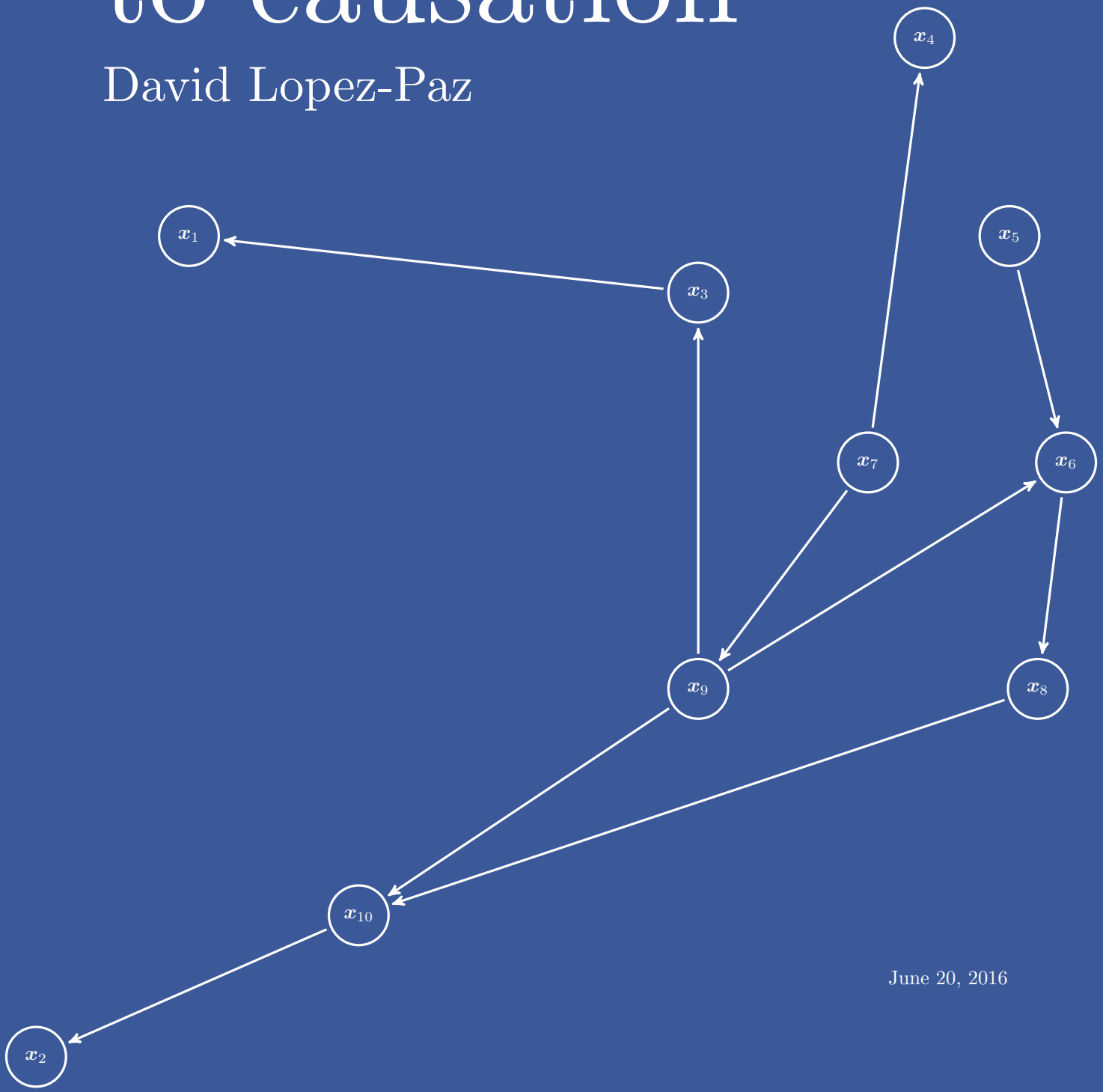
From dependence to causation

David Lopez-Paz



June 20, 2016

From Dependence to Causation

David Lopez-Paz

Abstract

Machine learning is the science of discovering statistical dependencies in data, and the use of those dependencies to perform predictions. During the last decade, machine learning has made spectacular progress, surpassing human performance in complex tasks such as object recognition, car driving, and computer gaming. However, the central role of prediction in machine learning avoids progress towards general-purpose artificial intelligence. As one way forward, we argue that *causal inference* is a fundamental component of human intelligence, yet ignored by learning algorithms.

Causal inference is the problem of uncovering the cause-effect relationships between the variables of a data generating system. Causal structures provide understanding about how these systems behave under changing, unseen environments. In turn, knowledge about these causal dynamics allows to answer "what if" questions, describing the potential responses of the system under hypothetical manipulations and interventions. Thus, understanding cause and effect is one step from machine learning towards machine reasoning and machine intelligence. But, currently available causal inference algorithms operate in specific regimes, and rely on assumptions that are difficult to verify in practice.

This thesis advances the art of causal inference in three different ways. First, we develop a framework for the study of statistical dependence based on copulas (models NPRV and GPRV) and random features (models RCA and RDC). Second, we build on this framework to interpret the problem of causal inference as the task of distribution classification. This new interpretation conceives a family of new causal inference algorithms (model RCC), which are widely applicable under mild learning theoretical assumptions. Third, we showcase RCC to discover causal structures in convolutional neural network features. All of the algorithms presented in this thesis are applicable to big data, exhibit strong theoretical guarantees, and achieve state-of-the-art performance in a variety of real-world benchmarks.

This thesis closes with a discussion about the state-of-affairs in machine learning research, and a review about the current progress on novel ideas such as machines-teaching-machines paradigms, theory of nonconvex optimization, and the supervision continuum. We have tried to provide our exposition with a philosophical flavour, as well as to make it a self-contained book.

Contents

List of Figures

List of Tables

List of Definitions

List of Theorems

List of Remarks

List of Examples

Notation

symbol	meaning
a	scalar or vector
a_i	entry at ith position of vector a
$a_{\mathcal{I}}$	vector $(a_i)_{i \in \mathcal{I}}$, for set of indices $\mathcal{I}$
A	matrix or tensor
$A_{i,j}$	entry at ith row and jth column of the matrix A
$A_{i,:}$	row vector from the ith row of the matrix A
$A_{:,j}$	column vector from the jth column of the matrix A
$A_{i,j,k}$	similar notations apply to higher-order tensors
$\mathcal{A}$	set
$\{x_i\}_{i=1}^n$	set $\{x_1, \ldots, x_n\}$
$\mathbb{R}$	the set of real numbers
$\mathbb{R}^n$	the set of vectors of size n with real entries
$\mathbb{R}^{n \times m}$	the set of matrices of size $n \times m$ with real entries
$\mathbb{R}^{n \times m \times d}$	similar notations apply to tensors
$\boldsymbol{a}$	scalar-valued or vector-valued random variable
$\boldsymbol{A}$	matrix-valued or tensor-valued random variable
$\boldsymbol{a} \equiv P$	$\boldsymbol{a}$ follows the distribution P
$a \sim P$	a is sampled from P
P^n	n-dimensional product distribution built from P
$\mathbb{P}_{\boldsymbol{x}}(e)$	probability of event e
$\mathbb{E}_{\boldsymbol{x}}[f(\boldsymbol{x})]$	expectation of $f(\boldsymbol{x})$ over the distribution P.
$\mathbb{V}_{\boldsymbol{x}}[f(\boldsymbol{x})]$	variance of $f(\boldsymbol{x})$.
$\boldsymbol{x} \perp\!\!\!\perp \boldsymbol{y}$	$\boldsymbol{x}$ is independent from $\boldsymbol{y}$
$\boldsymbol{x} \perp\!\!\!\perp \boldsymbol{y} \mid \boldsymbol{z}$	$\boldsymbol{x}$ is conditionally independent from $\boldsymbol{y}$ given $\boldsymbol{z}$
$\boldsymbol{x} \to \boldsymbol{y}$	$\boldsymbol{x}$ causes $\boldsymbol{y}$

The elements $p(\boldsymbol{x}) = p$ are the probability density *function* of $\boldsymbol{x}$. On the other hand, the elements $p(\boldsymbol{x} = x) = p(x)$ are the *value* of the probability density function at x. The same notations apply to cumulative distribution functions, denoted with an upper case P.

Table 1: Notations.

Chapter 1

Introduction

As put forward by David Hume over three centuries ago, our experience is shaped by the observation of constant conjunction of events. Rain follows drops in atmospheric pressure, sunlight energizes our mornings with warmth, the orbit of the Moon dances with the tides of the sea, mirrors shatter into pieces when we throw stones at them, heavy smokers suffer from cancer, our salary relates to the car we drive, and bad political decisions collapse stock markets. Such systematic variations suggest that these pairs of variables rely on each other to instantiate their values. These variables, we say, *depend* on each other.

Dependence is the necessary substance for statistics and machine learning. It relates the variables populating our world to each other, and enables the prediction of values for some variables given values taken by others. Let me exemplify. There exists a strong linear dependence between the chocolate consumption and the amount of Nobel laureates per country (Messerli, 2012). Therefore, we could use the data about these two variables from a small amount of countries to construct a linear function from chocolate consumption to number of Nobel laureates. Using this linear function we could, given the chocolate consumption in a new country, predict their national Nobel prize sprout. Dependencies like these leave patterns in the joint probability distribution of the variables under study. The goal of machine learning and statistics is then, as summarized by Vladimir Vapnik (1982), the inference of such patterns from empirical data, and their use to predict new aspects about such joint probability distribution.

But, how does dependence arise? The answer hides in the most fundamental of the connections between two entities: causation. According to the *principle of common cause* pioneered by Hans Reichenbach (1956), every dependence between two variables $\boldsymbol{x}$ and $\boldsymbol{y}$ is the observable footprint of one out of three possible causal structures: either $\boldsymbol{x}$ causes $\boldsymbol{y}$, or $\boldsymbol{y}$ causes $\boldsymbol{x}$, or there exists a third variable $\boldsymbol{z}$, called *confounder*, which causes both $\boldsymbol{x}$ and $\boldsymbol{y}$. The third structure reveals a major consequence: *dependence does not*

imply causation. Or, when the dependence between two variables $\boldsymbol{x}$ and $\boldsymbol{y}$ is due to a confounder $\boldsymbol{z}$, this dependence does not imply the existence of a causal relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$. Now, this explains the bizarre connection between chocolate eating and Nobel prize winning from the previous paragraph! It may be that this dependence arises due to the existence of an unobserved confounder: for example, the strength of the economy of the country.

The study of causation is not exclusive to philosophy, as it enjoys far-reaching consequences in statistics. While dependence is the tool to describe patterns about the distribution generating our data, causation is the tool to describe the reactions of these patterns when intervening on the distribution. In plain words, the difference between dependence and causation is the difference between *seeing* and *doing*. In terms of our running example: if we were a politician interested in increasing the number of Nobel prizes awarded to scientists from our country, the causal structure of the problem indicates that we should boost the national economy (the alleged common cause), instead of force-feeding chocolate to our fellow citizens. Thus, causation does not only describe which variables depend on which, but also how to manipulate them in order to achieve a desired effect.

More abstractly, causation bridges the distribution that generates the observed data to some different but related distribution, which is more relevant to answer the questions at hand (Peters, 2015). For instance, the question "Does chocolate consumption *cause* an increase in national Nobel laureates?" is not a question about the distribution generating the observed data. Instead, it is a question about a different distribution that we could obtain, for instance, by randomizing the chocolate consumption across countries. To answer the question we should check, after some decades of randomization, if the dependence between chocolates and Nobels remains in this new induced distribution. Although randomized experiments are considered the golden standard for causal inference, these are often unpractical, unethical, or impossible to realize. In these situations we face the need for *observational causal inference*: the skill to infer the causal structure of a data generating process without intervening on it.

As humans, we successfully leverage observational causal inference to reason about our changing world, and about the outcome of the interventions that we perform on it (*Will she reply if I text her?*). Observational causal inference is key to reasoning and intelligence (Bottou, 2014). Changing environments are a nuisance not only known to humans: machines face the same issues when dealing with changing distributions between training and testing times; multitask, domain adaptation, and transfer learning problems; and dynamic environments such as online learning and reinforcement learning. Causal inference is a promising tool to address these questions, yet ignored in most machine learning algorithms. Here we take a stance about the central importance of causal inference for artificial intelligence, and contribute to

the *cause* by developing novel theory and algorithms.

Let us begin this journey; one exploration into the fascinating concepts of statistical dependence and causation. We will equip ourselves with the necessary mathematical background in Part I. To understand causation one must first master dependence, so we will undertake this endeavour in Part II. Finally, Part III crosses the bridge from dependence to causation, and argues about the central role of the latter in machine learning, machine reasoning, and artificial intelligence. This thesis has a philosophical taste rare to our field of research; we hope that this is for the enjoyment of the reader.

Remark 1.1 *(The origin of dependence).* The word *dependence* originates from the Old French vocable *dependre*, which was first used around the 15^{th} century. The concept of statistical dependence was explicitly introduced in Abraham de Moivre's *The Doctrine of Chances* (1718), where he defines two events to be independent "when they have no connection one with the other, and that the happening of one neither forwards nor obstructs the happening of the other". On the other hand, he describes two events to be dependent "when they are so connected together as that the probability of either happening alters the happening of the other". In the same work, de Moivre's correctly calculates the joint probability of two independent events as the product of their marginal probabilities. Gerolamo Cardano (1501-1576) hinted the multiplication rule before, but not explicitly. The first precise mathematical characterization of statistical dependence is Pierre-Simon Laplace's *Théorie analytique des probabilités*, in 1812. ◊

1.1 Outline

The rest of this thesis is organized in seven chapters.

1. Chapter 2 introduces the necessary mathematics to understand this thesis. We will review well known but important results about linear algebra, probability theory, machine learning, and numerical optimization.

2. Chapter 3 reviews four techniques to represent data for its analysis: kernel methods, random features, neural networks, and ensembles. Data representations will be a basic building block to study statistical dependence and causation throughout this thesis. Chapters 2 and 3 are a personal effort to make this thesis a self-contained book.

3. Chapter 4 starts the study of statistical dependence by means of *generative models of dependence*, which estimate the full dependence structure of a multidimensional probability distribution. This chapter contains novel material from (Lopez-Paz et al., 2012, 2013b), where cited.

4. Chapter 5 concerns *discriminative models of dependence*, which, in contrast to generative models, summarize the dependence structure of a multidimensional probability distribution into a low-dimensional statistic. This chapter contains novel material from (Lopez-Paz et al., 2013a; Lopez-Paz et al., 2014), where cited.

5. Chapter 6 crosses the bridge from dependence to causation, introducing the language of causal modeling, and reviewing the state-of-the-art on algorithms for observational causal inference. This chapter contains novel material from (Hernández-Lobato et al., 2016), where cited.

6. Chapter 7 phrases observational causal inference as probability distribution classification. Under this interpretation, we describe a new framework of observational causal inference algorithms, which exhibit provable guarantees and state-of-the-art performance. Furthermore, we apply our algorithms to infer the existence of causal signals in convolutional neural network features. This chapter contains novel material from (Lopez-Paz et al., 2015, 2016b,c), where cited.

7. Chapter 8 closes the exposition with some reflections on the state-of-affairs in machine learning research, as well as some preliminary progress on three research questions: machines-teaching-machines paradigms, theory of nonconvex optimization, and the supervision continuum. This chapter contains novel material from (Lopez-Paz et al., 2016a), where cited.

The code implementing all the algorithms and experiments presented in this thesis is available at `https://github.com/lopezpaz`.

1.2 Contributions

We summarize the contributions contained in this thesis, as well as their location in the text, in both Table 1.1 and the corresponding back-references from the Bibliography. Most of these are works in collaboration with extraordinary scientists, including my wonderful advisors Bernhard Schölkopf and Zoubin Ghahramani. Our contributions are:

1. We introduce nonparametric vine copulas (NPRV), and their use to address semisupervised domain adaptation problems (Lopez-Paz et al., 2012). Vine copulas factorize multivariate densities into a product of marginal distributions and bivariate copula functions. Therefore, each of these factors can be adapted independently to learn from different domains. Experimental results on regression problems with real-world data illustrate the efficacy of the proposed approach when compared to the state-of-the-art.

2. We relax the "vine simplifying assumption" by modeling the latent functions that specify the shape of a conditional copula given its conditioning variables (Lopez-Paz et al., 2013b). We learn these functions by bringing sparse Gaussian processes and expectation propagation into the world of vines. We term our method GPRV. Our experiments show that modeling these previously ignored conditional dependencies leads to better estimates of the copula of the data.

3. We propose the Randomized Component Analysis (RCA) framework (Lopez-Paz et al., 2014). RCA extends linear component analysis algorithms, such as principal component analysis and canonical correlation analysis, to model nonlinear dependencies. We stablish theoretical guarantees for RCA using recent concentration inequalities for matrix-valued random variables, and provide numerical simulations that show the state-of-the-art performance of the proposed algorithms.

4. We extend the RCA framework into the Randomized Dependence Coefficient (RDC), a measure of dependence between multivariate random variables (Lopez-Paz et al., 2013a). RDC is invariant with respect to monotone transformations in marginal distributions, runs in log-linear time, has provable theoretical guarantees, and is easy to implement. RDC has a competitive performance when compared to the state-of-the-art measures of dependence.

5. We extend RCA to pose causal inference as the problem of learning to classify probability distributions (Lopez-Paz et al., 2015, 2016b). In particular, we will featurize samples from probability distributions using the kernel mean embedding associated with some characteristic kernel. Using these embeddings, we train a binary classifier (the Randomized Causation Coefficient or RCC) to distinguish between causal structures. We present generalization bounds showing the statistical consistency and learning rates of the proposed approach, and provide a simple implementation that achieves state-of-the-art cause-effect inference. Furthermore, we extend RCC to multivariate causal inference.

6. We propose a variant of RCC based on neural networks, termed NCC. We use NCC to reveal the existence of observable causal signals in computer vision features. In particular, NCC effectively separates contextual features from object features in collections of static images (Lopez-Paz et al., 2016c). This separation proves the existence of a relation between the direction of causation and the difference between objects and their context, as well as the existence of observable causal signals in collections of static images.

7. We introduce generalized distillation (Lopez-Paz et al., 2016a), a framework to learn from multiple data modalities and machines semisuper-

visedly. Compression (Buciluǎ et al., 2006), distillation (Hinton et al., 2015) and privileged information (Vapnik and Vashist, 2009) are shown particular instances of generalized distillation.

8. In our conclusion chapter, we provide research discussions about the concepts of supervision continuum and the theory of nonconvex optimization.

9. We provide a self-contained exposition, which provides all the necessary mathematical background, and allows to read this thesis as a book.

publication	cited in
Semi-Supervised Domain Adaptation with Non-Parametric Copulas David Lopez-Paz, José Miguel Hernández-Lobato and Bernhard Schölkopf NIPS, 2012 (Lopez-Paz et al., 2012)	Sections 4.4.6, 4.6.2
Gaussian Process Vine Copulas for Multivariate Dependence David Lopez-Paz, José Miguel Hernández-Lobato and Zoubin Ghahramani ICML, 2013 (Lopez-Paz et al., 2013b)	Sections 4.4.5, 4.6.1
The Randomized Dependence Coefficient David Lopez-Paz, Philipp Hennig and Bernhard Schölkopf NIPS, 2013 (Lopez-Paz et al., 2013a)	Section 5.2, 5.4
Two Numerical Models of Saturn Rings Temperature as Measured by Cassini Nicolas Altobelli, David Lopez-Paz et al. Icarus, 2014 (Altobelli et al., 2014)	—
Randomized Nonlinear Component Analysis David Lopez-Paz, Suvrit Sra, Alex Smola, Zoubin Ghahramani and Bernhard Schölkopf ICML, 2014 (Lopez-Paz et al., 2014)	Section 5.1, 5.4
The Randomized Causation Coefficient David Lopez-Paz, Krikamol Muandet and Benjamin Recht JMLR, 2015 (Lopez-Paz et al., 2016b)	Chapter 7
Towards A Learning Theory of Cause-Effect Inference David Lopez-Paz, Krikamol Muandet, Bernhard Schölkopf and Iliya Tolstikhin ICML, 2015 (Lopez-Paz et al., 2015)	Chapter 7
No Regret Bound for Extreme Bandits Robert Nishihara, David Lopez-Paz and Léon Bottou AISTATS, 2016 (Nishihara et al., 2016)	Section 3.3.3
Non-linear Causal Inference using Gaussianity Measures Daniel Hernandez-Lobato, Pablo Morales Mombiela, David Lopez-Paz and Alberto Suarez JMLR, 2016 (Hernández-Lobato et al., 2016)	Section 6.4.2
Unifying distillation and privileged information David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, Vladimir Vapnik ICLR, 2016 (Lopez-Paz et al., 2016a)	Section 8.1
Discovering causal signals in images David Lopez-Paz, Robert Nishihara, Soumith Chintala, Bernhard Schölkopf, Léon Bottou Under review, 2016 (Lopez-Paz et al., 2016c)	—
Lower bounds for realizable transductive learning Ilya Tolstikhin, David Lopez-Paz Under review, 2016 (Tolstikhin and Lopez-Paz, 2016)	—

Table 1.1: Main publications of the author.

Part I

Background

Chapter 2

Mathematical preliminaries

This chapter is a review of well-known results.

This chapter introduces the necessary mathematics to understand this thesis. We will review well known but important results about linear algebra, probability theory, machine learning, and numerical optimization.

2.1 Linear algebra

This section studies vector spaces over the field of the real numbers. It reviews basic concepts about vectors and matrices, as well as their respective infinite-dimensional generalizations as functions and operators.

2.1.1 Vectors

Vectors $u \in \mathbb{R}^d$ are one-dimensional arrays of d numbers

$$u = (u_1, \ldots, u_d)^\top.$$

The *inner product* between two vectors $u, v \in \mathbb{R}^d$ is

$$\langle u,\, v \rangle_{\mathbb{R}^d} = \sum_{i=1}^{n} u_i v_i,$$

where we omit the subscript $\mathbb{R}^d$ whenever this causes no confusion. Using the inner product, we measure the "size" of a vector $u \in \mathbb{R}^d$ using its *norm*

$$\|u\| = \sqrt{\langle u,\, u \rangle}.$$

Using the norm, we define the *distance* between two vectors as

$$\|u - v\| = \sqrt{\langle u - v,\, u - v \rangle}.$$

Two vectors u, v are *orthogonal* if $\langle u, v\rangle = 0$. A vector u is an *unit vector* if $\|u\| = 1$. If two vectors are unit and orthogonal, they are *orthonormal*. For any two vectors $u, v \in \mathbb{R}^d$, the *Cauchy-Schwartz inequality* states that

$$|\langle u, v\rangle| \leq \|u\| \, \|v\|.$$

One consequence of the Cauchy-Schwartz inequality is the *triangle inequality*

$$\|u + v\| \leq \|u\| + \|v\|,$$

where the two previous inequalities are valid for all $u, v \in \mathbb{R}^d$.

The previous results hold for any norm, although we will focus in the Euclidean norm $\|\cdot\|_2 = \|\cdot\|$, one special case of the p-norm

$$\|x\|_p = \left(\sum_{i=1}^{d} |x_i|^p \right)^{1/p},$$

when $p = 2$.

2.1.2 Matrices

Real matrices $X \in \mathbb{R}^{n \times d}$ are two-dimensional arrangements of real numbers

$$X = \begin{pmatrix} X_{1,1} & \cdots & X_{1,d} \\ \vdots & \ddots & \vdots \\ X_{n,1} & \cdots & X_{n,d} \end{pmatrix}.$$

We call the vector $X_{i,:} \in \mathbb{R}^d$ the i-th row of X, the vector $X_{:,j} \in \mathbb{R}^n$ the j-th column of X, and the number $X_{i,j} \in \mathbb{R}$ the (i, j)-entry of X. Unless stated otherwise, vectors $u \in \mathbb{R}^d$ are *column matrices* $u \in \mathbb{R}^{d \times 1}$. We adopt the usual associative, distributive, but not commutative matrix multiplication. Such product of two matrices $A \in \mathbb{R}^{n \times d}$ and $B \in \mathbb{R}^{d \times m}$ has entries

$$(AB)_{i,j} = \sum_{k=1}^{d} A_{i,k} B_{k,j},$$

for all $1 \leq i \leq n$ and $1 \leq j \leq m$. We call the matrix $X^\top$ the *transpose* of X, and it satisfies $X^\top_{j,i} = X_{i,j}$ for all $1 \leq i \leq n$ and $1 \leq j \leq d$. Real matrices $X \in \mathbb{R}^{n \times d}$ are *square* if $n = d$, and *symmetric* if $X = X^\top$. *Orthogonal* matrices $X \in \mathbb{R}^{n \times n}$ have orthonormal vectors for rows and columns. *Unitary matrices* U satisfy $U^\top U = I$. The vector $\operatorname{diag}(X) = (X_{1,1}, \ldots, X_{\min(n,d),\min(n,d)})$ is the *diagonal* of the matrix $X \in \mathbb{R}^{n \times d}$. *Diagonal* matrices have nonzero elements only on their diagonal. The *identity* matrix $I_n \in \mathbb{R}^{n \times n}$ is the diagonal matrix with $\operatorname{diag}(I_n) = (1, \ldots, 1)$.

