].

[Yu+22] J. Yu et al. "Scaling Autoregressive Models for Content-Rich Text-to-Image Generation". 2022.

[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 (2019), pp. 2805–2824.

[Yui01] A. Yuille. "CCCP algorithms to minimze the Bethe and Kikuchi free energies: convergent alternatives to belief propagation". In: *Neural Computation* 14 (2001), pp. 1691–1722.

[YW04] C. Yanover and Y. Weiss. "Finding the M Most Probable Configurations in Arbitrary Graphical Models". In: *NIPS*. 2004.

[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.

[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 fuer Informatik, 2019, 69:1–69:21.

[YZ22] Y. Yang and P. Zhai. “Click-through rate prediction in online advertising: A literature review”. In: *Inf. Process. Manag.* 59.2 (2022), p. 102853.

[ZA12] J. Zou and R. Adams. “Priors for Diversity in Generative Latent Variable Models”. In: *NIPS*. 2012.

[Zaf+22] M. Zaffran, A. Dieuleveut, O. Féron, Y. Goude, and J. Josse. “Adaptive Conformal Predictions for Time Series”. In: (2022). arXiv: 2202.07282 [stat.ML].

[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: (2020). arXiv: 2006.08573 [cs.LG].

[Zan21] N. Zanichelli. *IAML Distill Blog: Transformers in Vision*. 2021.

[ZB18] T. Zhou and J. Bilmes. “Minimax curriculum learning: Machine teaching with desirable difficulties and scheduled diversity”. In: *International Conference on Learning Representations*. 2018.

[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.

[Zbo+21] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. “Barlow twins: Self-supervised learning via redundancy reduction”. In: *arXiv preprint arXiv:2103.03230* (2021).

[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.

[ZE01a] B. Zadrozny and C. Elkan. “Obtaining calibrated probability estimaets from decision trees and naive Bayesian classifiers”. In: *ICML*. 2001.

[ZE01b] B. Zadrozny and C. Elkan. “Transforming classifier scores into accurate multiclass probability estimates”. In: *KDD*. 2001.

[Zec+18a] 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”. en. In: *PLoS Med.* 15.11 (Nov. 2018), e1002683.

[Zec+18b] 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.

[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.

[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.

[Zen+18] C. Zeno, I. Golan, E. Hoffer, and D. Soudry. “Task Agnostic Continual Learning Using Online Variational Bayes”. In: (2018). arXiv: 1803.10123 [stat.ML].

[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 (2021), pp. 3139–3177.

[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.

[ZF14] M. D. Zeiler and R. Fergus. “Visualizing and understanding convolutional networks”. In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[ZFV20] G. Zeni, M. Fontana, and S. Vantini. “Conformal Prediction: a Unified Review of Theory and New Challenges”. In: (2020). arXiv: 2005.07972 [cs.LG].

[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.

[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.

[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.

[Zha+13b] K. Zhang, B. Scholkopf, K. Muandet, and Z. Wang. “Domain Adaptation under Target and Conditional Shift”. In: *ICML*. Vol. 28. 2013.

[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).

[Zha+17] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. “Understanding deep learning requires rethinking generalization”. In: *ICLR*. 2017.

[Zha+18] G. Zhang, S. Sun, D. Duvenaud, and R. Grosse. “Noisy Natural Gradient as Variational Inference”. In: *ICML*. 2018.

[Zha+19a] X. Zhai, J. Puigcerver, A. Kolesnikov, P. Ruyssen, C. Riquelme, M. Lucic, J. Djolonga, A. S. Pinto, M. Neumann, A. Dosovitskiy, et al. “A large-scale study of representation learning with the visual task adaptation benchmark”. In: *arXiv preprint arXiv:1910.04867* (2019).

[Zha+19b] C. Zhang, J. Butepage, H. Kjellstrom, and S. Mandt. “Advances in Variational Inference”. In: *IEEE PAMI* (2019), pp. 2008–2026.

[Zha+19c] 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.

[Zha+19d] L. Zhao, K. Korovina, W. Si, and M. Cheung. "Approximate inference with Graph Neural Networks". 2019.

[Zha+20a] A. Zhang, Z. Lipton, M. Li, and A. Smola. *Dive into deep learning*. 2020.

[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.

[Zha+20c] R. Zhang, B. Dai, L. Li, and D. Schuurmans. "GenDICE: Generalized Offline Estimation of Stationary Values". In: *ICLR*. 2020.

[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.

[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).

[Zha+20f] Y. Zhang, X. Chen, Y. Yang, A. Ramamurthy, B. Li, Y. Qi, and L. Song. "Efficient Probabilistic Logic Reasoning with Graph Neural Networks". In: *ICLR*. 2020.

[Zha+21] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer. "Scaling vision transformers". In: *arXiv preprint arXiv:2106.04560* (2021).

[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.

[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.

[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.