Real symmetric matrices $X \in \mathbb{R}^{n\times n}$ are *positive-definite* if $z^\top Xz > 0$ for all nonzero z, or if all its eigenvalues are positive. Similarly, a real symmetric matrix is *positive-semidefinite* if $z^\top Xz \geq 0$ for all nonzero z. For positive-definite matrices we write $X \succ 0$, and for positive-semidefinite matrices we write $X \succeq 0$. If X, Y and $X - Y$ are three positive-definite matrices, we may establish the *Löwner order* between X and Y and say $X \succ Y$. Positive semidefinite matrices X satisfy $X = SS =: S^2$ for an unique S, called the square root of X. Finally, if X is positive-definite, then $Q^\top XQ$ is positive-definite for all Q.

The *rank* of a matrix is the number of linearly independent columns. These are the columns of a matrix that we can not express as a linear combination of the other columns in that same matrix. Alternatively, the rank of a matrix is the dimensionality of the vector space spanned by its columns. The rank of a matrix is equal to the rank of its transpose. A matrix $X \in \mathbb{R}^{n\times d}$ is *full rank* if $\text{rank}(X) = \min(n, d)$.

Given a full rank matrix $X \in \mathbb{R}^{n\times n}$, we call the unique matrix B satisfying $AB = BA = I_n$ the *inverse matrix* of A. Orthogonal matrices satisfy $X^\top = X^{-1}$. Square diagonal matrices D have diagonal inverses D^{-1} with $\text{diag}(D^{-1}) = (D_{1,1}^{-1}, \ldots, D_{n,n}^{-1})$. Positive-definite matrices have positive-definite inverses. One useful matrix identity involving inverses is the *Sherman-Morrison-Woodbury formula*:

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1}VA^{-1}. \tag{2.1}$$

As we did with vectors, we can calculate the "size" of a matrix using its norm. There are a variety of matrix norms that we can use. *Subordinate norms* have form

$$\|A\|_{\alpha,\beta} = \sup\left\{\frac{\|Ax\|_\alpha}{\|x\|_\beta} : x \in \mathbb{R}^n, x \neq 0\right\},$$

where the norms $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$ are vector norms satisfying

$$\|Ax\|_\beta \leq \|A\|_{\alpha,\beta}\|x\|_\alpha.$$

We adopt the short hand notation $\|A\|_{\alpha,\alpha} = \|A\|_\alpha$. The particular case $\|A\|_2$ is the *operator norm*. Another example of matrix norms are *entrywise norms*:

$$\|A\|_{p,q} = \left(\sum_{i=1}^{n}\left(\sum_{j=1}^{d}|A_{i,j}|^p\right)^{q/p}\right)^{1/q}.$$

When $p = q = 2$, we call this norm the *Frobenius norm*. All matrix norms are equivalent, in the sense that, for two different matrix norms $\|\cdot\|_a$ and $\|\cdot\|_b$, there exists two finite constants c, C such that $c\|A\|_a \leq \|A\|_b \leq C\|A\|_a$, for all matrices A. The operator and Frobenius norms are two examples of

unitary invariant norms: $\|A\| = \|UA\|$ for any matrix A and unitary matrix U. Unless specified otherwise, $\|A\|$ will denote the operator norm of A.

Given a symmetric matrix $A \in \mathbb{R}^{n\times n}$ and a real number λ, the nonzero vectors v that satisfy

$$Av = \lambda v$$

are the *eigenvectors of* A. Each *eigenvector* v is orthogonal to the others, and has an *eigenvalue* λ associated with it. Geometrically, eigenvectors are those vectors that, when we applied to the linear transformation given by some matrix A, change their magnitude by λ but remain constant in direction. For symmetric matrices, eigenvectors and eigenvalues provide with the *eigendecomposition*

$$A = Q\Lambda Q^\top,$$

where the columns of $Q \in \mathbb{R}^{n\times n}$ are the eigenvectors of A, and the real entries of the diagonal matrix $\Lambda \in \mathbb{R}^{n\times n}$ contain the associated eigenvalues. By convention, we arrange the decomposition such that $\Lambda_{1,1} \geq \Lambda_{2,2} \geq \cdots \geq \Lambda_{n,n}$. If the matrix A is asymmetric, different formulas apply (Horn and Johnson, 2012). In short, the eigendecomposition of a matrix informs about the directions and magnitudes that the linear operation Ax shrinks or expands vectors x.

Singular values generalize the concept of eigenvectors, eigenvalues, and eigendecompositions to rectangular matrices. The singular value decomposition of matrix $A \in \mathbb{R}^{n\times d}$ is

$$A = U\Sigma V^\top,$$

where $U \in \mathbb{R}^{n\times n}$ is an orthogonal matrix whose columns we call the *left singular vectors* of A, $\Sigma \in \mathbb{R}^{n\times d}$ is a diagonal matrix whose positive entries we call the *singular values* of A, and $V \in \mathbb{R}^{d\times d}$ is an orthogonal matrix whose columns we call the *right singular vectors* of A. The eigenvalue and singular value decompositions relate to each other. The left singular vectors of A are the eigenvectors of $AA^\top$. The right singular vectors of A are the eigenvectors of $A^\top A$. The nonzero singular values of A are the square root of the nonzero eigenvalues of both $AA^\top$ and $A^\top A$. The operator norm relates to the largest eigenvalue $\Lambda_{1,1}$ and the largest singular value $\Sigma_{1,1}$ of a square matrix A as

$$\|A\|_2 = \sqrt{\Lambda_{1,1}} = \Sigma_{1,1}.$$

Thus, the operator norm upper bounds how much does the matrix A modify the norm of a vector.

The product of all the eigenvalues of a matrix A is the *determinant* $|A|$. The sum of all the eigenvalues of a matrix A is the *trace* $\mathrm{tr}(A)$. The trace is also equal to the sum of the elements in the diagonal of the matrix. The trace and the determinant are similarity-invariant: the trace and the determinant of two matrices A and $B^{-1}AB$ are the same, for all B.

For a more extensive exposition on matrix algebra, consult (Golub and Van Loan, 2012; Horn and Johnson, 2012; Petersen and Pedersen, 2012).

2.1.3 Functions and operators

Section 2.1.1 studied d-dimensional vectors, which live in the d-dimensional Euclidean space $\mathbb{R}^d$. These are vectors u with d components $u_1, \ldots, u_d$, indexed by the integers $\{1, \ldots, d\}$. In contrast, it is possible to define *infinite-dimensional vectors* or *functions*, which live in a infinite-dimensional *Hilbert Space*. A Hilbert space $\mathcal{H}$ is a vector space, equipped with an inner product $\langle f, g \rangle_{\mathcal{H}}$, such that the norm $\|f\| = \sqrt{\langle f, f \rangle_{\mathcal{H}}}$ turns $\mathcal{H}$ into a complete metric space.

The key intuition here is the analogy between infinite-dimensional vectors and functions. Let us consider the Hilbert space $\mathcal{H}$ of functions $f : \mathbb{R} \to \mathbb{R}$. Then, the infinite-dimensional vector or function $f \in \mathcal{H}$ has shape $f = (f(x))_{x \in \mathbb{R}}$, where the indices are now real numbers x, arguments to the function f.

Similarly, *linear operators* are the infinite-dimensional extension of matrices. While matrices $A \in \mathbb{R}^{n \times d}$ are linear transformations of vectors $u \in \mathbb{R}^d$ into vectors $Au \in \mathbb{R}^n$, linear operators $L : \mathcal{F} \to \mathcal{H}$ are linear transformations of functions $f \in \mathcal{H}$ into functions $g \in \mathcal{H}$. In the following, assume that $\mathcal{F}$ and $\mathcal{H}$ contain functions from $\mathcal{X}$ to $\mathbb{R}$. We say that the linear operator $L : \mathcal{F} \to \mathcal{H}$ is bounded if there exists a $c > 0$ such that

$$\|Lf\|_{\mathcal{H}} \leq c\|f\|_{\mathcal{F}},$$

for all nonzero $f \in \mathcal{F}$. A linear operator is bounded if and only if it is continuous. If a bounded operator L has finite *Hilbert-Schmidt* norm

$$\|L\|_{\mathrm{HS}}^2 = \sum_{i \in \mathcal{I}} \|Lf_i\|^2,$$

we say the operator is a *Hilbert-Schmidt operator*. In the previous, the set $\{f_i : i \in \mathcal{I}\}$ is an orthonormal basis on $\mathcal{F}$. Finally, we say that an operator T is an integral transform if it admits the expression

$$(Tf)(u) = \int K(t, u) f(t) \mathrm{d}t,$$

for some kernel function $K : \mathcal{X} \times \mathcal{X}$. For example, by choosing the kernel

$$K(t, u) = \frac{e^{-\imath u t}}{\sqrt{2\pi}},$$

there $\imath$ denotes the imaginary unit, we obtain the Fourier transform.

Most of the material presented for vectors and matrices extends to functions and operators: the Cauchy-Schwartz inequality, the triangle inequality, eigen and singular value decompositions, and so on. We recommend the monograph of Reed and Simon (1972) to learn more about functional analysis.

2.2 Probability theory

Probability theory studies *probability spaces*. A probability space is a triplet $(\Omega, \mathcal{B}(\Omega), \mathbb{P})$. Here, the *sample space* Ω is the collection of outcomes of a random experiment. For example, the sample space of a "coin flip" is the set $\Omega = \{\text{heads}, \text{tails}\}$. The *$\sigma$-algebra* $\mathcal{B}(\Omega)$ is a nonempty collection of subsets of Ω such that i) Ω is in $\mathcal{B}(\Omega)$, ii) if $A \in \mathcal{B}(\Omega)$, so is the complement of A, and iii) if A_n is a sequence of elements of $\mathcal{B}(\Omega)$, then the union of A_n is in $\mathcal{B}(\Omega)$. Using De Morgan's law, one can also see that if A_n is a sequence of elements of $\mathcal{B}(\Omega)$, then the intersection of A_n is in $\mathcal{B}(\Omega)$. The power set of Ω is the largest σ-algebra of Ω. In plain words, the σ-algebra $\mathcal{B}(\Omega)$ is the collection of all the events (subsets of the sample space) that we would like to consider. Throughout this thesis, $\mathcal{B}(\Omega)$ will be the *Borel* σ-algebra of Ω. The *probability measure* $\mathbb{P}$ is a function $\mathcal{B}(\Omega) \to [0, 1]$, such that $\mathbb{P}(\emptyset) = 0$, $\mathbb{P}(\Omega) = 1$, and $\mathbb{P}(\cup_i A_i) = \sum_i \mathbb{P}(A_i)$ for all countable collections $\{A_i\}$ of pairwise disjoint sets A_i. For a fair coin, we could have $\mathbb{P}(\{\text{heads}\}) = \mathbb{P}(\{\text{tails}\}) = \frac{1}{2}$, $\mathbb{P}(\emptyset) = 0$ and $\mathbb{P}(\{\text{heads}, \text{tails}\}) = 1$.

Remark 2.1 *(Interpretations of probability).* There are two main interpretations of the concept of *probability*. *Frequentist probability* is the limit of the relative frequency of an event. For instance, if we get heads $h(n)$ times in n tosses of the same coin, the frequentist probability of the event "heads" is $\lim_{n\to\infty} h(n)/n$. On the other hand, *Bayesian probability* measures the degree of belief or plausibility of a given event. One way to understand the difference between the two is that frequentism considers data a random quantity used to infer a fixed parameter. Conversely, Bayesianism considers data a fixed quantity used to infer the distribution of a random parameter.

We say that Frequentist interpretations of probability are *objective*, since they rely purely on the observation of repetition of events. Conversely, Bayesian interpretations of probability are *subjective*, since they combine observations of events with prior beliefs not contained in the data nor the statistical model. It is beneficial to see both approaches as complementary: frequentist methods offer a formalism to study repeatable phenomena, and Bayesian methods offer a formalism to replace repeatability with uncertainty modeled as subjective probabilities. ♢

Probability spaces $(\Omega, \mathcal{B}(\Omega), \mathbb{P})$ are the basic building blocks to define *random variables*. Random variables take different values at random, each of them with probability given by the probability measure $\mathbb{P}$. More specifically, let $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$ be some measurable space. Then, a random variable taking values in $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$ is a $(\mathcal{B}(\Omega), \mathcal{B}(\mathcal{X}))$-measurable function $\boldsymbol{x} \colon \Omega \to \mathcal{X}$. For example, consider real-valued random variables, that is $\mathcal{X} = \mathbb{R}$. Then, the answer to the question "What is the probability of the random variable $\boldsymbol{x}$ taking the value $3.5 \in \mathbb{R}$?" is

$$\mathbb{P}(\{\omega \in \Omega : \boldsymbol{x}(\omega) = 3.5\}) =: \mathbb{P}(\boldsymbol{x} = 3.5).$$

Intuitively, random variables measure some property of an stochastic system. Then, the probability of the stochastic system $\boldsymbol{x}$ taking a particular value $x \in \mathcal{X}$ is the probability of the set of possible outcomes $\omega \in \Omega$ satisfying $\boldsymbol{x}(\omega) = x$. This thesis studies dependence and causation by characterizing sets of *random variables* and their relationships. For a cheat sheet on statistics, see (Vallentin, 2015).

2.2.1 Single random variables

We are often interested in the probability of a random variable taking values over a certain range. *Cumulative distribution functions* use probability measures to compute such probabilities.

Definition 2.1 (Cumulative distribution function)**.** *The cumulative distribution function (cdf) or distribution of a real random variable $\boldsymbol{x}$ is*

$$P(\boldsymbol{x} = x) = P(x) = \mathbb{P}(\boldsymbol{x} \leq x).$$

Distribution functions are nondecreasing and right-continuous. If the cdf P is strictly increasing and continuous, the inverse cdf P^{-1} is the *quantile* function. One simple way to estimate distributions from data is to use the empirical measure.

Definition 2.2 (Empirical measure)**.** *Consider the sample $x_1, \ldots, x_n \sim P(\boldsymbol{x})$. Then, the empirical probability measure of this sample is*

$$\mathbb{P}_n(\omega) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(x_i \in \omega)$$

for all events $\omega \subseteq \Omega$.

One central use of the empirical measure is to define the empirical distribution function:

Definition 2.3 (Empirical cumulative distribution function)**.** *The empirical cumulative distribution function (ecdf) of $x_1, \ldots, x_n \sim P(\boldsymbol{x})$ is*

$$P_n(\boldsymbol{x} = x) = \mathbb{P}_n((-\infty, x]) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(x_i \leq x).$$

The ecdf converges uniformly to the true cdf, as the sample size n grows to infinity. This uniform convergence is exponential, as stated in the next fundamental result.

Theorem 2.1 (Dvoretzky-Kiefer-Wolfowitz-Massart inequality)**.** *Let $x_1, \ldots, x_n \sim P(\boldsymbol{x})$ be a real-valued sample. Then, for all $t > 0$,*

$$\mathbb{P}\left(\sup_{x \in \mathbb{R}} |P_n(x) - P(x)| > t\right) \leq 2e^{-2nt^2}.$$

Proof. See (Massart, 1990). □

Sometimes we want to determine how likely it is that a random variable takes a certain value. For continuous random variables with differentiable cdfs, *the probability density function* provides us with these likelihoods.

Definition 2.4 (Probability density function)**.** *The probability density function (pdf) of a real random variable $\boldsymbol{x}$ is*

$$p(\boldsymbol{x} = x) = p(x) = \frac{\mathrm{d}}{\mathrm{d}x} P(x)$$

Pdfs satisfy $p(x) \geq 0$ for all $x \in \mathcal{X}$, and $\int_{\mathcal{X}} p(x)\mathrm{d}x = 1$.

For discrete random variables, these likelihoods are given by the probability mass function.

Definition 2.5 (Probability mass function)**.** *The probability mass function (pmf) of a random variable $\boldsymbol{x}$ over a discrete space $\mathcal{X}$ is*

$$p(x) = \mathbb{P}(\boldsymbol{x} = x).$$

Pmfs are nonnegative for all $x \in \mathcal{X}$, and zero for all $x \notin \mathcal{X}$. Pmfs satisfy $\sum_{x \in \mathcal{X}} p(x) = 1$.

In many cases we are interested in summaries of random variables. One common way to summarize a random variable into k numbers is to use its first k moments. The n-th *moment* of a random variable $\boldsymbol{x}$ is

$$\mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}^n] = \mathbb{E}[\boldsymbol{x}^n] = \int_{-\infty}^{\infty} x^n \mathrm{d}P(\boldsymbol{x} = x),$$

and the nth *central moment* of a random variable $\boldsymbol{x}$ is

$$\mathbb{E}[(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])^n].$$

The first moment of a random variable $\mathbb{E}[\boldsymbol{x}]$ is the mean or expected value of $\boldsymbol{x}$, and characterizes how does the "average" sample from $P(\boldsymbol{x})$ looks like. The second central moment of a random variable $\mathbb{V}[\boldsymbol{x}] = \mathbb{E}\left[(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])^2\right]$ is the variance of $\boldsymbol{x}$, and measures the spread of samples drawn from $P(\boldsymbol{x})$ around its mean $\mathbb{E}[\boldsymbol{x}]$.

2.2.2 Multiple random variables

Now we turn to the joint study of collections of random variables. We can study a collection of real-valued random variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d$ as the vector-valued random variable $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$. Thus, $\boldsymbol{x}$ is a random variable taking values in $\mathbb{R}^d$. The cdf of $\boldsymbol{x}$ is

$$P(t) = \mathbb{P}(\boldsymbol{x} \leq t) = \mathbb{P}(\boldsymbol{x}_1 \leq t_1, \ldots, \boldsymbol{x}_d \leq t_d),$$

In the multivariate case, the empirical measure from Definition 2.2 takes the same form, and the ecdf is

$$P_n(t) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(x_{i,1} \leq t_1, \ldots, x_{i,d} \leq t_d),$$

where $x_1, \ldots, x_n \sim P(\boldsymbol{x})$.

The generalization of Theorem 2.1 to multivariate random variables is a groundbreaking result by Vapnik and Chervonenkis.

Theorem 2.2 (Vapnik-Chervonenkis)**.** *Let $\mathcal{X}$ be a collection of measurable sets in $\mathbb{R}^d$. Then, for all $nt^2 \geq 1$,*

$$\mathbb{P}\left(\sup_{x\in\mathcal{X}} |\mathbb{P}(x) - \mathbb{P}_n(x)| > t\right) \leq 4s(\mathcal{X}, 2n)e^{-nt^2/8},$$

where

$$s(\mathcal{X}, n) = \max_{x_1,\ldots,x_n\in\mathbb{R}^d} |\{\{x_1, \ldots, x_n\} \cap X : X \in \mathcal{X}\}| \leq 2^n$$

is known as the n-th shatter coefficient *of $\mathcal{X}$.*

Proof. See Vapnik and Chervonenkis (1971). □

The density function of a vector-valued random variable $\boldsymbol{x}$ is

$$p(x) = \left.\frac{\partial^d P(x)}{\partial x_1 \cdots \partial x_d}\right|_x.$$

Now we review two fundamental properties relating the density or mass functions of two random variables. First, the *total probability rule*

$$p(\boldsymbol{x}) = \sum_{y\in\mathcal{Y}} p(\boldsymbol{x}, \boldsymbol{y} = y),$$
$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{y} = y)\mathrm{d}y,$$

for discrete and continuous variables, respectively, is useful to compute the *marginal distribution* $p(\boldsymbol{x})$ of a single random variable $\boldsymbol{x}$ given the *joint distribution* $p(\boldsymbol{x}, \boldsymbol{y})$ of two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$. Second, *conditional probability rule*

$$p(\boldsymbol{x} \,|\, \boldsymbol{y} = y) = p(\boldsymbol{x} \,|\, y) = \frac{p(\boldsymbol{x}, y)}{p(y)},$$

is useful to compute the *conditional distribution* $p(\boldsymbol{x} \,|\, y)$ of the random variable $\boldsymbol{x}$ when another random variable $\boldsymbol{y}$ takes the value y, whenever $p(y) \neq 0$. Applying the conditional probability rule in both directions yields *Bayes' rule*

$$p(\boldsymbol{x} = x \,|\, \boldsymbol{y} = y) = \frac{p(\boldsymbol{y} = y \,|\, \boldsymbol{x} = x)p(\boldsymbol{x} = x)}{p(\boldsymbol{y} = y)}.$$

Remark 2.2 *(One slight abuse of notation).* Throughout this thesis, the cumulative distribution *function* $P(\boldsymbol{x})$ takes *values*

$$P(\boldsymbol{x} = x) = P(x),$$

and the probability density *function* $p(\boldsymbol{x})$ takes *values*

$$p(\boldsymbol{x} = x) = p(x).$$

All these notations will be used interchangeably whenever this causes no confusion. The notation of conditional distributions is more subtle. While the element $p(\boldsymbol{x} \,|\, y)$ is a function, the element $p(\boldsymbol{x} = x \,|\, \boldsymbol{y} = y) = p(x \,|\, y)$ is a number. ◊

Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two continuous random variables taking values in $\mathcal{X}$ and $\mathcal{Y}$ respectively, and let $g : \mathcal{X} \to \mathcal{Y}$ be a bijection. Then:

$$p(\boldsymbol{y} = y) = \left| \frac{\mathrm{d}}{\mathrm{d}y} g^{-1}(y) \right| p(\boldsymbol{x} = g^{-1}(y)). \tag{2.2}$$

For two random variables taking values $x \in \mathcal{X}$, the distance between their respective density functions p and q is often measured using the Kullback-Liebler (KL) divergence

$$\mathrm{KL}(p \,\|\, q) = \int_{\mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \mathrm{d}x. \tag{2.3}$$

As it happened with single random variables, we can create summaries of multiple random variables and their relationships. One of these summaries are the (r_x, r_y)-mixed central moments:

$$\mathbb{E}[(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])^{r_x} (\boldsymbol{y} - \mathbb{E}[\boldsymbol{y}])^{r_y}],$$

For example, the $(2, 2)$-mixed central moment of two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ is their covariance

$$\mathrm{cov}(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}\left[(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])^2 (\boldsymbol{y} - \mathbb{E}[\boldsymbol{y}])^2\right].$$

When normalized, the covariance statistic becomes the correlation statistic

$$\rho(\boldsymbol{x}, \boldsymbol{y}) = \frac{\mathrm{cov}(\boldsymbol{x}, \boldsymbol{y})}{\sqrt{\mathbb{V}[\boldsymbol{x}]\, \mathbb{V}[\boldsymbol{y}]}} \in [-1, 1].$$

The correlation statistic describes to what extent the joint distribution of $\boldsymbol{x}$ and $\boldsymbol{y}$ can be described with a straight line. In other words, correlation measures to what extent two random variables are *linearly* dependent. Similar equations follow to derive the mixed central moments of a collection of more than two random variables.

2.2.3 Statistical estimation

The crux of statistics is to identify interesting aspects θ about random variables $\boldsymbol{x}$, and to approximate them as estimates $\theta_n(x) = \theta_n$ using n samples $x = x_1, \ldots, x_n \sim P(\boldsymbol{x})^n$. One can design multiple estimates θ_n for the same quantity θ; therefore, it is interesting to quantify and compare the quality of different estimates, in order to favour one of them for a particular application. Two of the most important quantities about statistical estimators are their *bias* and *variance*.

On the one hand, the *bias* measures the deviation between the quantity of interest θ and the expected value of our estimator θ_n:

$$\text{Bias}(\theta_n, \theta) = \mathbb{E}_{x \sim P^n}[\theta_n] - \theta.$$

Estimators with zero bias are *unbiased estimators*. Unbiasedness is unrelated to consistency, where consistency means that the estimator converges in probability to the true value being estimated, as the sample size increases to infinity. Thus, unbiased estimators can be inconsistent, and consistent estimators can be biased.

On the other hand, the *variance* of an estimator

$$\text{Variance}(\theta_n) = \mathbb{E}\left[(\theta_n - \mathbb{E}_{x \sim P^n}[\theta_n])^2\right]$$

measures its dispersion around the mean. The sum of the variance and the square of the bias is equal to the mean squared error of the estimator

$$\text{MSE}(\theta_n) = \text{Variance}(\theta_n) + \text{Bias}^2(\theta_n, \theta).$$

This reveals a key trade-off: different estimators achieving the same mean square error can have a different bias-variance decompositions. As a matter of fact, bias and variance are in many cases competing quantities. We discuss this fundamental issue in Section 3.5.

2.2.4 Concentration inequalities

We now review useful results concerning the concentration of averages of independent random variables. For a more extensive treatment, consult (Boucheron et al., 2013).

Theorem 2.3 (Jensen's inequality)**.** *Let $\boldsymbol{x}$ be a random variable taking values in $\mathcal{X}$, and let $f : \mathcal{X} \to \mathbb{R}$ be a convex function. Then, for all $\boldsymbol{x}$ and f,*

$$f(\mathbb{E}[\boldsymbol{x}]) \leq \mathbb{E}[f(\boldsymbol{x})]\,.$$

Theorem 2.4 (Markov's inequality)**.** *Let $\boldsymbol{x}$ be a random variable taking values in the nonnegative reals. Then,*

$$\mathbb{P}(\boldsymbol{x} \geq a) \leq \frac{\mathbb{E}[\boldsymbol{x}]}{a}.$$

Markov's inequality is tightly related to Chebyshev's inequality.

Theorem 2.5 (Chebyshev's inequality)**.** *Let $\boldsymbol{x}$ be a random variable with finite expected value μ and finite variance $\sigma^2 \neq 0$. Then, for all $k > 0$,*

$$\mathbb{P}(|\boldsymbol{x} - \mu| \geq k\sigma) \leq \frac{1}{k^2}.$$

As opposed to the polynomial concentration of Markov's and Chebyshev's inequalities, the more sophisticated *Chernoff bounds* offer exponential concentration. The simplest Chernoff bound is Hoeffding's inequality.

Theorem 2.6 (Hoeffding's inequality)**.** *Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ be a collection n independent random variables, where $\boldsymbol{x}_i$ takes values in $[a_i, b_i]$, for all $1 \leq i \leq n$. Let $\bar{\boldsymbol{x}} = \sum_{i=1}^n \boldsymbol{x}_i$. Then, for all $t > 0$,*

$$\mathbb{P}(\bar{\boldsymbol{x}} - \mathbb{E}[\bar{\boldsymbol{x}}] \geq t) \leq \exp\left(-\frac{2nt^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

One can sharpen Hoeffding's inequality by taking into account the variance of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, giving rise to Bernstein's inequality.

Theorem 2.7 (Bernstein's inequality)**.** *Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ be a collection of n independent random variables with zero-mean, where $|\boldsymbol{x}_i| \leq M$ for all $1 \leq i \leq n$ almost surely. Let $\bar{\boldsymbol{x}} = \sum_{i=1}^n \boldsymbol{x}_i$. Then, for all $t > 0$*

$$\mathbb{P}(\bar{\boldsymbol{x}} \geq t) \leq \exp\left(-\frac{1}{2}\frac{t^2}{\sum_{i=1}^n \mathbb{E}\left[\boldsymbol{x}_i^2\right] + \frac{1}{3}Mt}\right).$$

Furthermore, random variables $\boldsymbol{z}$ with Bernstein bounds of the form

$$\mathbb{P}(\boldsymbol{z} \geq t) \leq C \exp\left(-\frac{1}{2}\frac{t^2}{A + Bt}\right)$$

admit the upper bound

$$\mathbb{E}[\boldsymbol{z}] \leq 2\sqrt{A}(\sqrt{\pi} + \sqrt{\log C}) + 4B(1 + \log C).$$

We can also achieve concentration not only over random averages, but over more general *functions* f of random variables, assuming that the function f is well behaved. One example of such results is McDiarmid's inequality.

Theorem 2.8 (McDiarmid's inequality)**.** *Let $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ be a collection of n independent random variables taking real values, and let $f : \mathbb{R}^n \to \mathbb{R}$ be a function satisfying*

$$\sup_{x_1, \ldots, x_n, x_i'} \left|f(x_1, \ldots, x_n) - f(x_1, \ldots, x_i', \ldots, x_n)\right| \leq c_i$$

for all $1 \leq i \leq n$. Then, for all $t > 0$,

$$\mathbb{P}(f(\boldsymbol{x}) - \mathbb{E}[f(\boldsymbol{x})] \geq t) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right).$$

A key tool in the analysis of the presented algorithms in this thesis is the Matrix-Bernstein inequality (Tropp, 2015), which mirrors Theorem 2.7 for matrix-valued random variables.

Theorem 2.9 (Matrix Bernstein's inequality)**.** *Let $\boldsymbol{X}_1, \dots, \boldsymbol{X}_n$ be a collection of n independent random variables taking values in $\mathbb{R}^{d_1 \times d_2}$, where $\mathbb{E}[\boldsymbol{X}_i] = 0$ and $\|\boldsymbol{X}_i\|_2 \leq M$. Let $\bar{\boldsymbol{X}} = \sum_{i=1}^n \boldsymbol{X}_i$, and define*

$$\sigma^2 = \max\left(\left\|\mathbb{E}\left[\bar{\boldsymbol{X}}^\top \bar{\boldsymbol{X}}\right]\right\|, \left\|\mathbb{E}\left[\bar{\boldsymbol{X}}\bar{\boldsymbol{X}}^\top\right]\right\|\right).$$

Then, for all $t > 0$,

$$\mathbb{P}\left(\|\bar{\boldsymbol{X}}\|_2 \geq t\right) \leq (d_1 + d_2) \exp\left(-\frac{\frac{1}{2}t^2}{\sigma^2 + \frac{1}{3}Mt}\right).$$

Furthermore,

$$\mathbb{E}\left[\|\bar{\boldsymbol{X}}\|_2\right] \leq \sqrt{2\sigma^2 \log(d_1 + d_2)} + \frac{1}{3}M \log(d_1 + d_2).$$

One last fundamental result that we would like to mention is the Union bound.

Theorem 2.10 (Union bound)**.** *Let $E_1, \dots, E_n$ be a collection of events. Then,*

$$\mathbb{P}(\cup_{i=1}^n E_i) \leq \sum_{i=1}^n \mathbb{P}(E_i).$$

2.3 Machine learning

Imagine that I give you the sequence

$$1, 2, 3, \dots$$

and I ask: *What number comes next?*

Perhaps the more natural answer is *four*, assuming that the given sequence is the one of the positive integers. A more imaginative answer could be *two*, since that agrees with the sequence of the greatest primes dividing n. Or maybe *five*, which agrees with the sequence of numbers not divisible by a square greater than one. A more twisted mind would prefer the answer *two hundred and eleven*, since that is the next "home" prime. In any case, the more digits that we observe from the sequence and the less paranoid we are, the larger the amount of hypothesis we will be able to reject and the closer we will get to inferring the correct sequence. Machine learning uses the tools of probability theory and statistics to formalize *inference* problems like these.

This section reviews the fundamentals of learning theory, regression, classification, and model selection. For a more extensive treatment on machine learning topics, we refer the reader to the monographs (Mohri et al., 2012; Murphy, 2012; Shalev-Shwartz and Ben-David, 2014).

2.3.1 Learning theory

Consider two random variables: one *input* random variable $\boldsymbol{x}$ taking values in $\mathcal{X}$, and one *output* random variable $\boldsymbol{y}$ taking values in $\mathcal{Y}$. The usual problem in learning theory is to find a *function*, *dependence*, or *pattern* that "best" predicts values for the output variable given the values taken by the input variable. We have three resources to our disposal to solve this problem. First, a *sample* or *data*

$$\mathcal{D} = \{(x_1, y_1) \ldots, (x_n, y_n)\} \sim P^n(\boldsymbol{x}, \boldsymbol{y}),\ x_i \in \mathcal{X},\ y_i \in \mathcal{Y}. \tag{2.4}$$

Second, a *function class* $\mathcal{F}$, which is a set containing functions $f : \mathcal{X} \to \mathcal{Y}$. And third, a *loss function* $\ell : \mathcal{Y} \to \mathcal{Y}$, which penalizes departures between predictions $f(x)$ and true output values y. Using these three ingredients, one way to solve the learning problem is to find the function $f \in \mathcal{F}$ minimizing the *expected risk*

$$R(f) = \int_{\mathcal{X}\times\mathcal{Y}} \ell(f(x), y) \mathrm{d}P(\boldsymbol{x}, \boldsymbol{y}). \tag{2.5}$$

Unfortunately, we can not compute the expected risk (2.5), since we do not have access to the *data generating distribution* P. Instead, we are given a finite sample $\mathcal{D}$ drawn from P^n. Therefore, we may use instead the available data to minimize the *empirical risk*

$$R_n(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i),$$

which converges to the expected risk as the sample size grows, due to the law of large numbers.

Remark 2.3 *(Some notations in learning).* We call the set (2.4) *data* or *sample*, where each contained (x_i, y_i) is one *example*. Examples contain *inputs* x_i and *outputs or targets* y_i. When the targets are categorical, we will call them *labels*. When the inputs are vectors in $\mathbb{R}^d$, then $x_{i,j} \in \mathbb{R}$ is the jth feature of the ith example. Sometimes we will arrange the data (2.4) in two matrices: the *feature matrix* $X \in \mathbb{R}^{n\times d}$, where $X_{i,:} = x_i$, and the *target matrix* $Y \in \mathbb{R}^{n\times q}$, where $Y_{i,:} = y_i$. Finally, we sometimes refer to the data (2.4) as the *raw representation* or the *original representation*. ◊

Using the definitions of expected and empirical risk, construct the two functions

$$f^\star = \arg\min_{f\in\mathcal{F}} R(f),$$
$$f_n = \arg\min_{f\in\mathcal{F}} R_n(f),$$

called the *expected risk minimizer*, and the *empirical risk minimizer*. We say that a learning algorithm is *consistent* if, as the amount of available data

grows ($n \to \infty$), the output of the algorithm converges to the expected risk minimizer. The speed at which this convergence happens with respect to n is the *learning rate*. Also, consider the function

$$g^\star = \arg\min_{g:\mathcal{X}\to\mathcal{Y}} R(g).$$

The function $g^\star$ is the function from *the set of all measurable functions* attaining minimal expected risk in our learning problem. We call $g^\star$ the *Bayes predictor*, and $R(g^\star)$ the *Bayes error*. Note that perhaps $g^\star \notin \mathcal{F}$! In this setup, the goal of learning theory is

"How well does f_n describe the dependence between $\boldsymbol{x}$ and $\boldsymbol{y}$, when compared to $g^\star$?"

Mathematically, the answer to this question splits in two parts:

$$R(f_n) - R(g) = \underbrace{R(f_n) - R(f^\star)}_{\text{estimation error}} + \underbrace{R(f^\star) - R(g^\star)}_{\text{approximation error}}.$$

The *estimation error* arises because we approximate the expected risk minimizer with the empirical risk minimizer. The *approximation error* arises because we approximate the best possible function $g^\star$ with the best function from our function class $\mathcal{F}$. Let's take a look at the analysis of the estimation error.

$$\begin{aligned} R(f_n) &= R(f_n) - R(f^\star) + R(f^\star) \\ &\leq R(f_n) - R(f^\star) + R(f^\star) + R_n(f^\star) - R_n(f_n) \qquad (2.6) \\ &\leq 2 \sup_{f\in\mathcal{F}} |R(f) - R_n(f)| + R(f^\star). \qquad (2.7) \end{aligned}$$

The inequality (2.6) follows because we know that the empirical risk minimizer f_n satisfies

$$R_n(f^\star) - R_n(f_n) \geq 0. \qquad (2.8)$$

Importantly, f_n is the only function for which we can assure (2.8). Thus, the guarantees of empirical risk minimization only hold for function classes allowing the efficient computation of their empirical risk minimizers. In practical terms, this often means that finding f_n is a convex optimization problem. The inequality (2.7) follows by assuming twice the worst difference between the empirical and expected risk of one function. Summarizing, the estimation error allows the upper bound

$$R(f_n) - R(f^\star) \leq 2 \sup_{f\in\mathcal{F}} |R(f) - R_n(f)|. \qquad (2.9)$$

The right-hand side of this inequality is the suprema of the empirical process $\{|R(f) - R_n(f)|\}_{f\in\mathcal{F}}$. To upper bound this suprema in a meaningful way, we first measure the complexity of the function class $\mathcal{F}$. Defined next, Rademacher complexities are one choice to do this (Koltchinskii, 2001).

Definition 2.6 (Rademacher complexity)**.** *Let $\mathcal{F}$ be a class of functions from $\mathcal{X}$ to $[a,b]$, $\mathcal{D} = (x_1, \ldots, x_n)$ a vector in $\mathcal{X}^n$, and $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_n)$ be a vector of uniform random variables taking values in $\{-1, +1\}$. Then, the* empirical Rademacher complexity *of $\mathcal{F}$ is*

$$\mathrm{Rad}_{\mathcal{D}}(\mathcal{F}) = \mathbb{E}\left[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\sigma}_i f(x_i)\right]$$

Given $\mathcal{D} \sim P^n$, the Rademacher *complexity of $\mathcal{F}$ is*

$$\mathrm{Rad}_n(\mathcal{F}) = \mathbb{E}_{\mathcal{D}\sim P^n}[\mathrm{Rad}_{\mathcal{D}}(\mathcal{F})]\,.$$

The Rademacher complexity of a function class $\mathcal{F}$ measures the ability of functions $f \in \mathcal{F}$ to hallucinate patterns from random noise. Like the flexible mind of children imagining dragons in clouds, only flexible functions are able to imagine regularities in randomness. Thus, Rademacher complexities measure how flexible or rich the functions $f \in \mathcal{F}$ are. Rademacher complexities have a typical order of $O(n^{-1/2})$ (Koltchinskii, 2011). Although in this thesis we use Rademacher complexities, there exist other measures of capacity, such as the VC-Dimension, VC-Entropy, fat-shattering dimension, and covering numbers. The relationships between these are explicit, due to results by Hussler and Dudley (Boucheron et al., 2005). To link Rademacher complexities to the suprema (2.9), we need one last technical ingredient: the symmetrization inequality.

Theorem 2.11 (Symmetrization inequality)**.** *Let $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. Then, for any function class $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$,*

$$\mathbb{E}_{\mathcal{D}\sim P^n}\left[\sup_{f\in\mathcal{F}} |R(f) - R_n(f)|\right] \leq 2\mathrm{Rad}_n(\ell \circ \mathcal{F}).$$

Proof. See (Boucheron et al., 2005, page 5) □

Using Theorems 2.8 and 2.11, we can upper bound the suprema (2.9), which in turn upper bounds the excess risk between the empirical and expected risk minimizers in $\mathcal{F}$. The resulting upper bound depends on the error attained by the empirical risk minimizer, the Rademacher complexity of $\mathcal{F}$, and the size of training data.

Theorem 2.12 (Excess risk of empirical risk minimization)**.** *Let $\mathcal{F}$ be a set of functions $f : \mathcal{X} \to [0,1]$. Then, for all $\delta > 0$ and $f \in \mathcal{F}$,*

$$R(f) \leq R_n(f) + 2\mathrm{Rad}_n(\mathcal{F}) + \sqrt{\frac{\log\frac{1}{\delta}}{2n}},$$

$$R(f) \leq R_n(f) + 2\mathrm{Rad}_{\mathcal{D}}(\mathcal{F}) + \sqrt{\frac{\log\frac{2}{\delta}}{2n}},$$

with probability at least $1 - \delta$.

Proof. See, for example, (Boucheron et al., 2005, Theorem 3.2). □

Theorem 2.12 unveils two important facts. First, one sufficient condition for the consistency of empirical risk minimization is that the Rademacher complexity of $\mathcal{F}$ tends to zero as the amount of training data n tends to infinity. Second, the $O(n^{-1/2})$ speed of convergence, at least without further assumptions, is optimal (Shalev-Shwartz and Ben-David, 2014, Theorem 6.8).

Remark 2.4 *(Learning faster).* In some situations, it is possible to obtain a faster learning rate than the $O(n^{-1/2})$ rate from Theorem 2.12.

In binary classification, we can obtain a $O(n^{-1})$ learning rate for empirical risk minimization if i) our function class $\mathcal{F}$ has finite VC-Dimension, ii) the Bayes predictor $g^\star$ is in $\mathcal{F}$, and iii) the problem is not "too noisy". Massart (2000) formalizes the third condition as

$$\inf_{x \in \mathcal{X}} |2\mathbb{P}(\boldsymbol{y} = 1 \,|\, \boldsymbol{x} = x) - 1| > 0.$$

The fast rate (Bartlett et al., 2005, Corollary 5.3) stems from Talagrand's inequality, which refines the result from McDiarmid's inequality by taking into account second order statistics.

In regression, we can obtain a $O(n^{-1})$ learning rate if i) the loss is Lipschitz-continuous and bounded, ii) our function class $\mathcal{F}$ is a convex set containing uniformly bounded functions, and iii) the *local* Rademacher complexity of $\mathcal{F}$ is $o(n^{-1/2})$ (Bartlett et al., 2005, Corollary 5.3). ◊

Throughout the rest of this thesis, we will consider function classes of the form

$$\mathcal{F}_\phi = \left\{ f : f(x) = \langle A, \phi(x) \rangle_{\mathcal{H}_\phi} \right\},$$

where $\phi : \mathcal{X} \to \mathcal{H}_\phi$ is a feature map transforming the raw data x into the representation $\phi(x)$, and $A : \mathcal{X} \to \mathcal{Y}$ is a linear operator summarizing the representation $\phi(x)$ into the target function or pattern. The next chapter studies different techniques to construct the feature map ϕ, responsible for computing the data representations $\phi(x)$.

Remark 2.5 *(Subtleties of empirical risk minimization).* Throughout this thesis we will consider *identically and independently distributed* (iid) data. Mathematically, we write this as $x_1, \ldots, x_n \sim P^n$, where P^n is the n-product measure built from P. This will be the main assumption permitting learning: the relationship between examples in the past (the training data) and examples in the future (the test data) is that all of them are described by the same distribution P. This is the "machine learning way" to resolve Hume's *the problem of induction*: without assumptions, learning and generalization are impossible.

For the empirical risk minimization learning theory to work, one must choose the triplet formed by the data, the function class, and the loss function

independently. This means that theory only holds when we train *once*, and we do not adapt our algorithms and parameters to the training outcome. ◊

Remark 2.6 *(Universal consistency)*. *Universally consistent* learning algorithms provide with a sequence of predictors that converge to the Bayes predictor as the training data grows to infinity, for all data generating distributions. But, when considering all data generating distributions, universally consistent algorithms do not guarantee any learning rate. Formally, for any learning algorithm and $\varepsilon > 0$, there exists a distribution P such that

$$\mathbb{P}\left(R(f_n) \geq \frac{1}{2} - \varepsilon\right) = 1,$$

where f_n is the output of the learning algorithm when given data $D \sim P^n$ (Bousquet et al., 2004, Theorem 9). Simply put, we can always construct data generating distributions under which a given algorithm will require an exponential amount of data, or said differently, will learn exponentially slow. Since these distributions exist for all learning algorithms, we can conclude that *there is no free lunch* (Wolpert and Macready, 1997), and that all learning algorithms are equally "bad". But there is hope for good learning algorithms, since natural data is not arbitrary, but has rich structure. ◊

2.3.2 Model selection

Given different models —for instance, different function classes— to solve one learning task, which one should we prefer? This is the question of *model selection*.

Model selection is problematic when learning from finite noisy data. In such situations, the complexity of our learning algorithm will determine how well we can tell apart patterns from noise. If using a too flexible learning algorithm, we may hallucinate patterns in the random noise polluting our data. Such hallucinations will not be present in the test data, so our model will generalize poorly, and have high expected risk. We call this situation *overfitting*. On the other hand, if using a too simple learning algorithm, we will fail to capture all of the pattern of interest, both at training and test data, having high empirical and expected risk. We call this situation *underfitting*.

Figure 2.1 illustrates model selection. Here, we want to learn the pattern

$$f(x) = \cos(3x),$$

hinted by the noisy data depicted as gray dots. We offer three different solutions to the problem: f_a, f_b, and f_c. First, see the "complex" model f_c, depicted in red in the right-hand side of Figure 2.1. We say that f_c *overfits* the data, because it incorporates the random noise polluting the data into the learned pattern. Since future test data will have different random

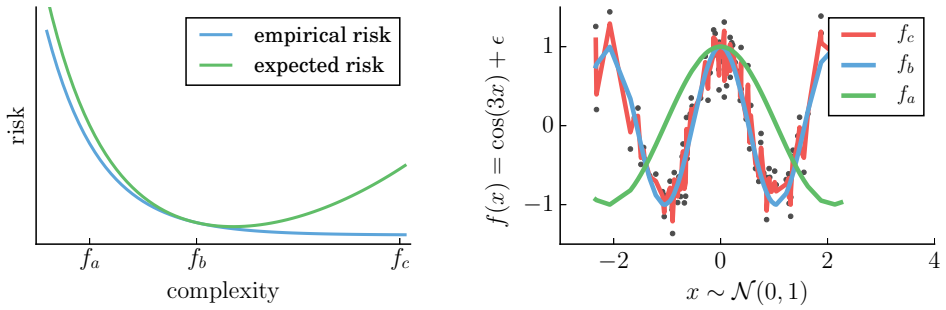


Figure 2.1: Model selection.

noise, f_c will wiggle at random and generalize poorly. This is seen in the left-hand side of Figure 2.1, where the expected risk of f_c is higher than its empirical risk. Second, the "simplistic" model f_b. We say that f_b *underfits* the data, because it is not flexible enough to describe the high frequency of the sinusoidal pattern of interest. In the left-hand side of Figure 2.1, this translates in both the empirical and expected risks of f_a being high. However, the model f_b achieves a good balance between complexity and simplicity, as it accommodates the pattern but ignores the noise in the data. This balance translates into minimal expected risk, as illustrated in the left-hand side of the figure. The model f_b allows an increase in empirical risk to ignore the noise, lower its expected risk, and improve generalization. The techniques sacrificing empirical risk in exchange to improved expected risk are known as *regularization*. As we will see in the next chapter, the differences between these three predictors relate to the bias-variance trade off, which will be discussed in Section 3.5.

The question of model selection often follows *Occam's razor*: prefer the "simplest" model (in terms of complexity) that explains the data "well" (in terms of empirical risk). Different model selection strategies give different meanings to the phrases "being simple" and "explaining the data well". Next, we review three of the most important model selection techniques.

Structural Risk Minimization

One alternative to model selection is the use of the theoretical results reviewed in this section. Observe that Theorem 2.12 upper bounds the expected risk of a predictor f as the sum of three terms: the training error $R_n(f)$ of the model, the complexity $\text{Rad}_n(\mathcal{F})$ of the model class $\mathcal{F}$, and the amount of available training data n. For a fixed amount of training data n, we can perform model selection by considering increasingly complex models, and selecting the one minimizing the sum $R_n(f) + 2\text{Rad}_n(\mathcal{F})$ from Theorem 2.12. This is *Structural Risk Minimization* (Vapnik, 1998). Unfortunately, the

upper bounds provided by results such as Theorem 2.12 are often too loose to use in practice, and function class complexity measures are too difficult or impossible to compute.

Bayesian model selection

One central quantity in Bayesian statistics is the *evidence* or *marginal likelihood*:

$$p(\boldsymbol{x} = x \,|\, \boldsymbol{m} = m) = \int p(\boldsymbol{x} = x \,|\, \boldsymbol{\theta} = \theta, \boldsymbol{m} = m) p(\boldsymbol{\theta} = \theta \,|\, \boldsymbol{m} = m) \mathrm{d}\theta. \tag{2.10}$$

In words, this integral expresses the probability of the data x coming from the model m as the integral over the *likelihood* of all possible model parameters θ, weighted by their *prior*. When deciding between two models m_1 and m_2, a Bayesian statistician will use the marginal likelihood to construct the ratio of posteriors or *Bayes factor*

$$\frac{p(\boldsymbol{m} = m_1 \,|\, \boldsymbol{x} = x)}{p(\boldsymbol{m} = m_2 \,|\, \boldsymbol{x} = x)} = \frac{p(\boldsymbol{m} = m_1) \cdot p(\boldsymbol{x} = x \,|\, \boldsymbol{m} = m_1)}{p(\boldsymbol{m} = m_2) \cdot p(\boldsymbol{x} = x \,|\, \boldsymbol{m} = m_2)},$$

where $p(\boldsymbol{m} = m_1)$ is his prior belief about the correct model being m_1, and similarly for m_2. If the Bayes factor is greater than 1, the Bayesian statistician will prefer the model m_1; otherwise, she will prefer the model m_2.

What is special about this procedure? Assume for simplicity that $p(\boldsymbol{m}) = \frac{1}{2}$ for both models m_1 and m_2. Since the marginal likelihood $p(\boldsymbol{x} \,|\, \boldsymbol{m})$ is a probability distribution, it has to normalize to one when integrated over all possible datasets $x \sim P^n$. Thus, flexible models need to assign small likelihoods to the large amount of datasets that they can describe, but simple models can assign large likelihoods to the small amount of datasets that they can describe. This trade-off serves as a model selection criteria: simpler models able to explain the data well give higher marginal likelihood.

In some situations, we need to select a model from an infinite amount of candidates, all of them parametrized as a continuous random variable $\boldsymbol{m}$. In these situations, Bayesian model selection is solving the optimization problem

$$m^\star = \arg\max_m p(\boldsymbol{m} = m) \cdot p(\boldsymbol{x} = x \,|\, \boldsymbol{m} = m). \tag{2.11}$$

Bayesian model selection faces some difficulties. First, the computation of the marginal likelihood, which is solving the integral (2.10), is often intractable. Second, even if the computation of the marginal likelihood is feasible, the Bayesian model selection optimization problem (2.11) is often nonconvex; thus, we are not protected from selecting an arbitrarily suboptimal model. Third, Bayesian methods are inherently subjective. In the context of model selection, this means that different prior beliefs about

models and their parameters can lead to two different Bayesian statisticians choosing different models, even if the data at hand is the same. Optimizing the marginal likelihood is yet another optimization problem, and there is no free lunch about it: if our models are over-parametrized we still risk overfitting. However, nonparametric Bayesian models often have a small amount of parameters, making Bayesian model selection a very attractive solution.

Cross-validation

In order to select the best model from a set of candidates, cross-validation splits the available data $\mathcal{D}$ in two random disjoint subsets: the training set $\mathcal{D}_{\text{tr}}$ and the validation set $\mathcal{D}_{\text{va}}$. Then, cross-validation trains each of the candidate models using the training data $\mathcal{D}_{\text{tr}}$, and chooses the model with the smallest risk on the unseen validation set $\mathcal{D}_{\text{va}}$. When the space of models is parametrized as a continuous random variable, cross-validation monitors the model error in the validation set, and stops the optimization when such error starts increasing. This is known as *early stopping*.

There are extensions of cross-validation which aim to provide a more robust estimate of the quality of each model in the candidate set. On the one hand, *leave-p-out cross-validation* uses p samples from the data as the validation set, and the remaining samples as the training set. Leave-p-out cross-validation selects the model with the smallest average error over all such splits. On the other hand, *k-fold cross-validation* divides the data into k disjoint subsets of equal size and performs cross-validation k times, each of them using as validation set one of the k subsets, and as training set the remaining $k-1$ subsets. Again, k-fold cross-validation selects the model achieving the smallest average error over all such splits. But beware! No theoretical guarantees are known for the correctness of the leave-p-out ($p > 1$) and k-fold cross-validation schemes, since they involve the repeated use of the same data.