[Zho20] G. Zhou. "Mixed Hamiltonian Monte Carlo for Mixed Discrete and Continuous Variable". In: (2020). arXiv: 1909.04852 [stat.CO].

[ZHT06] H. Zou, T. Hastie, and R. Tibshirani. "Sparse principal component analysis". In: *JCGS* 15.2 (2006), pp. 262–286.

[Zhu+17] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *ICCV*. 2017.

[Zhu+18] X. Zhu, A. Singla, S. Zilles, and A. N. Rafferty. "An overview of machine teaching". In: *arXiv preprint arXiv:1801.05927* (2018).

[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).

[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.

[ZIE16] R. Zhang, P. Isola, and A. A. Efros. "Colorful image colorization". In: *European conference on computer vision*. Springer. 2016, pp. 649–666.

[ZIE17] R. Zhang, P. Isola, and A. A. Efros. "Split-brain autoencoders: Unsupervised learning by cross-channel prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1058–1067.

[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.

[Ziy+19] L. Ziyin, Z. Wang, P. P. Liang, R. Salakhutdinov, L.-P. Morency, and M. Ueda. "Deep gamblers: Learning to abstain with portfolio theory". In: *NIPS*. June 2019.

[ZL21] A. Zhou and S. Levine. "Training on Test Data with Bayesian Adaptation for Covariate Shift". In: *NIPS*. 2021.

[ZLF21] M. Zhang, S. Levine, and C. Finn. "MEMO: Test Time Robustness via Adaptation and Augmentation". In: (2021). arXiv: 2110.09506 [cs.LG].

[ZLG20] R. Zivan, O. Lev, and R. Galiki. "Beyond Trees: Analysis and Convergence of Belief Propagation in Graphs with Multiple Cycles". In: *AAAI*. 2020.

[ZMB21] B. Zhao, K. R. Mopuri, and H. Bilen. "Dataset Condensation with Gradient Matching". In: *International Conference on Learning Representations*. 2021.

[ZMG19] G. Zhang, J. Martens, and R. B. Grosse. "Fast Convergence of Natural Gradient Descent for Over-Parameterized Neural Networks". In: *NIPS*. 2019, pp. 8082–8093.

[ZML16] J. J. Zhao, M. Mathieu, and Y. LeCun. "Energy-based Generative Adversarial Network". In: (2016).

[Zoe07] O. Zoeter. "Bayesian generalized linear models in a terabyte world". In: *Proc. 5th International Symposium on image and Signal Processing and Analysis*. 2007.

[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.

[ZP00] G. Zweig and M. Padmanabhan. "Exact alpha-beta computation in logarithmic space with application to map word graph construction". In: *ICSLP*. 2000.

[ZP96] N. Zhang and D. Poole. "Exploiting causal independence in Bayesian network inference". In: *JAIR* (1996), pp. 301–328.

[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.

[ZR19b] Z. M. Ziegler and A. M. Rush. "Latent Normalizing Flows for Discrete Sequences". In: *ICML*. 2019.

[ZSB19] Q. Zhao, D. S. Small, and B. B. Bhattacharya. "Sensitivity analysis for inverse probability weighting estimators via the percentile bootstrap". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 81.4 (2019), pp. 735–761. eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/rssb.12327.

[ZSE19] S. Zhao, J. Song, and S. Ermon. "InfoVAE: Information Maximizing Variational Autoencoders". In: *AAAI*. 2019.

[ZSH22] Z. Zhang, E. Strubell, and E. Hovy. "A Survey of Active Learning for Natural Language Processing". In: *EMNLP*. Oct. 2022.

[ZW11] D Zoran and Y Weiss. "From learning models of natural image patches to whole image restoration". In: *ICCV*. 2011.

[ZW12] D. Zoran and Y. Weiss. "Natural Images, Gaussian Mixtures and Dead Leaves". In: *NIPS*. 2012, pp. 1736–1744.

[ZWM97] C. S. Zhu, N. Y. Wu, and D. Mumford. "Minimax Entropy Principle and Its Application to Texture Modeling". In: *Neural Computation* 9.8 (1997).

[ZY08] J.-H. Zhao and P. L. H. Yu. "Fast ML Estimation for the Mixture of Factor Analyzers via an ECM Algorithm". In: *IEEE. Trans. on Neural Networks* 19.11 (2008).

[ZY21] Y. Zhang and Q. Yang. "A Survey on Multi-Task Learning". In: *IEEE Trans. Knowl. Data Eng.* (2021).

[ZY97] Z. Zhang and R. W. Yeung. "A non-Shannon-type conditional inequality of information quantities". In: *IEEE Transactions on Information Theory* 43.6 (1997), pp. 1982–1986.

[ZY98] Z. Zhang and R. W. Yeung. "On characterization of entropy function via information inequalities". In: *IEEE Transactions on Information Theory* 44.4 (1998), pp. 1440–1452.