On the positive side, cross-validation is easy to apply and only requires iid data. On the negative side, applying cross-validation involves intensive computation and throwing away training data, to be used as a validation set.

2.3.3 Regression as least squares

Assume data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ coming from the model

$$\begin{aligned} x_i &\sim P(\boldsymbol{x}), \\ \varepsilon_i &\sim \mathcal{N}(\boldsymbol{\varepsilon}; 0, \lambda^2), \\ y_i &\leftarrow f(x) + \varepsilon_i, \end{aligned}$$

where $x_i \in \mathbb{R}^d$ for all $1 \leq i \leq n$, and

$$\alpha \sim \mathcal{N}(0, \Sigma_\alpha),$$
$$f(x) \leftarrow \langle \alpha,\, x \rangle.$$

Therefore, we here assume a Gaussian prior over the parameter vector α, and additive Gaussian noise over the measurements y_i. To simplify notation, we do not include a bias term in f, but assume that $x_{i,d} = 1$ for all $1 \leq i \leq n$. Using Bayes' rule and averaging over all possible linear models, the distribution over the function value $f = f(x)$ is

$$\begin{aligned} p(\boldsymbol{f} \,|\, x, X, y) &= \int p(f \,|\, x, w) p(w \,|\, X, y) \mathrm{d}w \\ &= \mathcal{N}\left(\boldsymbol{f}; \lambda^{-2} x A^{-1} X^\top y, x A^{-1} x^\top\right) \end{aligned} \tag{2.12}$$

where $X = (x_1, \ldots, x_n)^\top \in \mathbb{R}^{n \times d}$, $y = (y_1, \ldots, y_n)^\top \in \mathbb{R}^{n \times 1}$, and $A = \lambda^{-2} X^\top X + \Sigma_\alpha^{-1}$ (Rasmussen and Williams, 2006). The mean of (2.12) is

$$\hat{\alpha} = (X^\top X + \lambda^2 \Sigma_\alpha)^{-1} X^\top y$$

and equals the *maximum a posteriori* solution of the Bayesian least squares problem. When $\Sigma_\alpha = I_d$, it coincides with the global minima of the least-squares empirical risk

$$R(\alpha, \lambda, \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} \left(\langle \alpha,\, x_i \rangle - y_i\right)^2 + \frac{\lambda^2}{2} \left\|\alpha\right\|_2^2. \tag{2.13}$$

The term λ^2 in (2.13) is a *regularizer*: larger values of λ will favour simpler solutions, which prevent absorbing the noise ε_i into the inferred pattern $\hat{\alpha}$. In least-squares, we can search for the best regularization value at essentially no additional computation (Rifkin and Lippert, 2007).

2.3.4 Classification as logistic regression

Logistic regressors $f : \mathbb{R}^d \to \mathbb{R}^q$ have form

$$f(x; W)_k = s(\langle W,\, x \rangle)_k,$$

for all $1 \leq k \leq q$, where $x \in \mathbb{R}^d$ and $W \in \mathbb{R}^{d \times q}$, and the softmax operation

$$s(z)_k = \frac{\exp(z_k)}{\sum_{j=1}^{q} \exp(z_j)}$$

outputs probability vectors $s(z)$, meaning that $s(z)_k \geq 0$ for all $1 \leq k \leq q$, and $\sum_{k=1}^{q} s(z)_k = 1$. Using a dataset $\{(x_i, y_i)\}_{i=1}^n$, we can learn a logistic regressor by maximizing the Multinomial likelihood

$$L(X, Y, W) = \prod_{i=1}^{n} \prod_{k=1}^{q} f(X_{i,:}; W)_k^{Y_{i,k}},$$

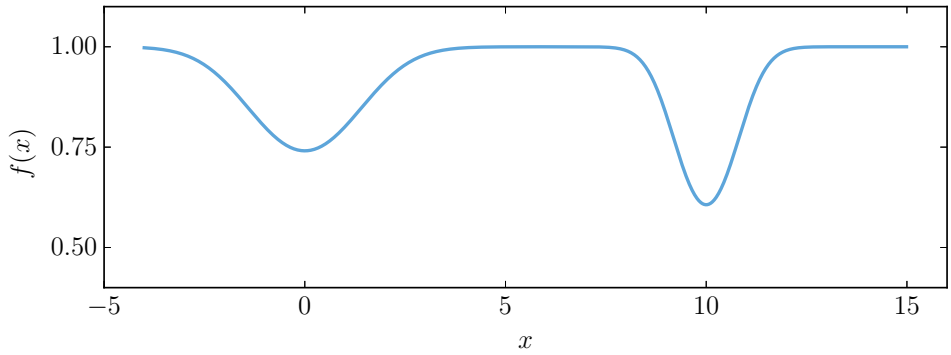


Figure 2.2: A one-dimensional nonconvex function, with a local minima at $x = 0$, a global minima at $x = 10$, and a saddle point at $x = 6$.

or equivalently, the log-likelihood

$$\log L(X, Y, W) = \sum_{i=1}^{n} \sum_{k=1}^{q} Y_{i,k} \log f(X_{i,:}; W)_k,$$

where $X = (x_1, \dots, x_n)^\top \in \mathbb{R}^{n \times d}$ and $Y = (y_1, \dots, y_n)^\top \in \mathbb{R}^{n \times q}$. Here, the target vectors y_i follow a *one-hot-encoding*: if the ith example belongs to the kth class, $y_{i,k} = 1$ and $y_{i,k'} = 0$ for all $k' \neq k$. Maximizing the log-likelihood is minimizing the risk

$$E(W) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(X_{i,:}; W), Y_{i,:}) \tag{2.14}$$

where ℓ is the *cross-entropy loss*

$$\ell(\hat{y}, y) = -\sum_{k=1}^{q} y_k \log \hat{y}_k.$$

Therefore, classification as logistic regression *is* a multivariate (or *multitask*) linear regression $\langle W, x \rangle$ under a different loss function: the composition of the softmax and the cross-entropy operations.

Minimizing (2.14) with respect to the parameters W is a convex optimization problem. The next section reviews how to solve these and other optimization problems, ubiquitous in this thesis.

2.4 Numerical optimization

Numerical optimization algorithms deal with the problem of computing the minimum value of functions, and where such minimum is. When we do not

impose any assumptions over the functions that we minimize, optimization is an NP-hard problem. In particular, numerical optimization is challenging because general functions have *local minima* and saddle points, that can be far away from their global minima. Figure 2.2 illustrates these challenges for a one-dimensional function. Think of rolling a marble down the graph of the function, starting at a random location, with the goal of landing the marble at the global minima $x = 10$. Then, we risk at getting the marble stuck at the local minima $x = 0$, or at the saddle point or *plateau* around $x = 6$. In higher dimensions, problems do only get worse.

The rest of this section reviews basic concepts about numerical optimization, such as function derivatives and gradients, convex functions, and gradient based methods for numerical optimization. Numerical optimization underlies much of this thesis and the whole field of machine learning. For a extensive treatise on numerical optimization, we recommend the monographs (Boyd and Vandenberghe, 2004; Nesterov, 2004; Bubeck, 2015).

2.4.1 Derivatives and gradients

First, we recall some basic definitions about multidimensional functions and their derivatives. Let $f : \mathbb{R}^d \to \mathbb{R}$ be a differentiable function, with partial derivatives

$$\frac{\partial f}{\partial x_i},$$

for all $1 \leq i \leq d$. Then, the gradient of f is the vector of all d partial derivatives

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_d} \right).$$

If we take all second derivatives and arrange them in an $d \times d$ matrix, we get the Hessian of f, with entries

$$H(f(x))_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}.$$

If the function f maps $\mathbb{R}^d$ into $\mathbb{R}^q$, then we can arrange all first derivatives into a matrix called the Jacobian of f, with entries

$$J(f(x))_{i,j} = \frac{\partial f_i}{\partial x_j}.$$

It is easy to verify that, if $f : \mathbb{R}^d \to \mathbb{R}$, then $J(\nabla f(x)) = H(f(x))$.

Now, two definitions to characterize the good behaviour of a function. First, we say that the function $f : \mathbb{R}^d \to \mathbb{R}^q$ is L-Lipschitz if it satisfies

$$\|f(x_1) - f(x_2)\| \leq L\|x_1 - x_2\|,$$

for all $x_1, x_2 \in \mathbb{R}^d$. L-Lipschitz functions have bounded gradients, $\|\nabla f(x)\| \leq L$. Second, we say that a function is β-smooth if its gradients are β-Lipschitz:

$$\|\nabla f(x) - \nabla f(x')\| \leq \beta\|x - x'\|.$$

2.4.2 Convex sets and functions

A set $\mathcal{X}$ is convex if for all $x_1, x_2 \in \mathcal{X}$ and $t \in [0,1]$, $tx_1 + (1-t)x_2 \in \mathcal{X}$. A function $f : \mathcal{X} \to \mathbb{R}$ is convex if, for all $x_1, x_2 \in \mathcal{X}$ and $t \in [0,1]$

$$tf(x_1) + (1-t)f(x_2) \geq f(tx_1 + (1-t)x_2).$$

A geometrical interpretation of the previous inequality is that if we draw a convex function in a paper, the straight line joining any two points in the graph of the function will lay above the graph of the function.

Convex sets $\mathcal{X}$ together with convex functions f define convex optimization problems:

$$\min_x f(x) \text{ such that } x \in \mathcal{X}.$$

Convex optimization problems are important because their local minima are global minima. This is in contrast to the nonconvex function depicted in Figure 2.2.

Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable and convex. Then,

$$f(x_2) \geq f(x_1) + \langle \nabla f(x_1),\, x_2 - x_1 \rangle.$$

A geometrical interpretation of the previous inequality is that the tangent line of a convex function at any point x_1 underestimates the function at all locations $x_2 \in \mathcal{X}$. Now let $f : \mathbb{R}^d \to \mathbb{R}$ be twice differentiable and convex. Then,

$$\nabla^2 f(x) \succeq 0.$$

A convex function is *strictly convex* if the previous three inequalities hold when replacing the "$\geq$" and "$\succeq$" symbols with the "$>$" and "$\succ$" symbols.

2.4.3 Gradient based methods

Gradient based methods start at a random location in the domain of the function of interest, and perform minimization by taking small steps along the direction of most negative gradient. Gradient based methods are therefore vulnerable to get stuck in local minima, that is, places where the gradient is very small but the function value is suboptimal. To understand this, see the example in Figure 2.2. If we start our gradient based optimization method at $x = -5$ and take sufficiently small steps, we will converge at the sub-optimal local minima $x = 0$. Another danger in this same example would be to get stuck in the *saddle point* around $x = 6$.

First order methods

The *first order* Taylor approximation of $f : \mathbb{R}^d \to \mathbb{R}$ at x_2 is

$$f(x_2) \approx f(x_1) + (x_2 - x_1)^\top \nabla f(x_1).$$

Imagine that we are optimizing f, and that we are currently positioned at x. Using the previous equation, and moving in the direction given by the unit vector u, we obtain

$$f(x+u) - f(x) \approx u^\top \nabla f(x).$$

Since we want to minimize $f(x+u)-f(x)$ using only function evaluations and function derivative evaluations, we should minimize $u^\top \nabla f(x)$ with respect to the direction unit vector u. This happens for $u = -\nabla f(x)/\|\nabla f(x)\|$. Thus, we can update our position *following the gradient descent*

$$x_{t+1} = x_t - \gamma \nabla f(x_t),$$

where $\gamma \in (0,1)$ is the *step size*, chosen smaller than the inverse of the Lipschitz constant of the function f.

Remark 2.7 *(Choosing the step size)***.** There exists a range of algorithms that provide a recipe to dynamically adjust the step size over the course of optimization. Most of these algorithms maintain a running average of the gradient, and adjust an individual step size per optimized variable, as a function of how much individual partial derivatives change over time. Some examples are the Adagrad algorithm (Duchi et al., 2011), and the RMSProp algorithm (Tieleman and Hinton, 2012). Another solution is to run a small amount of iterations of stochastic gradient descent with different step sizes, and select the step size giving best results for the rest of the optimization. $\Diamond$

Under some additional assumptions over the optimized function, it is possible to accelerate gradient descent methods using *Nesterov's accelerated gradient descent* (Nesterov, 2004).

Example 2.1 *(Stochastic gradient descent in learning)***.** In machine learning, we often optimize functions of the form

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x),$$

where n can be in the millions. Therefore, evaluating the gradients $\nabla f(x)$ is computationally prohibitive. *Stochastic gradient descent* (Bottou, 2010) is a modification of the gradient descent method, where the exact function gradients $\nabla f(x)$ are replaced with approximate gradients $\nabla f_i(x)$. Therefore, the update rules in stochastic gradient descent are

$$x_{t+1} = x_t - \gamma \nabla f_i(x).$$

The approximate gradients are also stochastic, because $i \sim \mathcal{U}[1,n]$ at each step. Stochastic gradients $\nabla f_i(x)$ are estimators of the gradients $\nabla f(x)$, and therefore exhibit variance. One compromise between the computational properties of stochastic gradient descent and the low variance of exact gradient

descent is to consider minibatches. In *minibatch gradient descent*, the update rules are

$$x_{t+1} = x_t - \frac{\gamma}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla f_i(x),$$

where $\mathcal{B}$ is a random subset of $m \ll n$ elements drawn from $\{1, \ldots, m\}$. $\Diamond$

Second order methods

The *second order* Taylor approximation of $f : \mathbb{R}^d \to \mathbb{R}$ at x_2 is

$$f(x_2) \approx f(x_1) + (x_2 - x_1)^\top \nabla f(x) + \frac{1}{2}(x_2 - x_1)^\top \nabla^2 f(x)(x_2 - x_1).$$

Therefore, moving from x in the direction given by u, we obtain

$$f(x + u) - f(x) \approx u^\top \nabla f(x) + \frac{1}{2} u^\top \nabla^2 u.$$

Therefore, based on function evaluations, first derivative evaluations, and second derivative evaluations, if we aim at minimizing $f(x + u) - f(x)$ we should minimize $u^\top \nabla f(x) + \frac{1}{2} u^\top \nabla^2 u$. This happens when

$$u = -(\nabla^2 f(x))^{-1} \nabla f(x).$$

Therefore, second order gradient descent methods implement the update rule

$$x_{t+1} = x_t - \gamma (\nabla^2 f(x_t))^{-1} \nabla f(x_t),$$

for some small step size $\gamma \in (0, 1)$. This is often called the Newton's update.

Remark 2.8 *(First order versus second order methods)***.** First order and second order gradient descent methods perform a local approximation of the optimized function at each point of evaluation. While first order methods perform a linear approximation, second order methods perform a quadratic approximation to learn something about the curvature of the function. Thus, second order methods use more information about f per iteration, and this translates in a fewer number of necessary iterations for convergence. After t iterations, the convex optimization error of first order methods is $O(L/\sqrt{t})$ for L-Lipschitz functions, $O(\beta/t)$ for β-smooth functions, and $O(\beta/t^2)$ for β-smooth functions when using Nesterov's accelerated gradient descent method. On the other hand, Newton's method generally achieves an optimization error of $O(1/t^2)$ (Bubeck, 2015).

Although second order methods need fewer iterations, each of their iterations is slower due to the inversion of the Hessian matrix. This operation requires $O(d^3)$ computations when optimizing a d-dimensional function. To alleviate these issues, *quasi-Newton* methods replace the Hessian matrix with a low-rank approximation which allows for faster inversion. One example is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm (see Nesterov (2004)). $\Diamond$

Chapter 3

Representing data

This chapter is a review of well-known results.

Pattern recognition is conceived in two steps. First, finding a *feature map*

$$\phi : \mathcal{X} \to \mathcal{H}$$

that transforms the *data* $\{x_i\}_{i=1}^n$, $x_i \in \mathcal{X}$, into the *representation* or *features* $\{\phi(x_i)\}_{i=1}^n$, $\phi(x_i) \in \mathcal{H}$. Second, revealing the pattern of interest in data as a linear transformation

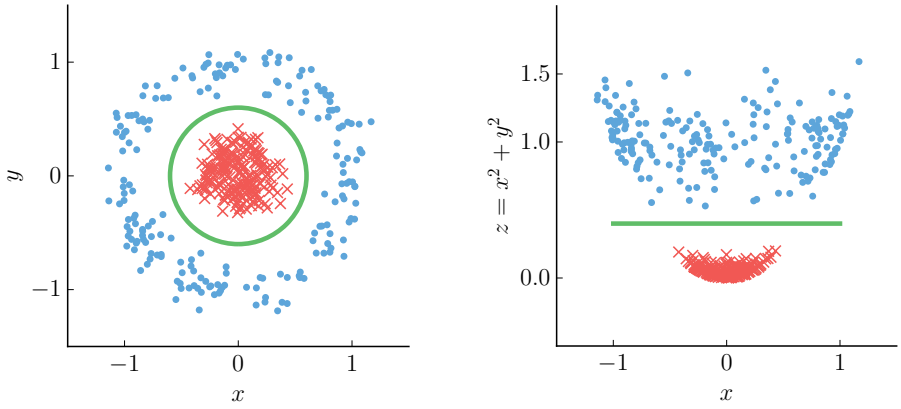
$$\langle A, \phi(x) \rangle_{\mathcal{H}} \tag{3.1}$$

of the representation.

Because of the simplicity of (3.1), most of the responsibility in learning from data falls in the feature map ϕ. Therefore, finding good feature maps is key to pattern recognition (Bengio et al., 2015). Good feature maps translate nonlinear statistics of data into linear statistics of their representation: they turn dependencies into correlations, stretch nonlinear relationships into linear regressions, disentangle the explanatory factors of data into independent components, and arrange different classes of examples into linearly separable groups. The following example illustrates the key role of representations in learning.

Example 3.1 *(Rings data).* Consider the problem of finding a linear function that separates the two classes of examples from Figure 3.1a. After some struggle, we conclude that under the raw representation $(x, y) \in \mathcal{X}$, no such function exists. To solve this issue, we engineer a third feature $z = x^2 + y^2$. This new feature elevates each example to an altitude proportional to its distance to the origin. Under the representation $(x, z) \in \mathcal{H}$, depicted in Figure 3.1b, there exists a linear function that separates the two classes of examples, solving the problem at hand. ◊

There exists a variety of methods to construct representations of data. In this chapter, we review four of them: kernel methods, random features, neural networks, and ensembles.



(a) Raw data (b) Data representation

Figure 3.1: The rings data

3.1 Kernel methods

The central object of study in kernel methods (Schölkopf and Smola, 2001) is the kernel function. Throughout this section, we assume that $\mathcal{X}$ is a compact metric space.

Definition 3.1 (Kernel function)**.** *A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite kernel function, or kernel, if for all $n \geq 1$, $x_1, \ldots, x_n \in \mathbb{R}$, and $c_1, \ldots, c_n \in \mathbb{R}$*

$$\sum_{i=1}^{n} c_i c_j k(x_i, x_j) \geq 0.$$

Each kernel k provides with a fixed feature map ϕ_k.

Definition 3.2 (Kernel representation)**.** *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel if and only if there exists a Hilbert space $\mathcal{H}$ and a* feature map *$\phi_k : \mathcal{X} \to \mathcal{H}$ such that for all $x, x' \in \mathcal{X}$*

$$k(x, x') = \langle \phi_k(x),\, \phi_k(x') \rangle_{\mathcal{H}},$$

We refer to $\phi_k(x) \in \mathcal{H}$ as a kernel representation of $x \in \mathcal{X}$.

Kernel representations often lack explicit closed forms, but we can access them implicitly using the inner products $\langle \phi_k(x),\, \phi_k(x') \rangle$ computed as $k(x, x')$. In general, there exists more than one feature map ϕ_k and Hilbert space $\mathcal{H}$ satisfying $k(x, x') = \langle \phi_k(x),\, \phi_k(x') \rangle_{\mathcal{H}}$, for a fixed given k. But, every kernel k is associated to an unique *Reproducing Kernel Hilbert Space* (RKHS) $\mathcal{H}_k$, with corresponding unique *canonical feature map* $k(x, \cdot) \in \mathcal{H}_k$, such that

$$k(x, x') = \langle k(x, \cdot),\, k(x', \cdot) \rangle_{\mathcal{H}_k}.$$

The following important result highlights a key property of reproducing kernel Hilbert spaces.

Theorem 3.1 (Moore-Aronszajn)**.** *Let $\mathcal{H}_k$ be a Hilbert space of functions from $\mathcal{X}$ to $\mathbb{R}$. Then, $\mathcal{H}_k$ is a RKHS if and only if there exists a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that*

$$\begin{aligned} &\forall x \in \mathcal{X}, \quad k(x, \cdot) \in \mathcal{H}_k, \\ &\forall f \in \mathcal{H}_k, \quad \langle f(\cdot), k(x, \cdot) \rangle \text{ (reproducing property)} \end{aligned}$$

If such k exists, it is unique, and k is the reproducing kernel of $\mathcal{H}_k$*. Every kernel k reproduces a unique RKHS $\mathcal{H}_k$.*

Proof. See Theorem 3 in (Berlinet and Thomas-Agnan, 2011). □

The reproducing property is attractive from a computational perspective, because it allows to express any function $f \in \mathcal{H}_k$ as the linear combination of evaluations of the reproducing kernel k. Presented next, the representer theorem leverages the reproducing property to learn patterns from data using kernels.

Theorem 3.2 (Representer)**.** *Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a kernel with corresponding RKHS $\mathcal{H}_k$. Assume data $\{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathcal{X} \times \mathbb{R}$, a strictly monotonically increasing function $g\colon [0, \infty) \to \mathbb{R}$, and an arbitrary risk function $R\colon (\mathcal{X} \times \mathbb{R}^2)^n \to \mathbb{R} \cup \{\infty\}$; then, any $f^\star \in H_k$ satisfying*

$$f^\star = \underset{f \in H_k}{\arg\min}\, R\left((x_1, y_1, f(x_1)), ..., (x_n, y_n, f(x_n))\right) + g\left(\|f\|\right)$$

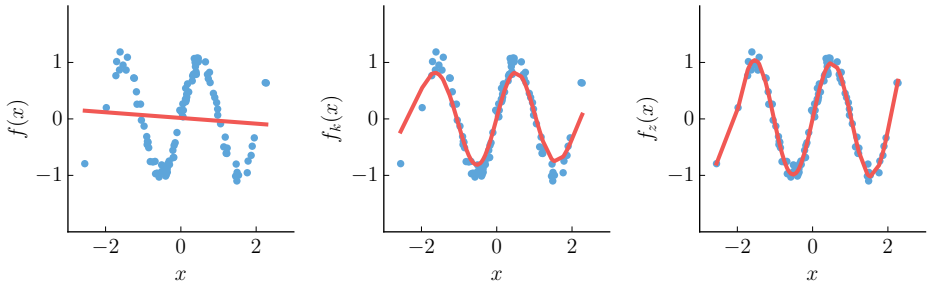
admits the representation

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot),$$

where $\alpha_i \in \mathbb{R}$ for all $1 \leq i \leq n$.

Proof. See Section 4.2. of (Schölkopf and Smola, 2001). □

Simply put, the representer theorem states that if we use a kernel function associated with a rich RKHS, we will be able to use it to learn rich patterns from data.



(a) Linear LS (b) Kernel LS (c) Randomized LS

Figure 3.2: Different types of least-squares regression.

3.1.1 Learning with kernels

Learning with kernels involves three steps. First, stating the learning problem of interest in terms of the Gram matrix $G \in \mathbb{R}^{n \times n}$, with entries $G_{ij} = \langle x_i, x_j \rangle$, for all pairs of inputs (x_i, x_j). Second, replacing the Gram matrix G by the kernel matrix $K \in \mathbb{R}^{n \times n}$, with entries $K_{ij} = k(x_i, x_j)$. Third, solving the learning problem by computing linear statistics of the kernel matrix K. This manipulation is known as the *kernel trick*.

The following example illustrates the use of the kernel trick to extend the capabilities of least-squares regression to model nonlinear relations between random variables.

Example 3.2 *(Kernel least-squares regression).* Recall the problem of least squares, described in Section 2.3.3. Figure 3.2 illustrates a one-dimensional dataset where $\boldsymbol{x} \equiv \mathcal{N}(0, 1)$ and $\boldsymbol{y} = \sin(3\boldsymbol{x}) + \mathcal{N}(0, \lambda^2)$. As shown in Figure 3.2a, *linear* least-squares regression fails to recover the true nonlinear relationship $f(x) = \mathbb{E}[\boldsymbol{y}|x] = \sin(3x)$. We solve this issue by performing least-squares regression on some kernel representation ϕ_k. To apply the kernel trick, we must first state (2.13) in terms of the $n \times n$ Gram matrix $G = XX^\top$. For this, we use the Sherman-Morrison-Woodbury formula (2.1) to rewrite (2.13) as

$$\hat{\alpha} = (X^\top I_n X + \lambda I_d)^{-1} X^\top I_n y = X^\top (XX^\top + \lambda I_n)^{-1} y.$$

Then, our regression function is

$$f(x) = \langle \hat{\alpha}, x \rangle = \left\{ (XX^\top + \lambda I_n)^{-1} y \right\}^\top Xx^\top.$$

Next, replace the Gram matrix $XX^\top \in \mathbb{R}^{n \times n}$ by the kernel matrix $K \in \mathbb{R}^{n \times n}$ with entries $K_{ij} = k(x_i, x_j)$, and the vector $Xx^\top \in \mathbb{R}^n$ by the vector $k_x \in \mathbb{R}^n$ with entries $k_{x,i} = k(x, x_i)$:

$$f_k(x) = \langle \hat{\alpha}, x \rangle = \left((K + \lambda I_n)^{-1} y \right)^\top k_x = \sum_{i=1}^{n} \beta_i k(x_i, x), \tag{3.2}$$

where $\beta_i = \big((K + \lambda I_n)^{-1} y\big)_i$, for all $1 \leq i \leq n$. Figure 3.2b illustrates the least-squares regression obtained using the kernel representation, which successfully captures the nonlinear pattern describing the data. $\Diamond$

Example 3.2 reveals a key property of kernel representations. As shown in (3.2), the nonlinear regression function f_k is a linear transformation of the n-dimensional representation

$$(k(x_1, x), \ldots, k(x_n, x)),$$

also called the *empirical kernel map*. Such kernel representations are *nonparametric*: given n data, kernels representations are effectively n-dimensional. Nonparametric representations are a double-edged sword. On the positive side, nonparametric representations allow each point x_i in the data to speak by itself, as one dedicated dimension of the representation. This makes intuitive sense, because when having more data, we should be able to afford a more sophisticated representation. On the negative side, learning using n-dimensional representations requires computations prohibitive for large n. In the previous example, the computational burden is $O(n^3)$ due to the construction and inversion of the $n \times n$ kernel matrix K. Furthermore, kernel machines (3.2) need access to all the data $\{x_i\}_{i=1}^n$ for their evaluation, thus requiring $O(nd)$ permanent storage.

Remark 3.1 *(Nonparametric versus parametric representations)***.** The dimensionality of nonparametric representations grows linearly with the amount of data n, but not with respect to the complexity of the pattern of interest. Even when recovering a simple pattern from $n = 10^6$ samples of $d = 10^3$ dimensions, an orthodox use of kernels will require $O(10^{18})$ computations and $O(10^9)$ memory storage. As we will see later in this chapter, *parametric* representations are attractive alternatives to deal with big data, since we can tune their size according to the difficulty of the learning problem at hand. In any case, nonparametric representations are essentially parameter-free, since they use the given training data as parameters. This translates in learning algorithms with a small amount tunable parameters, which is a desirable property.

Moreover, nonparametric representations are useful when the dimensionality of our data d is greater than the sample size n. In this case, computing the (dual) $n \times n$ kernel matrix is cheaper than computing the (primal) $d \times d$ covariance matrix of the data. $\Diamond$

3.1.2 Examples of kernel functions

There exists a wide catalog of kernel functions (Souza, 2010). Favouring the choice of one kernel over another is a problem specific issue, and will depend on the available prior knowledge about the data under study. For example, the Gaussian kernel is an effective choice to discover smooth

patterns. Alternatively, the arc-cosine kernel is a better choice to model patterns with abrupt changes. Or, if data contains patterns that repeat themselves, periodical kernels induce more suitable data representations.

Kernel functions have closed-form expressions and a small number of tunable parameters. The simplest kernel function is the *polynomial kernel*

$$k(x, x') = (\langle x,\, x' \rangle + c)^d,$$

with offset parameter $c \geq 0$ and degree parameter $d \in \mathbb{N}$. For $c = 0$ and $d = 1$, the representation $\phi_k(x)$ induced by the polynomial kernel matches the original data x. As d grows, the polynomial kernel representation captures increasingly complex patterns, described by polynomials of degree d. The offset parameter c trades-off the influence between the higher-order and the lower-order terms in the polynomial representation.

The most widely-used kernel function is the *Gaussian kernel*

$$k(x, x') = \exp\left(-\gamma \|x - x'\|_2^2\right). \tag{3.3}$$

The *bandwidth* parameter $\gamma \geq 0$ controls the complexity of the representation. Large values of γ induce more complex representations, and small values of γ induce representations closer to the original raw data. In practice, one sets γ to roughly match the scale of the data. One alternative to do this is the *median heuristic*, which selects γ to be the inverse of the empirical median of the pairwise distances $\|x_i - x_j\|_2^2$, with (x_i, x_j) subsampled from the data. The Gaussian kernel is differentiable an infinite amount of times, making it appropriate to model smooth patterns.

Another important kernel function is the *arc-cosine kernel*

$$\begin{aligned} k(x, x') &= 2 \int \frac{\exp(-\frac{1}{2} \|w\|^2)}{(2\pi)^{d/2}} \Theta(\langle w,\, x \rangle) \Theta(\langle w,\, x' \rangle) \langle w,\, x \rangle^q \langle w,\, x' \rangle^q \mathrm{d}w, \\ &= \frac{1}{\pi} \|x\|^q \left\|x'\right\|^q (-1)^q (\sin\theta)^{2q+1} \left(\frac{1}{\sin\theta} \frac{\partial}{\partial\theta}\right)^q \left(\frac{\pi - \theta}{\sin\theta}\right), \end{aligned} \tag{3.4}$$

where $\theta := \cos^{-1}\left(\langle x,\, x' \rangle / \left(\|x\| \left\|x'\right\|\right)\right)$ and $q \in \mathbb{N}$ (Cho and Saul, 2011). For $q = 1$, the arc-cosine kernel data representation is piece-wise linear, and properly describes patterns exhibiting abrupt changes.

Finally, we can construct new kernels as the combination of other kernels. For instance, if $k_1(x, x')$ and $k_2(x, x')$ are two kernels, then $k(x, x') = k_1(x, x') + k_2(x, x')$ and $k(x, x') = k_1(x, x') k_2(x, x')$ are also kernels. Or, for any kernel k and function $f : \mathbb{R}^d \to \mathbb{R}$, $k(x, x') = f(x) k(x, x') f(x')$ and $k(x, x') = k(f(x), f(x'))$ are also kernels. Bishop (2006, page 296) offers a detailed table of rules to build kernels out of other kernels. Duvenaud et al. (2013) proposes a genetic algorithm that explores these kind of compositions to evolve complex kernels with interpretable meanings.

3.2 Random features

We have seen that kernel methods induce nonparametric representations, that is, representations that have n effective dimensions when learning from n data. One drawback of nonparametric representations is their associated computational requirements. For instance, kernel least-squares requires $O(n^3)$ computations and $O(nd)$ memory storage. As n grows to the tens of thousands, these computational and memory requirements become prohibitive.

This section proposes two alternatives to approximate n-dimensional nonparametric kernel representations as m-dimensional parametric representations, where m can depend on the complexity of the learning problem at hand. In some cases, m will be much smaller than n.

3.2.1 The Nyström method

The Nyström method (Williams and Seeger, 2001) approximates the representation induced by a kernel k as

$$\phi(x) = M^{-1/2} \left(k(w_1, x), \ldots, k(w_m, x)\right)^\top \in \mathbb{R}^m, \tag{3.5}$$

where the matrix $M \in \mathbb{R}^{m \times m}$ has entries $M_{ij} = k(w_i, w_j)$, with $w_i \in \mathbb{R}^d$ for all $1 \leq j \leq m$. In practice, the set $\{w_1, \ldots, w_m\}$ is a subset of the data $\{x_i\}_{i=1}^n$ sampled at random, or m representative data prototypes computed using a clustering algorithm (Kumar et al., 2012).

The analysis of the Nyström approximation considers the rank-m approximate kernel matrix

$$K_m = \Phi\Phi^\top \in \mathbb{R}^{n \times n},$$

where

$$\Phi := \left(\phi(x_i), \ldots, \phi(x_i)\right)^\top \in \mathbb{R}^{n \times m},$$

and ϕ follows (3.5). If the set $\{w_1, \ldots, w_m\}$ is a subset of the data sampled using a carefully chosen probability distribution (Drineas and Mahoney, 2005), then

$$\|K - K_m\|_2 \leq \|K - K_m^\star\|_2 + O\left(\frac{n}{\sqrt{m}}\right),$$

where $K_m^\star$ is the best rank-m approximation to K.

The Nyström method has two main advantages. First, it allows the approximation of arbitrary kernel representations. Second, the set $\{w_1, \ldots, w_m\}$ is an opportunity to adapt the representation to the geometry of the data at hand. This adaptation results in a reduction of the Nyström approximation error from $O(\frac{n}{\sqrt{m}})$ to $O(\frac{n}{m})$ when the gap between the two largest eigenvalues of the true kernel matrix K is large (Yang et al., 2012). On the negative side, Nyström approximations face the same problems than regular kernel representations: it is necessary to construct and invert the $m \times m$ matrix

M, multiply against it to construct the approximate kernel representation, and store the set $\{w_1, \ldots, w_m\}$ in $O(md)$ memory at all times. Like in exact kernel methods, this requires a prohibitive amount of computation when a large amount m of representation features is necessary.

In the following, we review *random Mercer features*, an alternative approximation to a specific class of kernel representations which overcomes the two short-comings of the Nyström method.

3.2.2 Random Mercer features

Random Mercer features approximate kernel functions which satisfy *Mercer's condition*, by exploiting their expansion as a sum.

Theorem 3.3 (Mercer's condition)**.** *Let $\mathcal{X}$ be a compact metric space, and let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a continuous kernel which is square-integrable on $\mathcal{X} \times \mathcal{X}$ and satisfies*

$$\int_{\mathcal{X}} \int_{\mathcal{X}} k(x, x') f(x) f(x') \mathrm{d}x \mathrm{d}x' \geq 0$$

for all $f \in L^2(\mathcal{X})$. Then, k admits a representation

$$k(x, x') = \sum_{j=1}^{\infty} \lambda_j \phi_{\lambda_j}(x) \phi_{\lambda_j}(x'), \tag{3.6}$$

where $\lambda_j \geq 0$, $\langle \phi_i, \phi_j \rangle = \delta_{ij}$, and the convergence is absolute and uniform.

Proof. See (Mercer, 1909). □

As pioneered by Rahimi and Recht (2007, 2008); Le et al. (2013), one can approximate the expansion (3.6) by random sampling. More specifically, for trace-class kernels, those with finite $\|\lambda\|_1 := \sum_{i=1}^{\infty} \lambda_j$, we can normalize the kernel expansion (3.6) to mimic an expectation

$$k(x, x') = \|\lambda\|_1 \mathbb{E}_{\lambda \sim p(\lambda)} \left[\phi_\lambda(x) \phi_\lambda(x') \right], \tag{3.7}$$

where

$$p(\lambda) = \begin{cases} \|\lambda\|_1^{-1} \lambda & \text{if } \lambda \in \{\lambda_1, \ldots\}, \\ 0 & \text{else.} \end{cases}$$

Now, by sampling $\lambda_j \sim p(\lambda)$, for $1 \leq j \leq m$, we can approximate the expectation (3.7) with the Monte-Carlo sum

$$k(x, x') \approx \frac{\|\lambda\|_1}{m} \sum_{j=1}^{m} \phi_{\lambda_j}(x) \phi_{\lambda_j}(x'),$$

from which we can recover the m-dimensional, parametric representation

$$\phi(x) = \sqrt{\frac{\|\lambda\|_1}{m}} \left(\phi_{\lambda_1}(x), \ldots, \phi_{\lambda_m}(x) \right)^\top. \tag{3.8}$$

The functions $\{\phi_{\lambda_j}\}_{j=1}^{\infty}$ are often unknown or expensive to compute. Fortunately, there are some exceptions. For example, the arc-cosine kernel (3.4) follows the exact form of an expectation under the d-dimensional Gaussian distribution. Therefore, by sampling $w_1, \ldots, w_m \sim \mathcal{N}(0, 1)$, we can approximate (3.4) by

$$\begin{aligned} k(x, x') &= 2 \int \frac{\exp(-\frac{1}{2}\|w\|^2)}{(2\pi)^{d/2}} \Theta(\langle w,\, x\rangle)\Theta(\langle w,\, x'\rangle)\langle w,\, x\rangle^q \langle w,\, x'\rangle^q \mathrm{d}w \\ &\approx \frac{2}{m} \sum_{j=1}^{m} \Theta(\langle w,\, x\rangle)\Theta(\langle w,\, x'\rangle)\langle w,\, x\rangle^q \langle w,\, x'\rangle^q. \end{aligned} \tag{3.9}$$

For instance, consider $q = 1$. Then, we can combine (3.8) and (3.9) to construct the m-dimensional representation

$$\phi(x) = \sqrt{\frac{2}{m}} \left(\max(\langle w_1,\, x\rangle), \ldots, \max(\langle w_m,\, x'\rangle)\right)^\top \in \mathbb{R}^m,$$

formed by rectifier linear units, which approximates the arc-cosine kernel, in the sense that $\langle \phi(x),\, \phi(x')\rangle$ converges to (3.4) pointwise as $m \to \infty$.

Another class of kernels with easily computable basis $\{\phi_j\}_{j=1}^{\infty}$ is the class of continuous shift-invariant kernels, those satisfying $k(x, x') = k(x - x', 0)$ for all $x, x' \in \mathcal{X}$. In this case, $\{\phi_j\}_{j=1}^{\infty}$ is the Fourier basis, as hinted by the following result due to Salomon Bochner.

Theorem 3.4 (Bochner)**.** *A function k defined on a locally compact Abelian group G with dual group G' is the Fourier transform of a positive measure p on G' if and only if it is continuous and positive definite.*

Proof. See Section 1.4.3 from (Rudin, 1962). □

One consequence of Bochner's theorem is that continuous shift-invariant kernels $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ are the Fourier transform of a positive measure defined on $\mathbb{R}^d$ (Rahimi and Recht, 2007, 2008). Then,

$$\begin{aligned} k(x - x', 0) &= c_k \int_\Omega p_k(w) \exp\left(\imath\langle w,\, x - x'\rangle\right) \mathrm{d}w \\ &= c_k \int_\Omega p_k(w) \cos(\langle w,\, x - x'\rangle) + \imath \sin(\langle w,\, x - x'\rangle) \mathrm{d}w \end{aligned}$$

where $p_k(w)$ is a positive measure and c_k is a normalization constant, both depending on k. If the kernel function k and the probability measure p are real,

$$\begin{aligned} k(x - x', 0) &= c_k \int_\Omega p_k(w) \cos(\langle w,\, x - x'\rangle) + \imath \sin(\langle w,\, x - x'\rangle) \mathrm{d}w \\ &= c_k \int_\Omega p_k(w) \cos(\langle w,\, x - x'\rangle) \mathrm{d}w. \end{aligned}$$

Next, using the trigonometric identity

$$\cos(a-b) = \frac{1}{\pi}\int_0^{2\pi} \cos(a+x)\cos(b+x)\mathrm{d}x,$$

it follows that

$$\begin{aligned} k(x-x',0) &= c_k \int_\Omega p_k(w)\cos(\langle w,\, x-x'\rangle)\mathrm{d}w \\ &= \frac{c_k}{\pi}\int_\Omega\int_0^{2\pi} p_k(w)\cos(\langle w,\, x\rangle + b)\cos(\langle w,\, x'\rangle + b)\mathrm{d}w\mathrm{d}b \\ &= 2c_k \int_\Omega\int_0^{2\pi} p_k(w)u(b)\cos(\langle w,\, x\rangle + b)\cos(\langle w,\, x'\rangle + b)\mathrm{d}w\mathrm{d}b, \end{aligned} \tag{3.10}$$

where $u(b) = (2\pi)^{-1}$ is the uniform distribution on the closed interval $[0, 2\pi]$. We can approximate this expression by drawing m samples $w_1, \ldots, w_m \sim p$, m samples $b_1, \ldots, b_m \sim u$, and replacing the integral (3.10) with the sum

$$\begin{aligned} k(x,x') &= 2c_k \int_\Omega\int_0^{2\pi} p_k(w)u(b)\cos(\langle w,\, x\rangle + b)\cos(\langle w,\, x'\rangle + b)\mathrm{d}w\mathrm{d}b \\ &\approx \frac{2c_k}{m}\sum_{j=1}^{m}\cos(\langle w_j,\, x\rangle + b_j)\cos(\langle w_j,\, x'\rangle + b_j). \end{aligned}$$

From this, we can recover the m-dimensional, explicit representation

$$\phi(x) = \sqrt{\frac{2c_k}{m}}\left(\cos(\langle w_1,\, x\rangle + b_1), \ldots, \cos(\langle w_m,\, x\rangle + b_m)\right)^\top \in \mathbb{R}^m,$$

which approximates the associated shift-invariant kernel k in the pointwise convergence

$$\langle \phi(x),\, \phi(x')\rangle_{\mathbb{R}^m} \to k(x-x',0)$$

as $m \to \infty$.

Example 3.3 *(Gaussian kernel).* The Gaussian kernel (3.3) is shift-invariant, and its Fourier transform is the Gaussian distribution $\mathcal{N}(0, 2\gamma I_d)$. Therefore, the map

$$\phi(x) = \sqrt{\frac{2}{m}}\left(\cos(\langle w_1,\, x\rangle + b_1), \ldots, \cos(\langle w_m,\, x\rangle + b_m)\right)^\top \in \mathbb{R}^m \tag{3.11}$$

with $w_j \sim \mathcal{N}(0, 2\gamma I_d)$ and $b_j \sim \mathcal{U}[0, 2\pi]$ for all $1 \leq j \leq m$ approximates the Gaussian kernel in the sense of the pointwise convergence

$$\langle \phi(x),\, \phi(x')\rangle_{\mathbb{R}^m} \to \exp\left(-\gamma \left\|x - x'\right\|\right)$$

as $m \to \infty$. ◊

Remark 3.2 *(Computing Gaussian random features faster).* Constructing the representation (3.11) involves computing the dot product $\langle W, x\rangle$, where $W \in \mathbb{R}^{d\times m}$ is a matrix of Gaussian random numbers. Naïvely, this is a $O(md)$ computation. Le et al. (2013) introduce *Fastfood*, a technique to approximate dot products involving Gaussian matrices W, accelerating their computation from $O(md)$ to $O(m \log d)$ operations. Fastfood replaces the Gaussian matrix W with a concatenation of $d \times d$ blocks with structure

$$V := \frac{1}{\sigma\sqrt{d}} SHG\Pi HB,$$

where $\Pi \in \{0,1\}^{d\times d}$ is a permutation matrix, and H is the Walsh-Hadamard matrix. S, G and B are *diagonal* matrices containing, in order, kernel function dependent scaling coefficients, Gaussian random numbers, and random $\{-1, +1\}$ signs. All matrices allow sub-quadratic computation, and the only storage requirements are the $m \times m$ diagonal matrices S, G, B. Le et al. (2013) provide with an analysis of the quality of the Fastfood approximation.

The benefits of Fastfood are most noticeable when representing high-dimensional data. For instance, when working with color images of 32×32 pixels, Fastfood allows to compute the representations (3.11) up to 265 times faster. ◊

For other examples of shift-invariant kernel approximations using Bochner's theorem, see Table 1 of (Yang et al., 2014). Sriperumbudur and Szabó (2015) characterize the approximation error of d-dimensional shift-invariant kernels on $\mathcal{S} \subset \mathbb{R}^d$ using Bochner's theorem

$$\mathbb{P}\left(\sup_{x,x'\in\mathcal{S}} \left|\hat{k}(x,x') - k(x,x')\right| \geq \frac{h(d,|\mathcal{S}|,c_k) + \sqrt{2t}}{\sqrt{m}}\right) \leq \exp\left(-t\right), \quad (3.12)$$

where $\mathcal{S} \subset \mathbb{R}^d$ is a compact set of diameter $|\mathcal{S}|$, and

$$h(d,|\mathcal{S}|,c_k) = 32\sqrt{2d\log(|\mathcal{S}|+1)} + 32\sqrt{2d\log(c_k+1)} + 16\sqrt{2d(\log(|\mathcal{S}|+1))^{-1}}. \quad (3.13)$$

Remark 3.3 *(Multiple kernel learning).* Random feature maps allow the use of different representations simultaneously. For instance, we could sample

$$\begin{aligned} w_1, \ldots, w_m &\sim \mathcal{N}(0, 2\cdot 0.1 \cdot I_d), \\ w_{m+1}, \ldots, w_{2m} &\sim \mathcal{N}(0, 2\cdot 1 \cdot I_d), \\ w_{2m+1}, \ldots, w_{3m} &\sim \mathcal{N}(0, 2\cdot 10 \cdot I_d), \end{aligned}$$

to construct a $3m$-dimensional representation approximating the sum of three Gaussian kernels with bandwidths γ of 0.1, 1, and 10. Or, for example, we could construct a $2m$-dimensional representation where the first half

m random features approximate a Gaussian kernel, and the second half of m random features approximate an arc-cosine kernel. This strategy is closely related to multiple kernel learning (Gönen and Alpaydın, 2011). The concatenation of random feature maps approximate the sum of their associated kernels. The outer-product of random feature maps approximates the product of their associated kernels. Finally, it is possible to learn the distribution $p_k(w)$ from which we sample the random features (Băzăvan et al., 2012; Wilson, 2014). ◊

As opposed to the Nyström method, random features do not require the multiplication of any $m \times m$ matrix for their construction, a costly operation for large m. Furthermore, random features do not require storage, since they can be efficiently resampled at test time. On the negative side, and as opposed to the Nyström method, random features are independent from the data under study. Therefore, complex learning problems require the use of large amounts of random features. For example, Huang et al. (2014) used 400.000 random features to build a state-of-the-art speech recognition system. To sum up, the intuition behind random features is that each random feature provides with a random summary or view of the data. Thus, when using large amounts of random features, chances are that linear combinations of these random views can express any reasonable pattern of interest.

Remark 3.4 *(Boltzmann brains).* An early consideration of structure arising from randomness is due to Ludwig Boltzmann (1844-1906). Under the second law of thermodynamics, our universe evolves (modulo random fluctuations) from low to high entropy states, that is, from highly ordered states to highly unordered states. Such direction of time, imposed by increasing entropy, strongly contradicts the existence and evolution of organized life forms. Therefore, Boltzmann argues that our existence is a random departure from a higher-entropy universe. Using this argument, Boltzmann concludes that it is much more likely for us to be self-aware entities floating in a near-equilibrium thermodynamic soup (and be called *Boltzmann brains*) instead of highly-organized physical beings embedded in a highly-organized environment, like our perception suggests. Consequently, our knowledge about the universe is highly biased: we observe this unlikely low-entropy universe because it is the only one capable of hosting life; this bias is the *anthropic principle*. ◊

As with kernels, we exemplify the use of random features on a regression problem.

3.2.3 Learning with random features

Learning with random features involves two steps. First, transforming the data $\{x_i\}_{i=1}^n$ into the representation $\{\phi(x_i)\}_{i=1}^n$, where ϕ follows (3.8) for some kernel k. Second, solving the learning problem by performing linear

statistics on the random representation.

Example 3.4 *(Randomized least-squares regression).* Recall Example 3.2. The solution to the least-squares regression problem (2.13) is

$$\hat{\alpha} = \operatorname*{arg\,min}_{\alpha \in \mathbb{R}^d} R(\alpha, \lambda, \mathcal{D}) = (X^\top X + \lambda I_d)^{-1} X^\top y,$$

where $X = (x_1, \ldots, x_n)^\top \in \mathbb{R}^{n \times d}$ and $y = (y_1, \ldots, y_n)^\top \in \mathbb{R}^n$.

To model nonlinear relationships using random features, replace the data $\{x_i\}_{i=1}^n$ with the m-dimensional random representation $\{\phi(x_i)\}_{i=1}^n$ from (3.8). Then solve again the least-squares problem, this time to obtain the m-dimensional vector of coefficients

$$\hat{\beta} = \operatorname*{arg\,min}_{\beta \in \mathbb{R}^m} R(\alpha, \lambda, \{(\phi(x_i), y_i)\}) = (\Phi^\top \Phi + \lambda I_d)^{-1} \Phi^\top y,$$

where $\Phi = (\phi(x_1), \ldots, \phi(x_n))^\top \in \mathbb{R}^{n \times m}$ and $y = (y_1, \ldots, y_n)^\top \in \mathbb{R}^n$. This produces the regression function

$$f_z(x) = \langle \beta, \, \phi(x) \rangle,$$

which successfully captures the nonlinear pattern in data, as depicted in Figure 3.2c. Learning this function takes $O(m^2 n)$ time, while the exact kernel solution from Example 3.2 took $O(n^3)$ time, a much longer computation for $n \gg m$. ◊

Kernel and random representations are independent from the data under study. Instead of relying on fixed data representations, it should be possible to *learn* them from data. This is the philosophy implemented by neural networks, reviewed next.

3.3 Neural networks

In the beginning of this chapter, we claimed that pattern recognition involves two steps. First, transforming the data $x_i \in \mathcal{X}$ into a suitable representation $\phi(x_i) \in \mathcal{H}$. Second, inferring the pattern of interest in the data as a linear statistic $\langle A, \, \phi(x) \rangle$ of the representation.

Kernel and random representations, reviewed in the previous two sections, approach pattern recognition in a rather simple way: they apply a *fixed* feature map ϕ to the data, and then perform linear operations in the associated fixed representation. More specifically, kernel and random representations have form

$$\phi(x) = (\sigma(x, w_1, b_1), \ldots, \sigma(x, w_m, b_m))^\top \in \mathbb{R}^m$$

for some *fixed* set of parameters $w_1, \ldots, w_m \in \mathbb{R}^d$, $b_1, \ldots, b_m \in \mathbb{R}$, and *nonlinearity* function $\sigma : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. In kernels, $\sigma(x, w_j, b_j) = k(x, w_j)$, with $w_j = x_j$ for all $1 \leq j \leq m = n$. In random features, we choose the

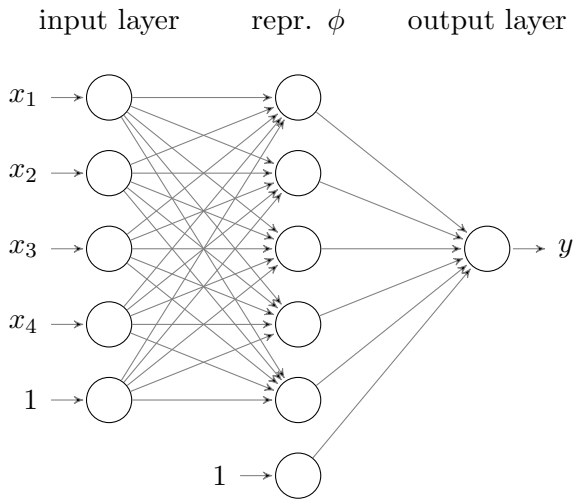


Figure 3.3: A shallow neural network.

representation dimensionality m *a priori*, by considering the complexity of the learning problem at hand, the amount of available data, and our computational budget. Then, each parameter $w_j \sim p_k(w)$, $b_j \sim \mathcal{U}[0, 2\pi]$, and $\sigma(x, w_j, b_j) = \sqrt{\frac{2}{m}} \cos(\langle w_j, x \rangle + b_j)$ for all $1 \leq j \leq m$. In both cases, the feature map ϕ is independent from the data: only a small number of tunable parameters, such as the degree for the polynomial kernel or the variance of the Gaussian random features, are adaptable to the problem at hand using cross-validation (see Section 2.3.2).

We can parallel the previous exposition to introduce neural networks, and highlight an structural equivalence between them and kernel methods. In neural networks, the nonlinearity function $\sigma(z)$ is fixed to the rectifier linear unit $\sigma(z) = \max(z, 0)$, the hyperbolic tangent $\sigma(z) = \tanh(z)$, or the sigmoid $\sigma(z) = (1 + \exp(-z))^{-1}$, to name a few. However, the representation parameters $\{(w_j, b_j)\}_{j=1}^m$ are not fixed but learned from data. This is a challenging task: neural networks often have millions of parameters, so training them requires the approximation of a high-dimensional, nonconvex optimization problem. This is a challenging task both from a computational perspective (solving such high-dimensional optimization problems), and an statistical perspective (properly tuning millions of parameters calls for massive data). For a historical review on artificial neural networks, we recommend the introduction of (Bengio et al., 2015).

Neural networks are organized in a sequence of layers, where each layer contains a vector of neurons. The neurons between two subsequent layers are connected by a matrix of *weights*. The strength of the weights connecting two neurons is one real number contained in the parameter set $\{(w_j, b_j)\}_{j=1}^m$.

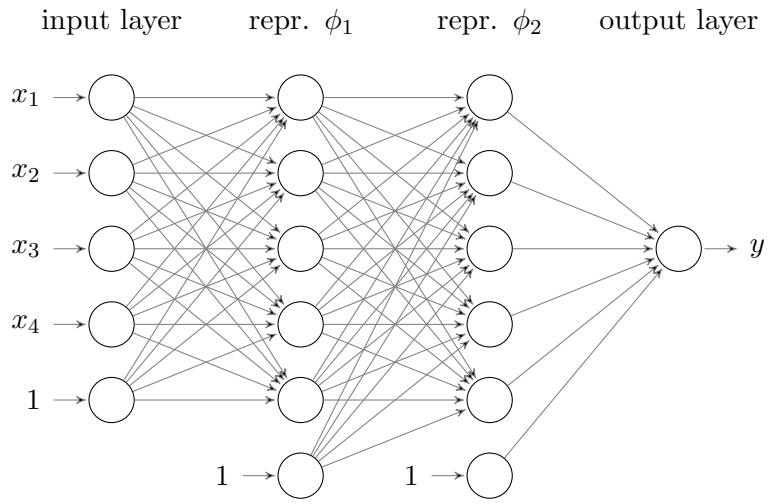


Figure 3.4: A *deep* neural network.

Neural networks contain three types of layers: input, hidden, and output layers. First, the input layer receives the data. Second, the data propagates forward from the input layer to the hidden layer, who is in charge of computing the data representation. In particular, the j-th neuron in the hidden layer computes the j-th feature $\phi(x)_j = \sigma(\langle w_j, x\rangle + b_j)$ of the representation, for all $1 \leq j \leq m$. Third, the representation propagates forward from the hidden layer to the output layer. Finally, the output layer returns the pattern of interest, computed as the linear transformation $\langle A, \phi(x)\rangle$ of the hidden layer representation. Figure 3.3 illustrates a neural network, where each circle depicts a neuron, and each arrow depicts a weight connecting two neurons from subsequent layers together. The depicted network accepts as input four-dimensional data through its input layer, transforms it into a five-dimensional representation on its hidden layer, and outputs the one-dimensional pattern

$$y = f(x_1, x_2, x_3, x_4) = \langle \alpha, \sigma(\langle (W, b), (x, 1)\rangle)\rangle + \beta$$

through its output layer. Neural networks like the one depicted in Figure 3.3 are *fully connected* neural networks, since all the neurons in a given layer connect to all the neurons in the next layer.

3.3.1 Deep neural networks

Deep neural networks implement data representations computed as the composition of *multiple* hidden layers. For instance, the neural network depicted in Figure 3.4 has two layers, which implement the representation

$$\phi(x) = \phi_2(\phi_1(x)) \in \mathbb{R}^5,$$

where

$$\phi_2 : \mathbb{R}^5 \to \mathbb{R}^5, \quad \phi_2(z) = (\sigma(\langle z,\, w_{2,1}\rangle + b_{2,1}), \ldots, \sigma(\langle z,\, w_{2,5}\rangle + b_{2,5}))^\top \in \mathbb{R}^5,$$
$$\phi_1 : \mathbb{R}^4 \to \mathbb{R}^5, \quad \phi_1(x) = (\sigma(\langle x,\, w_{1,1}\rangle + b_{1,1}), \ldots, \sigma(\langle x,\, w_{1,5}\rangle + b_{1,5}))^\top \in \mathbb{R}^5,$$

and $z = \phi_1(x)$, $w_{1,j} \in \mathbb{R}^4$, $w_{2,j} \in \mathbb{R}^5$, and $b_{1,j}, b_{2,j} \in \mathbb{R}$ for all $1 \leq j \leq 5$.

More generally, the parameters of a deep neural network are a collection of weight matrices $W_1, \ldots, W_L$, with $W_i \in \mathbb{R}^{m_{i-1} \times m_i}$, $m_0 = d$, $m_L = m$, biases $b_1, \leq, b_L \in \mathbb{R}$, and compute the representation

$$\begin{aligned} \phi(x) &= \phi_L, \\ \phi_l &= \sigma(\langle W_l,\, \phi_{l-1}\rangle + b_l), \\ \phi_0 &= x, \end{aligned}$$

where the nonlinearity $\sigma : \mathbb{R} \to \mathbb{R}$ operates entrywise. The number of free parameters in a deep representation is $O(m_0 m_1 + m_1 m2 + \ldots + m_{L-1} m_L)$. Note the contrast with the number of parameters of a nonparametric Gaussian kernel machine; most likely, two: the Gaussian kernel bandwidth, and the regression regularizer.

Deep neural networks are hierarchical compositions of representations, each capturing increasingly complex patterns from data. Each hidden layer takes as input the output of the previous layer, and processes it to learn a slightly more abstract representation of the data. For example, when training deep neural networks to recognize patterns from images, the first representation ϕ_1 detects edges of different orientations from raw pixels in the image, and the subsequent representations $\phi_2, \ldots, \phi_L$ learn how to combine those edges into parts, those parts into objects, and so on.

Representing data with *deep models*, also known as *deep learning*, has been the most successful technique to learn intricate patterns from large data in recent years, defining the new state-of-the-art in complex tasks such as image or speech recognition (Bengio et al., 2015; LeCun et al., 2015).

One key property fueling the power of deep representations is that these are *distributed* representations (Hinton et al., 1986). This concept is better understood using a simple example. Consider the task of classifying images of cars. If using a kernels, our n-dimensional representation $\phi(x)$ would contain one feature $\phi(x)_j = k(x, x_j)$ per car image x_j, for all $1 \leq j \leq n$. This means that our representation would contain one dedicated feature describing the image "small yellow Ferrari", and another dedicated feature describing the image "big red Tesla". These representations are *local* representations, and partition our data in an number of groups *linear* in the sample size n. On the other hand, compositional architectures such as deep neural networks could arrange their representation to depict the three binary features "yellow or red", "small or big", and "Ferrari or Tesla". Each of these three binary features is the computation implemented by a sequence of hidden layers, which

process and recombine the data in multiple different ways. The key point here is that these three binary features exhibit a many-to-many relationship with respect to the data: many samples in the data are partially described by the same feature, and many features describe data example. Importantly, these *distributed* representations, these "attribute sharing" structure of data, allow a separation in a number of groups *exponential* in the dimensionality d: the three binary features in our example can describe an exponential amount of 2^3 different images of cars.

Remark 3.5 *(Is it necessary to be deep?)***.** Universal kernels learn, up to an arbitrary precision, any continuous bounded pattern from data. Therefore, why should we care about deep neural network representations, their millions of parameters, and their complicated numerical optimization?

Because when learning some functions, restricting the representation to have one single layer results in requiring its dimensionality to be exponentially large. For example, Gaussian kernel machines $f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i)$ need at least $n = O(2^d)$ terms to represent the parity function of a binary string x of d bits. In the language of neural networks, learning some functions require an exponential amount of hidden neurons when the network has only one hidden layer. In contrast, the parity function is learnable using a $O(d)$ dimensional representation with two layers (Bengio et al., 2015, Section 14.6). In sum, deep representations incorporate the compositional structure of the world as their prior knowledge. Such compositional structure, constructing features out of features, leads to exponential gains in representational power. Rephrasing the comparison in terms of sample complexity, functions with an exponential amount of different regions may require an exponential amount of data when learned using shallow representations, and a linear amount of data when learned using deep representations. Moreover, deep models are a generalization of shallow models, making them an object of both theoretical and practical interest. ◊

3.3.2 Convolutional neural networks

Kernel methods, random features and neural networks are general-purpose tools to construct representations from data. In particular, all of them are *permutation invariant*: they learn the same representation from two different versions of the same data, if the only difference between the two is the order of their variables. But for some data, the order of variables is rich prior knowledge, exploitable to build better representations.

For example, consider the design of a machine to classify the hand-written digits from Figure 3.6a into "fives" or "eights". As humans, solving this task is easy because of the way on which the pixels, edges, strokes, and parts of the digits are arranged on the two-dimensional surface of the paper. The task becomes much more difficult if we scramble the pixels of the digit images using a fixed random permutation, as illustrated in Figure 3.6b.

Although the transformation from Figure 3.6a to Figure 3.6b destroys the spatial dependencies between the variables under study, permutation invariant methods treat equivalently both versions of the data. Permutation invariant methods therefore would ignore the local spatial dependence structures between neighbouring pixels in natural images. To some extent, permutation invariant methods will search patterns over the space of all images, including images formed by random pixels, instead of focusing their efforts on the smaller set of images that feel natural to perception. Therefore, discarding spatial dependencies is a waste of our resources! How can we leverage these dependence structures, instead of ignoring them?

One way is to apply the same feature map along different local spatial groups of variables. In the case of images, this means extracting the same representation from different small neighbourhoods of pixels in the image, and returning the concatenation of all of these local representations as the image representation. After all, to locate an object in an image, all we care about is *what* features are present in the image, regardless of *where*. This is known as *translational invariance*.

Convolutional neural networks implement this idea by extending the architecture of feedforward neural networks. Deep convolutional neural networks alternate three different types of layers: convolutional layers, nonlinearity layers, and pooling layers. We now detail the inner workings of these three types of layers. For simplicity, assume that the data under study are color images. The mathematical representation of an image is the three-dimensional volume or *tensor* $X \in \mathbb{R}^{w \times h \times d}$, where w and h are the width and the height of the image in pixels, and d is the depth of the image in channels or features.

First, convolution layers accept three inputs: the input image $X \in \mathbb{R}^{w \times h \times d}$, the filter bank $W \in \mathbb{R}^{s \times s \times d \times d'}$ containing d' filters of size $s \times s \times d$, and the bias vector $b \in \mathbb{R}^{d'}$. Convolution layers return one output image $X' \in \mathbb{R}^{(w-s+1) \times (h-s+1) \times d'}$, with entries

$$X' = \text{conv}(X; W, b),$$
$$X'_{i',j',k'} = \sum_{i=1}^{s} \sum_{j=1}^{s} \sum_{k=1}^{d} X_{i'+i-1,j'+j-1,k} W_{i,j,k,k'} + b_{k'},$$

for all $i' \in \{1, \ldots, w'\}$, $j' \in \{1, \ldots, h'\}$, and $d' \in \{1, \ldots, d'\}$. In practice, the input images X are *padded* with zeros before each convolution, so that the input and output images have the same size. The intensity of the output pixel $X'_{i,j,k}$ relates to the presence of the filter $W_{:,:,:,k}$ near the input pixel $X_{i,j,:}$. Figure 3.5 exemplifies the convolution operation.

Second, nonlinearity layers $\sigma(\cdot)$ apply a nonlinear function entrywise

$$X' = \sigma(X),$$
$$X'_{i,j,k} = \sigma(X_{i,j,k}),$$

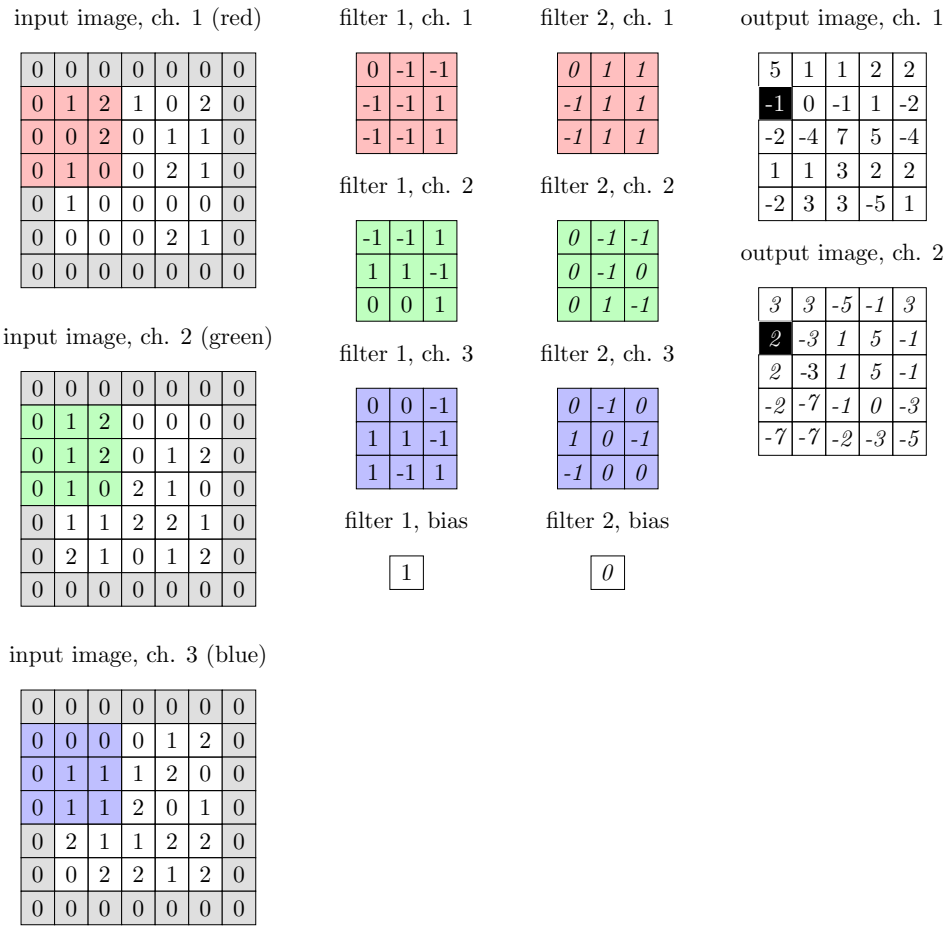


Figure 3.5: A convolution layer transforming a zero-padded input color image $X \in \mathbb{R}^{5\times5\times3}$ into an output image $X' \in \mathbb{R}^{5\times5\times2}$, using a filter bank $W \in \mathbb{R}^{3\times3\times3\times2}$ and its corresponding two biases $b = (b_1, b_2)$. Highlighted in black, the output pixel computed by the color-highlighted patches of the input image and filter bank. Figure adapted from (Karpathy, 2015).

(a) Original handwritten digit images.

(b) Handwritten digit images with randomly permuted pixels.

Figure 3.6: The MNIST handwritten digits dataset.

for all $i \in \{1, \ldots, w\}$, $j \in \{1, \ldots, h\}$, and $k \in \{1, \ldots, d\}$.

Third, pooling layers summarize each neighbourhood of $\alpha \times \alpha$ pixels in a given input image $X \in \mathbb{R}^{w \times h \times d}$ into one pixel of the output image $X' \in \mathbb{R}^{(w/\alpha) \times (h/\alpha) \times d}$. For instance, in *max pooling* each of the pixel values of the output image is the maximum value of the pixel values within each $\alpha \times \alpha$ neighbourhood in the output image. In most applications, $\alpha = 2$; in this case, simply write $X' = \text{pool}(X)$. Pooling layers reduce the computational requirements of deep convolutional neural networks, since they reduce the size of the input image passed to the next convolution. Pooling layers operate independently per channel. To remove the need of pooling layers, some authors suggest to implement convolution layers with *large stride*. In these large stride convolutions, the filter slides multiple pixels at a time, effectively reducing the size of the output image (Springenberg et al., 2014).

In short, the representation implemented by a deep convolutional neural network has form

$$\begin{aligned} \phi(X) &= \phi_L, \\ \phi_l &= \text{pool}(\sigma(\text{conv}(\phi_{l-1}; W_l, b_l))), \\ \phi_0 &= X, \end{aligned}$$

where L can be in the dozens (Bengio et al., 2015). The feature map of a convolutional deep neural network is "elastic", in the sense that it accepts images of arbitrary size. The only difference is that the convolution operation will slide over a larger input image, thus producing a larger output image. If we require a final representation of a fixed dimensionality, we can use the last pooling layer to downscale the dimensionality of the final output image appropriately.

Remark 3.6 *(Recurrent neural networks).* Some data, such as speech, video, and stock quotes, are naturally presented as a time series. The temporal dependence structure in these data is yet another instance of prior knowledge that can be conveniently exploited to build better representations. *Recurrent* neural networks (see, for example, (Sutskever, 2013)) are neural networks

adapted to learn from time series. ◊

3.3.3 Learning with neural networks

Neural networks, fully connected or convolutional, shallow or deep, are trained using the *backpropagation* algorithm (Rumelhart et al., 1986). Usually, before employing backpropagation, we fill each weight matrix $W_l \in \mathbb{R}^{d_{l-1} \times d_L}$ in the neural network with random numbers sampled from

$$\mathcal{U}\left[-\sqrt{\frac{6}{d_{l-1}+d_L}}, +\sqrt{\frac{6}{d_{l-1}+d_L}}\right],$$

where $\mathcal{U}$ denotes the uniform distribution (Glorot and Bengio, 2010).

Once the neural network has been randomly initialized, the backpropagation algorithm runs for a number of iterations. Each backpropagation iteration implements two computations. First, the raw data makes a *forward pass* through the network, from the input layer to the output layer, producing predictions. Second, the prediction errors make a *backwards pass* through the network, from the output layer to the input layer. In this backward pass, the backpropagation algorithm computes how should we modify each of the weights in the network to lower its average prediction error. Backpropagation proceeds recursively: the weight updates in one layer depend on the prediction errors made by the next layer. Thanks to the differentiation chain rule, backpropagation is effectively implemented as a gradient descent routine on neural networks with architectures described by directed acyclic graphs. The backpropagation algorithm updates the network for a number of iterations, until the average error over some held-out validation data stops decreasing or starts to increase. For a full description of the backpropagation algorithm and its history, refer to (Bengio et al., 2015, Section 6.4).

Bear in mind that, in opposition to kernels and random features, training neural networks requires approximating the solution to a high-dimensional nonconvex optimization problem. Nonconvex optimization problems have multiple local minima, so initializing the network to a different set of weights will result in backpropagation converging to a different solution, and this solution will have a different generalization error (Section 2.4). To alleviate this issue, practitioners train multiple neural networks on the same data, starting from different random initializations, and then average their outputs for a final prediction. The nonconvexity of deep neural networks is a double edged sword: it allows the learning of highly complex patterns, but hinders the development of theoretical guarantees regarding their generalization performance.

We now exemplify how to learn a single-hidden-layer neural network to perform nonlinear least-squares regression.

Example 3.5 *(Neural least-squares).* As in Example 3.2 the goal here is to minimize the least-squares regression error

$$E = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2, \tag{3.14}$$

with respect to the parameters $\{(\alpha_j, w_j, b_j)\}_{j=1}^m$, and β of the neural network

$$f(x) = \sum_{j=1}^{m} \alpha_j \sigma(\langle w_j,\, x\rangle + b_j) + \beta$$

for some nonlinearity function $\sigma : \mathbb{R} \to \mathbb{R}$. We use the backpropagation algorithm. First, propagate all the training data through the network. Then, compute the derivatives of the error function (3.14) with respect to each parameter of the network:

$$\begin{aligned}
\frac{\partial E}{\partial w_{j,k}} &= \frac{2}{n}\sum_{i=1}^{n}(f(x_i) - y_i)\cdot\alpha_j\cdot\sigma'(\langle w_j,\, x_i\rangle + b_j)\cdot x_{i,k}, \qquad (3.15)\\
\frac{\partial E}{\partial b_j} &= \frac{2}{n}\sum_{i=1}^{n}(f(x_i) - y_i)\cdot\alpha_j\cdot\sigma'(\langle w_j,\, x_i\rangle + b_j),\\
\frac{\partial E}{\partial \alpha_j} &= \frac{2}{n}\sum_{i=1}^{n}(f(x_i) - y_i)\cdot\sigma(\langle w_j,\, x_i\rangle + b_j),\\
\frac{\partial E}{\partial \beta} &= \frac{2}{n}\sum_{i=1}^{n}(f(x_i) - y_i).
\end{aligned}$$

We can observe the recursive character in (3.15): the updates of the weights in a given layer depend on the next layer. Similar, slightly more complicated formulas follow for deep and convolutional neural networks. Using the gradients (3.15), we update T times each parameter in the network using the update rule

$$w_{j,k} = w_{j,k} - \gamma\frac{\partial E}{\partial w_{j,k}},$$

where $\gamma \in (0, 1)$ is a small *step size* (Section 2.4). Similar update rules follow for $\{\alpha_j, b_j\}_{j=1}^m$ and b. To decide the number of gradient descent iterations T, we can monitor the performance of the neural network on some held-out validation set, and stop the optimization when the validation error error stops decreasing. The computation of the gradients of (3.14) takes $O(n)$ time, a prohibitive requirement for large n or large number of iterations T. Because of this reason, neural networks are commonly trained using stochastic gradient descent (Remark 2.1). ♢

The previous example illustrates how to tune the network parameters $\{(\alpha_j, w_j, b_j)\}$ and b, but it does not comment on how to choose the architectural aspects of the network, such as the nonlinearity function, the step size in the gradient descent optimization, the number of hidden layers, the number of neurons in each hidden layer, and so on. These parameters are usually tuned using cross-validation, as detailed in Section 2.3.2. The candidate set of neural network architectures is often chosen at random from some reasonable distribution over the architecture parameters (Bergstra and Bengio, 2012; Nishihara et al., 2016). Then, the final neural network is the best or the average of the top best performing on the validation set.

Because of the great flexibility of deep neural network representations, it is important to implement regularization schemes along with their optimization. Three popular alternatives are dropout regularization (Srivastava et al., 2014) batch normalization (Ioffe and Szegedy, 2015), and early stopping. Dropout regularization reduces the risk of overfitting by deactivating a random subset of the neurons at each iteration of gradient descent, so the network can not excessively rely on any single neuron. Batch normalization readjusts the parameters of the network periodically during learning, so that the neuron pre-nonlinearity activations have zero mean and unit variance. Early stopping stops the training of the neural network as soon as possible, since the generalization error of algorithms trained with stochastic gradient descent increases with the number of iterations (Hardt et al., 2015).

3.4 Ensembles

Ensembles are combinations of different predictors, or *weak learners*, to solve one single learning problem. Ensembling is a powerful technique: the winning entry of the $1,000,000 *Netflix Prize* was a combination of more than 100 different weak learners (Bell et al., 2008). There are two main ways of combining weak learners together: *boosting* and *stacking*.

First, boosting ensembles learn a sequence of weak learners, where each weak learner corrects the mistakes made by previous ones. Given some data $\{(x_i, y_i)\}_{i=1}^n$, gradient boosting machines (Friedman, 2001) perform regression as follows. First, compute the constant

$$f_0(x) = \gamma_0 = \arg\min_{\gamma} \sum_{i=1}^{n} \ell(\gamma, y_i),$$

where $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a differentiable loss function. Second, for a number of boosting iterations $1 \leq t \leq T$, use the *pseudo-residual* data

$$\left\{ \left(x_i, -\frac{\partial \ell(f_{t-1}(x_i), y_i)}{\partial f_{t-1}(x_i)} \right) \right\}_{i=1}^{n}$$

to fit a weak learner h_t, and incorporate it into the ensemble as

$$f_t(x) = f_{t-1}(x) + \gamma h_t(x),$$

where

$$\gamma = \arg\min_{\gamma} \sum_{i=1}^{n} \ell(f_{t-1}(x_i) + \gamma h_t(x_i), y_i).$$

The ensemble f_T is the final predictor.

Second, stacking ensembles construct T weak learners independently and in parallel, and their predictions are the input to another machine, that learns how to combine them into the final prediction of the ensemble. Bagging is one popular variation of stacking, where one trains each of the T independent weak learners on a subset of the data sampled at random with replacement. The predictions of a bagging ensemble are simply the average of all the weak learners. Bagging reduces the error variance of individual predictions. To see this, write the error variance of the ensemble as

$$\begin{aligned}\mathbb{E}\left[\left(\frac{1}{T}\sum_{i=1}^{n}\boldsymbol{\varepsilon}_i\right)^2\right] &= \frac{1}{T^2}\mathbb{E}\left[\sum_i\left(\boldsymbol{\varepsilon}_i^2 + \sum_{j\neq i}\boldsymbol{\varepsilon}_i\boldsymbol{\varepsilon}_j\right)\right] \\ &= \frac{1}{T}\mathbb{E}\left[\boldsymbol{\varepsilon}_i^2\right] + \frac{k-1}{k}\mathbb{E}[\boldsymbol{\varepsilon}_i\boldsymbol{\varepsilon}_j]\,.\end{aligned}$$

We see that if the weak learners are independent, the error covariances $\mathbb{E}[\boldsymbol{\varepsilon}_i\boldsymbol{\varepsilon}_j]$ tend to zero, so the ensemble will have an average error variance T times smaller than the individual weak learner error variances (Bengio et al., 2015).

Random forests are one popular example of bagging ensembles (Breiman, 2001), considered one of the most successful learning algorithms (Fernández-Delgado et al., 2014). Random forests are bags of decision trees, each of them trained on a random subset of both the data examples and the data features. Random forests induce a random representation, like the ones studied in Section 3.2. A random forest with m decision trees of l leafs each implements a lf-dimensional random feature map ϕ, with features

$$\phi(x)_j = \mathbb{I}\left(\text{leaf}\left(\left\lfloor\frac{j-1}{m}+1\right\rfloor, x\right) = (\text{mod}(j-1,m)+1)\right), \tag{3.16}$$

where $\text{leaf}(t,x)$ returns the leaf index from the t-th tree where the sample x fell, for all $1 \leq j \leq lf$.

3.5 Trade-offs in representing data

Finding good representations is both the most important and challenging part of pattern recognition. It is important, because they allow to extract

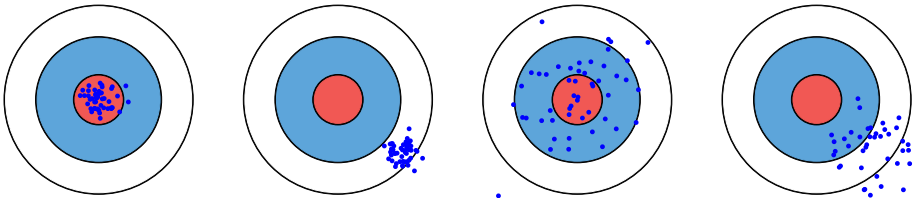


Figure 3.7: Illustration of bias and variance, when playing darts. From left to right, low bias and low variance, high bias and low variance, low bias and high variance, high bias and high variance.

nontrivial intelligence from data. And it is challenging, because it involves multiple intertwined trade-offs. The only way of favouring one representation over another is the use of prior knowledge about the specific data under study. Every representation learning algorithm excelling at one task will fail miserably when applied to others. As a matter of fact, when averaged over all possible pattern recognition tasks, no method is better than other. In mathematical jargon, *there is no free lunch* (Wolpert and Macready, 1997).

The first major trade-off is the one between the flexibility of a representation and its *sample complexity*. Learning flexible patterns calls for flexible feature maps, and flexible feature maps contain a large amount of tunable parameters. In turn, a larger amount of data is necessary to tune a larger amount of parameters. For instance, consider representing d dimensional data using a feature map with $O(md)$ free parameters. In the simplest case, where each of the parameters is binary can only take two different values, we face a search amongst 2^{md} possible representations. As a modest example, if learning from data containing $d = 10$ dimensions, there is an exponential amount

$$2^{d \times m} = 2^{100} \approx 1.25 \times 10^{30}$$

of single-hidden-layer neural networks with $m = 10$ hidden neurons connected by binary weights. Bellman (1956) termed this exponential rate of growth in the size of optimization problems *the curse of dimensionality*.

Second, flexible feature maps call for nonconvex numerical optimization problems, populated by local minima and saddle point solutions (recall Figure 2.2). But flexibility also contradicts invariance. For example, if learning to classify handwritten digit images like the ones depicted in Figure 3.6a, we may favour representations that are invariant with respect to slight rotations of the digits, given that the same digit can appear in the data at different angles, when written by different people. However, representations taking this invariance to an extreme would deem "sixes" indistinguishable from "nines", and perform poorly.

Third, from a statistical point of view, flexibility controls the *bias-variance trade-off* discussed in Sections 2.2.3 and 2.3.2. The trade-off originates from

the fact that learning drinks from two simultaneous, competing sources of error. First, the bias, which is the error derived from erroneous assumptions built in our representation. For example, linear feature maps exhibit high bias when trying to unveil a complex nonlinear pattern. High bias results in over-simplifying the pattern of interest, that is, underfitting. Second, the variance, which is the error derived from the sensitivity to noise in the training set. A learning algorithm has large variance when small changes in the training data produce large deviations on its predictions. High variance causes overfitting, which is the undesirable effect of hallucinating patterns from the noise polluting the data. In short, too-simple models have high bias and low variance, while too-complex models have low bias and high variance. Figure 3.7 illustrates the bias-variance trade-off when playing to hit the bullseye in the game of darts[1]. Good representations should aim at optimally balancing bias and variance to maximize performance at subsequent learning tasks.

3.6 Representing uncertainty

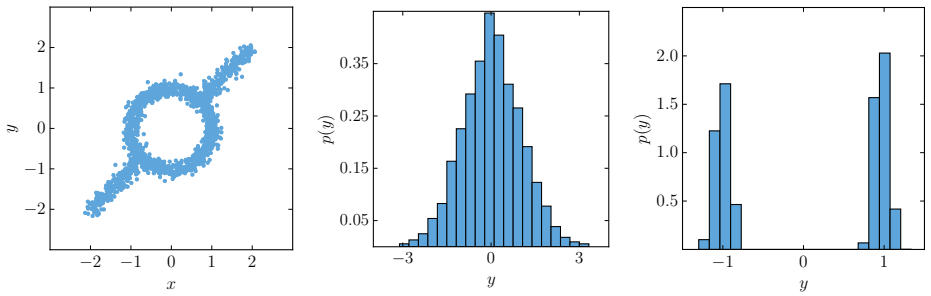
Uncertainty is ubiquitous in data. It arises due to human or mechanical errors in data collection, incomplete models, or fluctuations of unmeasured or missing variables. Even if we have the most Laplacian deterministic view of the universe, our limited knowledge and perception turns deterministic systems into partially random. Furthermore, describing complex processes using a few uncertain rules is simpler than describing them using a large amount of deterministic rules.

We can accommodate uncertainty in learning by assuming that predictions are not deterministic quantities $\hat{f}(x)$, but *predictive distributions* $\hat{P}(\boldsymbol{y} \mid x)$. For instance, consider access to some data $\{(x_i, y_i)\}_{i=1}^n$, where $y_i = f(x_i) + \epsilon_i$ for some function $f : \mathbb{R}^d \to \mathbb{R}$ that we wish to learn, and some additive noise $\epsilon_i \sim \mathcal{N}(0, \lambda^2)$, for all $1 \leq i \leq n$. Before seeing the measurements y_i, we can use our prior knowledge about the data under study, and define a *prior distribution* over the kind of functions f that we expect to see linking the random variables $\boldsymbol{x}$ and $\boldsymbol{y}$. For instance, we may believe that the possible regression functions f follow a Gaussian process (Rasmussen and Williams, 2006) prior:

$$f \sim \mathcal{N}(0, K),$$

where the $n \times n$ covariance matrix is the kernel matrix K, with entries $K_{ij} = k(x_i, x_j)$. Here, the kernel function k describes the shape of the interactions between pairs of points (x_i, x_j), for all $1 \leq i, j \leq n$, and depends on prior knowledge, but not on the data. Given a new observation x, the

[1] Figure based on `http://scott.fortmann-roe.com/docs/BiasVariance.html`



(a) Data (b) Gaussian process (c) Random forest

Figure 3.8: Measuring uncertainty with predictive distributions.

$n+1$ measurement locations $(x_1, \ldots, x_n, x)$ are still jointly Gaussian:

$$\begin{pmatrix} y \\ f(x) \end{pmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} K + \lambda^2 I_n & k_x \\ k_x^\top & k(x,x) \end{pmatrix}\right), \tag{3.17}$$

where the column vector $k_x \in \mathbb{R}^n$ has entries $k_{x,i} = k(x, x_i)$ for all $1 \leq i \leq n$.

Now, let us take into account the measurements $\{y_i\}_{i=1}^n$. By applying the conditional distribution rule of multivariate Gaussians (4.1), we can transform the Gaussian process prior (3.17) into the Gaussian process *posterior* or predictive distribution

$$\begin{aligned} f(x) &\sim \mathcal{N}(\mu(x), \sigma(x)) \\ \mu(x) &= k_x^\top (K + \lambda^2 I_n)^{-1} y, \\ \sigma(x) &= k(x,x) - k_x^\top (K + \lambda^2 I_n)^{-1} k_x. \end{aligned} \tag{3.18}$$

As seen in Equation (3.18), the predictions from Gaussian processes are Gaussian distributions. In some situations, however, predictive distributions can be far from Gaussian: heavy-tailed, multimodal, and so forth. One method to approximate arbitrary predictive distributions is the *bootstrap method* (Efron, 1979). The bootstrap method trains K weak learners that solve the learning problem at hand, each of them on a different *bootstrap set* $\{x_{k(i)}\}_{i=1}^m$, for all $1 \leq k \leq K$. Each bootstrap set is a random subset of m examples of the data sampled with replacement (Kleiner et al., 2014). At test time, the bootstrap method returns K different answers, one per weak learner. The ensemble then summarizes the K bootstrap answers into a predictive distribution. Random forests (Section 3.4) are one simple form of bootstrapping. The predictions provided by random forests are a collection of predictions made by the individual decision trees forming the forest. Thus, one can use these individual predictions to estimate a predictive distribution.

Figure 3.8 illustrates the predictive distributions $P(\boldsymbol{y} \,|\, \boldsymbol{x} = 0)$ estimated by a Gaussian process and a random forest, using the data from Figure 3.8a.

On the one hand, the Gaussian process returns a Gaussian predictive distribution, depicted in Figure 3.8b, which erroneously characterizes the true, bimodal predictive distribution at $x = 0$. On the other hand, the random forest is able to determine, as seen in Figure 3.8c, that the true predictive distribution at $x = 0$ has two pronounced modes. In any case, the Gaussian process correctly captures the variance (uncertainty) of the true predictive distribution. And this is everything we could hope for, since Gaussian process predictive distributions are Gaussian, and therefore unimodal.

Part II

Dependence

Chapter 4

Generative dependence

This chapter contains novel material. First, Section 4.4.5 introduces the use of expectation propagation and sparse Gaussian processes to model multivariate conditional dependence in copulas (Lopez-Paz et al., 2013b). We illustrate the effectiveness of our approach in the task of modeling regular vines (Section 4.6.1). We call this model the Gaussian Process Regular Vine (GPRV). Second, Section 4.4.6 proposes a nonparametric copula model, along with its associated conditional distributions (Lopez-Paz et al., 2012). We exemplify the effectiveness of our approach in the task of semisupervised domain adaptation using regular vines (Section 4.6.2). We call this model the Non-Parametric Regular Vine (NPRV).

Generative models use samples

$$x = \{x_1, \ldots, x_n\} \sim P^n(\boldsymbol{x}),\ x_i \in \mathbb{R}^d$$

to *estimate* the probability density function

$$p(\boldsymbol{x}) = \frac{\partial^d P(\boldsymbol{x})}{\partial x_1 \cdots \partial x_d},$$

where $P(\boldsymbol{x})$ is a continuously differentiable cdf. So, generative models aim at describing all the marginal distributions and dependence structures governing the multivariate data x by estimating its density function $p(\boldsymbol{x})$. This task of *density estimation problem* is often posed as a *maximum likelihood estimation*[1], and solved in two steps. First, choose one *generative model*, that is, a collection of density functions $\mathcal{P}_\Theta = \{p_\theta\}_{\theta \in \Theta}$ indexed by their parameter vector $\theta \in \Theta$. Second, choose the density $p_{\hat{\theta}} \in \mathcal{P}_\Theta$ that best describes the samples x, by maximizing the log-likelihood objective

$$L(\theta) = \sum_{i=1}^{n} \log p_\theta(x_i),$$

[1] We call *estimation* the process of obtaining point-estimates of parameters from observations. We call *inference* the process of deriving posterior distributions from previous beliefs and observations.

with respect to the distribution parameters θ. Let $\hat{\theta}$ be the parameter vector maximizing the previous objective on the data x. Then, the maximum likelihood solution to the density estimation problem is the density $p_{\hat{\theta}}$.

Why is generative modeling of interest? A good estimate for the data generating density function $p(\boldsymbol{x})$ allows all sorts of complex manipulations, including:

1. *Evaluating the probability of data.* This allows to detect outliers, or to manipulate samples as to increase or decrease their likelihood with respect to the model.

2. *Sampling new data.* Generating new samples is useful to synthesize artificial data, such as images and sounds.

3. *Computing conditional distributions* of output variables $\boldsymbol{x}_{\mathcal{O}}$ given input variables $\boldsymbol{x}_{\mathcal{I}} = x$. The conditional distribution $p(\boldsymbol{x}_{\mathcal{O}} \mid \boldsymbol{x}_{\mathcal{I}} = x_{\mathcal{I}})$ could characterize, for instance, the distribution of missing variables: their expected value, variance (uncertainty), and so on. Conditional distributions also allow to use generative models for discriminative tasks, like regression and classification.

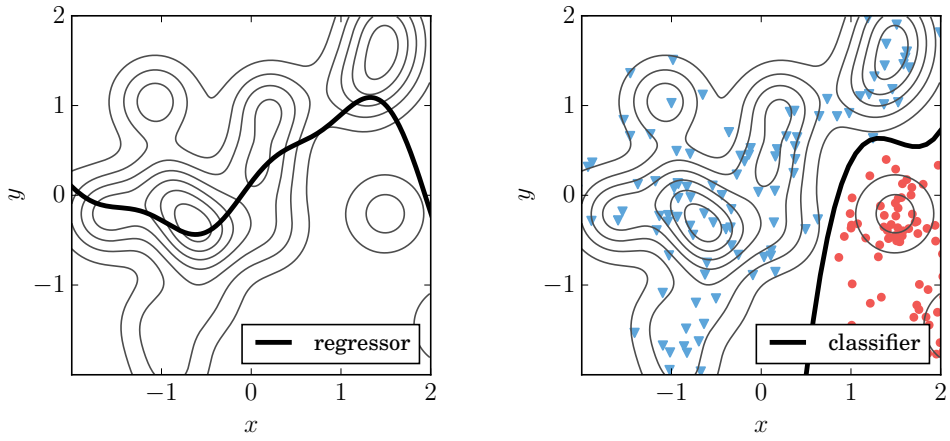
4. *Computing marginal distributions* of variables $\boldsymbol{x}_{\mathcal{M}}$, by integrating out (or marginalizing out) all the variables in $\bar{\mathcal{M}}$:

$$p(\boldsymbol{x}_{\mathcal{M}}) = \int_{\bar{\mathcal{M}}} p(\boldsymbol{x}_{\mathcal{M}}, \boldsymbol{x}_{\bar{\mathcal{M}}} = x_{\bar{\mathcal{M}}}) \mathrm{d}x_{\bar{\mathcal{M}}}.$$

Under mild conditions, the probability density function $p(\boldsymbol{x})$ contains all the observable information about the data generating distribution $P(\boldsymbol{x})$. Thus, accurately estimating the density function of our data amounts to solving multiple statistical learning problems at once, including regression, classification, and so forth. This erects density estimation as the silver bullet to all statistical learning problems. But, with great powers comes great responsibility: density estimation, the most general of statistical problems, is also a most challenging task. To better understand this, take a look at Figure 4.1. In both regression and classification tasks on the depicted density, the statistic of interest is shown as a black line. Either the depicted regressor or the depicted classifier is a much simpler object than the full density of the data. Thus, for problems such as regression or classification, density estimation is often a capricious intermediate step. In these situations, density estimation is a living antagonist of Vapnik's principle:

When solving a problem of interest, do not solve a more general problem as an intermediate step. (Vapnik, 1998)

A second challenge of density estimation is its computational intractability. This difficulty arises because probabilities require normalization, and such



(a) regression (b) classification

Figure 4.1: Density estimation is more difficult than (a) regression, and (b) classification.

normalization involves solving challenging and multidimensional integrals over the density function. In fact, different density computations pose different trade-offs; for instance, generative models allowing for easy sampling may be difficult to condition and marginalize, or vice versa (Goodfellow et al., 2014, Table 2). Luckily, normalized probabilities are necessary only when combining different generative models together: for example, when evaluating the likelihood of a sample with respect to two different generative models.

A third challenge of generative modeling is their evaluation: generative models trained for different purposes should be evaluated differently. Theis et al. (2015) illustrates this dilemma for generative models of natural images. Using a fixed dataset, the authors construct a generative model with high log-likelihood but producing poor samples, and a generative model with low log-likelihood but producing great samples. The latter model simply memorizes the training data. This memorization allows to produce perfect samples (the training data itself), but assigns almost zero log-likelihood to unseen test data. More formally, when we approximating a density function p with a model p_θ using a metric d over probability measures, there are multiple ways to be $d(p, p_\theta) = \varepsilon > 0$ wrong. Unsurprisingly, some of these ways to be wrong are more appropriate to solve some problems (like log-likelihood maximization for data compression), and less appropriate for others (for instance, a higher degree of memorization leads to better sample quality). This relates to the notion of loss functions in supervised learning, since different losses aim at different goals.

This chapter explores five models for density estimation: Gaussian models, transformation models, mixture models, copula models, and product models. Each model has different advantages and disadvantages, and excels

at modeling different types of data.

Remark 4.1 *(Wonders and worries in maximum likelihood estimation).* Maximum likelihood relies on two assumptions: the *likelihood principle* and the *law of likelihood*. The likelihood principle assumes that, given a generative model like $\mathcal{P}_\Theta$, the log-likelihood function $L(\theta)$ contains all the relevant information to estimate the parameter $\theta \in \Theta$. On the other hand, the law of likelihood states that the ratio $p_{\theta_1}(x)/p_{\theta_2}(x)$ equals the amount of evidence supporting the model p_{θ_1} in favour of the model p_{θ_2}, given the data x.

Maximum likelihood estimation is *consistent*: the sequence of maximum likelihood estimates converges to the true value under estimation, as the sample size grows to infinity. Maximum likelihood is also *efficient*: no other consistent estimator has lower asymptotic mean squared error. Technically, this is because maximum likelihood estimation achieves the absolute Cramér-Rao bound.

When working with finite samples, there are alternative estimators that outperform maximum likelihood estimation in mean squared error. A notable example is the James-Stein estimator of the mean of a d-dimensional Gaussian, for $d \geq 3$. We exemplify it next. Consider observing one sample $y \sim \mathcal{N}(\mu, \sigma^2 I)$; then, the maximum likelihood estimation of the mean is $\hat{\mu}_{\text{MLE}} = y$, which is an estimation taking into account each of the d coordinates separately. In contrast, the James-Stein estimator is $\hat{\mu}_{\text{JS}} = (1 - (m-2)\sigma^2 \|y\|^{-2})y$, which is an estimation taking into account the norm of y to estimate each of the d coordinates jointly. ◊

4.1 Gaussian models

The Gaussian distribution is the most important of probability distributions because of two reasons. First, due to the *central limit theorem*, the sum of n independent random variables converges to an unnormalized Gaussian distribution, as $n \to \infty$. Second, Gaussian distributions model linear dependencies, and this enables a linear algebra over Gaussian distributions convenient for computation.

The Gaussian distribution is a distribution over the real line, with density function

$$\mathcal{N}(\boldsymbol{x}; \mu, \sigma^2) = p(\boldsymbol{x} = x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{(2\pi)}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

fully parametrized by its first two moments: the mean μ and the variance σ^2. The special case $\mathcal{N}(\boldsymbol{x}|0, 1)$ is the *Normal distribution*. The Gaussian cumulative distribution function does not have a closed form, but is approximated numerically. Figure 4.2 plots the probability density function, cumulative

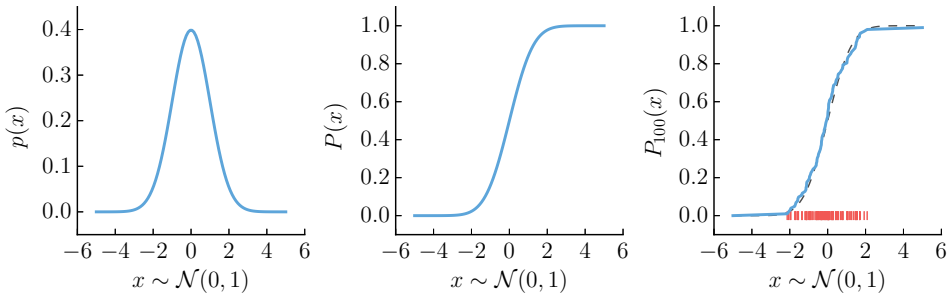


Figure 4.2: Probability density function (pdf), probability cumulative function (cdf), and empirical cdf based on 100 samples (depicted in red) for a Normal distribution.

distribution function, and empirical cumulative distribution function (see Definition (2.3)) of a Normal distribution.

Assume now d Gaussian random variables $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$ with $\boldsymbol{x}_i \equiv \mathcal{N}(\mu_i, \Sigma_{ii})$. If the dependencies between the components in x are linear, the joint distribution of $\boldsymbol{x}$ is a *multivariate Gaussian distribution*, with a density function

$$\mathcal{N}(\boldsymbol{x}; \mu, \Sigma) = p(\boldsymbol{x}; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$$

fully characterized by its mean vector $\mu = (\mu_1, \ldots, \mu_d) \in \mathbb{R}^d$, and the positive-definite covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$. The d diagonal terms $\Sigma_{i,i}$ are the d variances of each of the Gaussian random variables $\boldsymbol{x}_i$ forming the random vector $\boldsymbol{x}$, and each off-diagonal term Σ_{ij} $(i \neq j)$ is the covariance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Therefore, uncorrelated Gaussian random variables have diagonal covariance matrices. In particular, for any $\sigma^2 > 0$, we call the distribution $\mathcal{N}(\mu, \sigma^2 I_d)$ *isotropic or spheric*, see the left side of Figure 4.3. Two Gaussian random variables may not be jointly Gaussian; in this case, their dependencies are nonlinear. One important consequence of this fact is that two random variables can be simultaneously uncorrelated and dependent. So remember: independent implies uncorrelated, but uncorrelated does not imply independent!

The Gaussian distribution is a member of the elliptical distributions. These are the distributions with contours of regions of equal density described by ellipses. The center of the ellipse is the vector μ, the sizes of its semiaxis are the diagonal elements from Λ, and the rotation of the ellipse with respect to the coordinate system of the Euclidean space is U, where $\Sigma = U\Lambda U^\top$.

The linear dependencies described by Gaussian distributions reduce their modeling capabilities, but bring computational advantages. First, affine

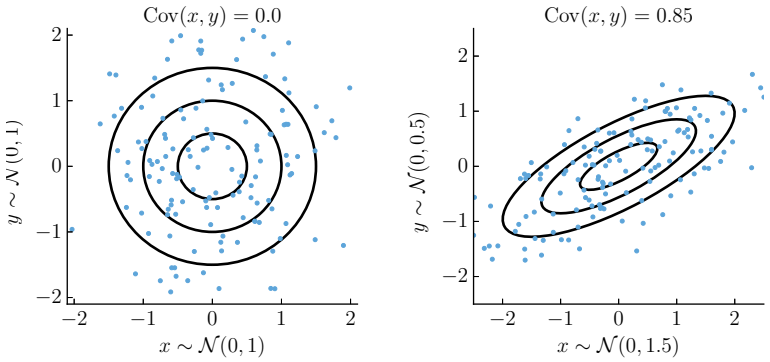


Figure 4.3: Samples drawn from two different Gaussian distributions, along with lines indicating one, two, and three standard deviations.

transformations of Gaussian random vectors are also Gaussian; in particular,

$$\left.\begin{aligned} \boldsymbol{x} &\equiv \mathcal{N}(\mu, \Sigma) \\ \boldsymbol{y} &\leftarrow A\boldsymbol{x} + b \end{aligned}\right\} \Rightarrow \boldsymbol{y} \equiv \mathcal{N}(b + A\mu, A\Sigma A^\top). \tag{4.1}$$

One consequence of the multivariate Gaussian affine transform is that any marginal distribution of a Gaussian distribution is also Gaussian. For example, to compute the marginal distribution of $(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_4)$, set $b = 0$ and use

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix},$$

that is, dropping the irrelevant terms from μ and the irrelevant rows and columns from Σ. The multivariate Gaussian affine transform also implies that sums of Gaussian random variables are also Gaussian. Finally, if $\boldsymbol{x} \equiv \mathcal{N}(\mu, \Sigma)$ with

$$\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix},$$

such that $\mu_1 \in \mathbb{R}^p$, $\mu_2 \in \mathbb{R}^q$, and Σ has the appropriate block structure, then

$$p(\boldsymbol{x}_1 | \boldsymbol{x}_2 = a) = \mathcal{N}\left(\mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(a - \mu_2), \Sigma_{11} - \Sigma_{22}^{-1}\Sigma_{21}\right)$$

Finally, two important information-theoretic quantities have closed form formulae for Gaussian distributions. These are the entropy of a Gaussian

$$H(\mathcal{N}(\mu, \Sigma)) = \frac{1}{2} \ln\left((2\pi e)^d \cdot |\Sigma|\right),$$

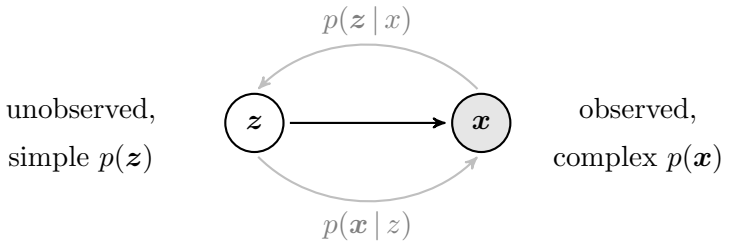


Figure 4.4: The canonical transformation model.

and the Kullback-Liebler divergence between two Gaussians

$$D_{\mathrm{KL}}(\mathcal{N}(\mu_0, \Sigma_0) \| \mathcal{N}(\mu_1, \Sigma_1)) = \\ \frac{1}{2}\left(\mathrm{tr}\left(\Sigma_1^{-1}\Sigma_0\right) + (\mu_1 - \mu_0)^\top \Sigma_1^{-1}(\mu_1 - \mu_0) - d + \ln \frac{|\Sigma_1|}{|\Sigma_0|}\right).$$

For additional identities involving the multiplication, division, integration, convolution, Fourier transforms, and constrained maximization of Gaussians, consult (Roweis, 1999).

4.2 Transformation models

Transformation (or latent variable) models assume that the random variable under study $\boldsymbol{x}$ is explained by some simpler latent random variable $\boldsymbol{z}$. While the distribution of the observed variable $\boldsymbol{x}$ may be in general complex and high dimensional, it is common to assume that the distribution of the latent explanatory factors $\boldsymbol{z}$ is low-dimensional and easy to model. Figure 4.4 illustrates the canonical transformation generative model.

Let us make this definition concrete with one simple example. Consider that we are designing a generative model of 100×100 pixel images of handwritten digit images. Instead of directly modeling the dependence structure of the $10,000$ pixels forming $\boldsymbol{x}$, we could consider instead high level descriptions $\boldsymbol{z}$ like "a *thick*, *quite-round* number *six*, which is *slightly-rotated-to-the-left*". In such descriptions, italic words describe the values of the latent explanatory factors "digit thickness", "digit roundness", "digit class", and "digit rotation", which incarnate the intensities of the observed pixels. Using the latent variable $\boldsymbol{z}$, modeling the distribution of $\boldsymbol{x}$ translates into modeling i) the distribution of $\boldsymbol{z}$, and ii) the function f mapping $\boldsymbol{z}$ to $\boldsymbol{x}$. Therefore, transformation models are useful to model high-dimensional data when this is described as a function of a small amount of explanatory factors, and when these explanatory factors follow a distribution that is easy to model (for instance, when the explanatory factors are mutually independent).

In the following, we will use the notation $X \in \mathbb{R}^{n \times d}$ to denote the data matrix constructed by stacking the samples $x_1, \ldots, x_n \sim P(\boldsymbol{x})$ as rows.

4.2.1 Gaussianization

Let us start with one of the simplest transformation models. *Gaussianization* (Chen and Gopinath, 2001) computes an invertible transformation from the input feature matrix $X^{(0)} \in \mathbb{R}^{n \times d}$, which follows a continuous distribution with strictly positive density, into the output explanatory factor matrix $Z = X^{(T)} \in \mathbb{R}^{n \times d}$, which approximately follows a Normal density function. Gaussianization computes this transformation by iterating two computations. First, it employs the ecdf (Definition 2.3) and the inverse cdf of the Normal distribution to make each column of $X^{(t)}$ follow a Normal distribution. Let $M^{(t)} \in \mathbb{R}^{n \times d}$ be the matrix containing the result of these d one-dimensional transformations. Second, Gaussianization transforms $M^{(t)}$ into $X^{(t+1)} \in \mathbb{R}^{n \times d}$ by applying a simple transformation. When this transformation is a random rotation, the principal component analysis rotation, or the independent component analysis rotation, the sample $X^{(t+1)}$ follows a distribution closer to the Normal distribution than the previous iterate $X^{(t)}$ (Laparra et al., 2011).

Denote by f the Gaussianization transformation after a sufficiently large number of iterations, and observe that this function is invertible. Then, the data $f(X^{(0)})$ approximately follows a d-dimensional Normal distribution. Using f, we can obtain a new sample from $P(\boldsymbol{x})$ by sampling $z \sim \mathcal{N}(0, I_d)$ and returning $x = f^{-1}(z)$. We can also approximate likelihood $p(\boldsymbol{x} = x)$ by evaluating the Normal likelihood $\mathcal{N}(f(x); 0, I_d)$ and renormalizing with Equation 2.2. On the negative side, obtaining the conditional and marginal distributions of $p(\boldsymbol{x})$ using the Gaussianization framework is nontrivial, and the necessary number of iterations to obtain Gaussianity is often large. Moreover, Gaussianization models obtained from T iterations require storing $O(Tnd + Td^2)$ parameters.

4.2.2 Variational inference

One central computation in Bayesian statistics is posterior inference, implemented by applying Bayes' rule on the observed variables $\boldsymbol{x}$ and the latent variables $\boldsymbol{z}$. That is, to compute quantities

$$p_\theta(z \,|\, x) = \frac{p_\theta(x \,|\, z) p_\theta(z)}{p_\theta(x)}.$$

Commonly, the statistician decides the shape of the likelihood $p_\theta(\boldsymbol{x} \,|\, z)$ and prior $p_\theta(\boldsymbol{z})$ distributions. However, the marginal likelihood or data distribution $p(\boldsymbol{x})$ is often unknown, turning the inference of the posterior $p(\boldsymbol{z} \,|\, x)$ intractable. One way to circumvent this issue (Jordan, 1998) is to

introduce an approximate or *variational* posterior distribution $q_\phi(\boldsymbol{z} \mid x)$, and analyze its Kullback-Liebler divergence (Equation 2.3) to the true posterior $p(\boldsymbol{z} \mid x)$ using samples:

$$\begin{aligned}\mathrm{KL}(q_\phi(z \mid x) \| p_\theta(z \mid x)) &= \mathbb{E}_q\left[\log \frac{q_\phi(z \mid x)}{p_\theta(z \mid x)}\right] \\ &= \mathbb{E}_q[\log q_\phi(z \mid x)] - \mathbb{E}_q[\log p_\theta(z \mid x)] \\ &= \mathbb{E}_q[\log q_\phi(z \mid x)] - \mathbb{E}_q[\log p_\theta(z, x)] + \log p_\theta(x).\end{aligned}$$

The previous manipulation implies that

$$\log p_\theta(x) = \mathrm{KL}(q_\phi(z \mid x) \| p_\theta(z \mid x)) + \underbrace{(\mathbb{E}_q[\log q_\phi(z \mid x)] - \mathbb{E}_q[\log p_\theta(z, x)])}_{\mathcal{L}(\phi,\theta) := \mathrm{ELBO}}.$$

Since p does not depend on q, maximizing the ELBO (Evidence Lower BOund) results in minimizing the Kullback-Liebler divergence between the variational posterior $q_\phi(\boldsymbol{z} \mid x)$ and the target posterior $p_\theta(\boldsymbol{z} \mid x)$. So, if our variational posterior is rich enough, we hope that maximizing the ELBO will result in a good approximation to the true posterior. The ELBO can be rewritten as

$$\mathcal{L}(\phi, \theta) = -\mathrm{KL}(q_\phi(z \mid x) \| p_\theta(z)) + \mathbb{E}_q[\log p_\theta(x \mid z)], \tag{4.2}$$

an expression in terms of known terms $p_\theta(z)$ and $p_\theta(x \mid z)$, and $q_\phi(z \mid x)$.

At this point, we can use gradient descent optimization on (4.2) to learn both the variational parameters ϕ and the generative parameters θ. Unfortunately, the expectations in (4.2) are in general intractable, so one approximates them by sampling. Since such sampling depends on the variables that we are optimizing, the stochastic gradients $\nabla_\phi \mathcal{L}$ have large variance. To alleviate this issue, we can use the *reparametrization trick*

$$\mathbb{E}_{q_\phi(\boldsymbol{z} \mid x)}[f(\boldsymbol{z})] = \mathbb{E}_{p(\boldsymbol{\epsilon})}[f(g_\phi(\boldsymbol{\epsilon}, x))], \tag{4.3}$$

where g_ϕ is a deterministic function depending on the variational parameters ϕ (Kingma and Welling, 2013). Observe that the right hand side of (4.3) is an expectation with respect to a distribution $p(\boldsymbol{\epsilon})$ that no longer depends on the shape of the variational posterior. For example, Gaussian variational posteriors $\mathcal{N}(\boldsymbol{z} \mid \mu, \Sigma)$ can be reparametrized into Normal posteriors $\mathcal{N}(\boldsymbol{\epsilon} \mid 0, I_d)$ and deterministic functions $g_\phi = \Sigma^\top \boldsymbol{\epsilon} + \mu$. As a result, the reparametrization trick reduces the variance of the stochastic gradients $\nabla_\phi \mathcal{L}$.

In practice (Kingma and Welling, 2013; Kingma et al., 2014; Rezende et al., 2014), it is common to set $p(\boldsymbol{z}) = \mathcal{N}(0, I_d)$ and parametrize both $p(\boldsymbol{x} \mid z)$ and $q(\boldsymbol{z} \mid x)$ using deep neural networks (Section 3.3).

4.2.3 Adversarial networks

The Generative Adversarial Network (GAN) framework (Goodfellow et al., 2014) is a game between two players: a *generator* G_θ and a *discriminator* D_ϕ. In this game, the generator G_θ aims at generating samples $x = G_\theta(z)$ that look as if they were drawn from the data generating distribution $p(\boldsymbol{x})$, where z is drawn from some simple noise distribution $q(\boldsymbol{z})$. On the other hand, the responsibility of the discriminator D_ϕ is to tell if a sample was drawn from the data generating distribution $p(\boldsymbol{x})$ or if it was synthesized by the generator G_θ. Mathematically, playing this game is solving the optimization problem

$$\min_\theta \max_\phi \mathbb{E}[\log D(\boldsymbol{x})] + \mathbb{E}[\log(1 - D(G(\boldsymbol{z})))]\,,$$

with respect to the generator parameters θ and the discriminator parameters ϕ. In (Goodfellow et al., 2014), both the generator and the discriminator are deep neural networks (Section 3.3). On the negative side, the GAN framework does not provide an explicit mechanism to evaluate the probability density function of the obtained generator.

4.3 Mixture models

Let $p_{\theta_1}, \ldots, p_{\theta_k}$ be a collection of density functions, and let $\pi_1, \ldots, \pi_k \in [0, 1]$ be a collection of numbers summing to one. Then, the function

$$p(\boldsymbol{x}) = \sum_{i=1}^{k} \pi_i p_{\theta_i}(\boldsymbol{x})$$

is also a density function, called a *mixture model*. At a high level, mixtures implement the "`OR`" operation of the mixed densities, also called *mixture components*.

Sampling from mixture models is as easy as sampling from each the mixture components: just sample from the i-th mixture component, where the index $i \sim \text{Multinomial}(\pi_1, \ldots, \pi_k)$. Similarly, because of the linearity of integration, computing the marginal distributions of a mixture model is as easy as computing the marginal distributions of each the mixture components. At the same time, since marginalization is feasible, conditional mixture distributions are easy to compute.

Given data $\{x_1, \ldots, x_n\}$ and number of mixture components k, mixture models are parametric if $k < n$, and are nonparametric if $k = n$. Next, we briefly review how to estimate both parametric and nonparametric mixture models.

4.3.1 Parametric mixture models

Expectation Maximization or EM (Dempster et al., 1977) is often the tool of choice to train parametric mixture models.

Consider the task of modeling the data $\{x_1, \ldots, x_n\}$ using a mixture of k components, parametrized by the parameter vector θ. To this end, introduce a set of n latent variables $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n\}$; for all $1 \leq i \leq n$, the latent variable $\boldsymbol{z}_i \in \{1, \ldots, k\}$, and indicates from which of the k mixture components the example x_i was drawn.

Expectation maximization runs for a number of iterations $1 \leq t \leq T$, and executes two steps at each iteration. First, the *expectation step* computes the function

$$Q(\theta \mid \theta^{(t)}) = \mathbb{E}_{\boldsymbol{z} \mid x, \theta^{(t)}} \left[\sum_{i=1}^{n} \log p(x_i, \boldsymbol{z}_i \mid \theta^{(t)}) \right].$$

Second, the maximization step updates the parameter vector as

$$\theta^{(t+1)} = \arg\max_{\theta} Q(\theta \mid \theta^{(t)}).$$

If we treat the parameter θ as yet another latent variable, the EM algorithm relates to variational inference (Section 4.2.2). For general mixtures, the EM algorithm approximates the solution to a nonconvex optimization problem, and guarantees that the likelihood of the mixture increases per iteration.

4.3.2 Nonparametric mixture models

Nonparametric mixture models, also known as *Parzen-window estimators* or *kernel density estimators* (Parzen, 1962), dedicate one mixture component per example comprising our d-dimensional data, and have form

$$p(\boldsymbol{x}) = \frac{1}{nh^d} \sum_{i=1}^{k} K\left(\frac{\boldsymbol{x} - x_i}{h} \right),$$

where K is a nonnegative function with mean zero that integrates to one, and $h > 0$ is a bandwidth parameter proportional to the smoothness of the mixture. While nonparametric mixture models avoid the need of EM, they do require n mixture components, $O(nd)$ memory requirements. Due to the curse of dimensionality, modeling a high-dimensional space in a nonparametric manner requires an exponential amount of data; thus, nonparametric mixture models tend to overfit in moderate to high dimensions (Wasserman, 2010).

4.3.3 Gaussian mixture models

Gaussian Mixture Models (GMMs) are mixture models where each of the mixture components is Gaussian with known mean and covariance:

$$p(\boldsymbol{x}) = \sum_{i=1}^{k} \pi_i \mathcal{N}(\boldsymbol{x}; \mu_i, \Sigma_i).$$

One example of a Gaussian mixture model with 10 components is the one plotted with contours Figure 4.1 (a) and (b). GMMs are popular generative models because, when given enough components, they are universal probability density function estimates (Plataniotis, 2000). Gaussian Parzen-window estimators have form

$$p(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{(h\sqrt{2\pi})^d} \exp\left(-\frac{1}{2} \frac{\|\boldsymbol{x} - x_i\|^2}{h^2}\right).$$

Remark 4.2 *(Transformations in mixture models).* Mixture models can exploit the benefit of transformation models (Section 4.2). Simply treat the parameters of the mixture as a function of the input x, instead of fixed quantities, like in

$$p(x) = \sum_{i=1}^{k} \pi_i(x) p(x|\theta_i(x)),$$

and make use of the reparametrization trick (Section 4.2.2). ◊

4.4 Copula models

If you were to measure the speed of a car, would you measure it in kilometers per hour? Or in miles per hour? Or in meters per second? Or in the logarithm of yards per minute? Each alternative will shape the distribution of the measurements differently, and if these measurements are recorded together with some other variables, also the shape of their joint distribution, dependence structures, and the results of subsequent learning algorithms.

The previous illustrates that real-world data is composed by variables greatly different in nature and form. Even more daunting, all of these variables interact with each other in heterogeneous and complex patterns. In the language of statistics, such depiction of the world calls for the development of flexible multivariate models, able to separately characterize the marginal distributions of each of the participating random variables from the way on which they interact with each other. Copulas offer a flexible framework to model joint distributions by separately characterizing the marginal distributions of the involved variables, and the dependence structures joining these random variables together. The richness of copulas allows to model subtle

structures, such as heavy tailed and skewed dependencies, difficult to capture with transformation and mixture models.

Let $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ be two continuous random variables with positive density almost everywhere. Then, if $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are independent, their joint cdf is the product of the two marginal cdfs:

$$P(\boldsymbol{x}_1, \boldsymbol{x}_2) = P(\boldsymbol{x}_1)P(\boldsymbol{x}_2). \tag{4.4}$$

However, when $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are not independent this is no longer the case. Nevertheless, we can correct these differences by using a specific function C to couple the two marginals together into the bivariate model of interest:

$$P(\boldsymbol{x}_1, \boldsymbol{x}_2) = C(P(\boldsymbol{x}_1), P(\boldsymbol{x}_2)). \tag{4.5}$$

This cdf C is the *copula* of the distribution $P(\boldsymbol{x}_1, \boldsymbol{x}_2)$. Informally speaking, C links the univariate random variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ into a bivariate distribution $P(\boldsymbol{x}_1, \boldsymbol{x}_2)$ which exhibits a dependence structure fully described by C. Thus, any continuous bivariate distribution is the product of three independent building blocks: the marginal distribution of the first random variable $\boldsymbol{x}_1$, the marginal distribution of the second random variable $\boldsymbol{x}_2$, and the copula function C describing how the two variables interact with each other.

Remark 4.3 *(History of copulas).* The birth of copulas dates back to the pioneering work of Hoeffding (1994), who unwittingly invented the concept as a byproduct of scale-invariant correlation theory. Their explicit discovery is due to Sklar (1959), who established the fundamental result that now carries his name. Although copulas played an important role in the early development of dependence measures (Schweizer and Wolff, 1981) and probabilistic metric spaces (Schweizer and Sklar, 1983), their mainstream presence in the statistics literature had to wait for four decades, with the appearance of the monographs of Joe (1997) and Nelsen (2006). Since then, copulas have enjoyed great success in a wide variety of applications such as finance (Cherubini et al., 2004; Trivedi and Zimmer, 2007), extreme events in natural phenomena (Salvadori et al., 2007), multivariate survival modeling (Georges et al., 2001), spatial statistics, civil engineering, and random vector generation (Jaworski et al., 2010). Copula theory has likewise greatly expanded its boundaries, including the development of conditional models (Patton, 2006) and nonparametric estimators (Rank, 2007).

Perhaps surprisingly, the machine learning community has until recently been ignorant to the potential of copulas as tools to model multivariate dependence. To the best of our knowledge, the work of Chen and Gopinath (2001) in Gaussianization and the one of Kirshner (2007) on averaged copula tree models were the first to appear in a major machine learning venue. Since then, the applications of copulas in machine learning have extended to scale-invariant component analysis (Ma and Sun, 2007; Kirshner and Póczos, 2008), measures of dependence (Póczos et al., 2012; Lopez-Paz et al., 2013a),

semiparametric estimation of high-dimensional graph models (Liu et al., 2009), nonparametric Bayesian networks (Elidan, 2010), mixture models (Fujimaki et al., 2011; Tewari et al., 2011), clustering (Rey and Roth, 2012), Gaussian processes (Wilson and Ghahramani, 2010) and financial time series modeling (Hernández-Lobato et al., 2013). Elidan (2013) offers a monograph on the ongoing synergy between machine learning and copulas. $\Diamond$

Much of the study of joint distributions is the study of copulas (Trivedi and Zimmer, 2007). Copulas offer a clearer view of the underlying dependence structure between random variables, since they clean any spurious patterns generated by the marginal distributions. Let us start with the formal definition of a copula:

Definition 4.1 (Copula)**.** *A copula is a function* $C : [0,1]^2 \mapsto [0,1]$ *s.t.:*

- *for all* $u, v \in [0,1]$, $C(u,0) = C(0,v) = 0$, $C(u,1) = u$, $C(1,v) = v$,
- *for all* $u_1, u_2, v_1, v_2 \in [0,1]$ *s.t.* $u_1 \leq u_2$ *and* $v_1 \leq v_2$,

$$C(u_2, v_2) - C(u_2, v_1) - C(u_1, v_2) + C(u_1, v_1) \geq 0.$$

Alternatively, $C : [0,1]^2 \mapsto [0,1]$ *is a copula if* $C(u,v)$ *is the joint cdf of a random vector* (U,V) *defined on the unit square* $[0,1]^2$ *with uniform marginals (Nelsen, 2006, Def. 2.2.2.).*

As illustrated by Equation 4.5, the main practical advantage of copulas is that they decompose the joint distribution P into its marginal distributions $P(\boldsymbol{x}_1)$, $P(\boldsymbol{x}_2)$ and its dependence structure C. This means that one can estimate $P(\boldsymbol{x}_1, \boldsymbol{x}_2)$ by separately estimating $P(\boldsymbol{x}_1)$, $P(\boldsymbol{x}_2)$ and C. Such modus operandi is supported by a classical result due to Abe Sklar, which establishes the unique relationship between probability distributions and copulas.

Theorem 4.1 (Sklar)**.** *Let* $P(\boldsymbol{x}_1, \boldsymbol{x}_2)$ *have continuous marginal cdfs* $P(\boldsymbol{x}_1)$ *and* $P(\boldsymbol{x}_2)$*. Then, there exists a unique copula* C *such that for all* $x_1, x_2 \in \mathbb{R}$,

$$P(\boldsymbol{x}_1, \boldsymbol{x}_2) = C(P(\boldsymbol{x}_1), P(\boldsymbol{x}_2)). \tag{4.6}$$

If $P(\boldsymbol{x}_1)$, $P(\boldsymbol{x}_2)$ *are not continuous,* C *is uniquely identified on the support of* $P(\boldsymbol{x}_1) \times P(\boldsymbol{x}_2)$*. Conversely, if* C *is a copula and* $P(\boldsymbol{x}_1), P(\boldsymbol{x}_2)$ *are some continuous marginals, the function* $P(\boldsymbol{x}_1, \boldsymbol{x}_2)$ *in (4.6) is a valid* $2-$*dimensional distribution with marginals* $P(\boldsymbol{x}_1), P(\boldsymbol{x}_2)$ *and dependence structure* C*.*

In terms of density functions, the relationship (4.6) is

$$p(\boldsymbol{x}_1, \boldsymbol{x}_2) = p(\boldsymbol{x}_1)p(\boldsymbol{x}_2)c(P(\boldsymbol{x}_1), P(\boldsymbol{x}_2)), \tag{4.7}$$

where $p(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{\partial^2 P(\boldsymbol{x}_1, \boldsymbol{x}_2)}{\partial x_1 \partial x_2}$, $p(\boldsymbol{x}_i) = \frac{\partial P(\boldsymbol{x}_i)}{\partial x_i}$ *and* $c = \frac{\partial^2 C}{\partial x_1 \partial x_2}$.

Proof. See (Nelsen, 2006, Thm. 2.3.3.). □

There is an useful asymmetry in the previous claim. Given a distribution $P(\boldsymbol{x}_1, \boldsymbol{x}_2)$, we can uniquely identify its underlying dependence structure or copula C. On the other hand, given a copula C, there are infinitely multiple different bivariate models, each obtained by selecting a different pair of marginal distributions $P(\boldsymbol{x}_1)$ and $P(\boldsymbol{x}_2)$. This one-to-many relationship between copulas and probability distributions is the second most attractive property of the former: copulas are invariant with respect to strictly monotone increasing transformations of random variables.

Lemma 4.1 (Scale-invariance of copulas)**.** *Let $f_1, f_2 : \mathbb{R} \mapsto \mathbb{R}$ be two strictly monotone increasing functions. Then, for any pair of continuous random variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, the distributions:*

$$P(\boldsymbol{x}_1, \boldsymbol{x}_2) = C(P(\boldsymbol{x}_1), P(\boldsymbol{x}_2)) \text{ and}$$
$$P(f_1(\boldsymbol{x}_1), f_2(\boldsymbol{x}_2)) = C(P(f_1(\boldsymbol{x}_1)), P(f_2(\boldsymbol{x}_2)))$$

share the same copula function C (Nelsen, 2006, Thm. 2.4.3.).

Another way to understand the scale invariance of copulas is that they always exhibit uniformly distributed marginals. This is due to a classical result of Rosenblatt (1952):

Theorem 4.2 (Probability integral transform)**.** *Let the random variable $\boldsymbol{x}$ have a continuous distribution with cumulative distribution function P. Then, the random variable $\boldsymbol{y} \equiv P(\boldsymbol{x})$ is uniformly distributed.*

Scale invariance makes copulas an attractive tool to construct scale-invariant (also known as weakly equitable) statistics, such as measures of dependence (Póczos et al., 2012; Lopez-Paz et al., 2013a). We now turn to this issue, the one of measuring statistical dependence using copulas.

4.4.1 Describing dependence with copulas

The first use of copulas as explicit models of dependence is due to Schweizer and Wolff (1981), as a mean to guarantee the scale invariance (Lemma 4.1) imposed by Rényi's axiomatic framework for measures of dependence (Rényi, 1959). Given their interpretation as dependence structures, it is no surprise that copulas share an intimate relationship with well known dependence statistics, like Spearman's ρ, Kendall's τ, and mutual information:

$$\begin{aligned}
\rho(\boldsymbol{x}_1, \boldsymbol{x}_2) &= 12 \iint_0^1 (C(u_1, u_2) - u_1 u_2) \mathrm{d}u_1 \mathrm{d}u_2, \\
\tau(\boldsymbol{x}_1, \boldsymbol{x}_2) &= 4 \iint_0^1 C(u_1, u_2) \mathrm{d}C(u_1, u_2) - 1, \\
I(\boldsymbol{x}_1, \boldsymbol{x}_2) &= \iint_0^1 c(u_1, u_2) \log c(u_1, u_2) \mathrm{d}u_1 \mathrm{d}u_2 \,,
\end{aligned} \tag{4.8}$$

where $u_1 = P(\boldsymbol{x}_1 = x_1)$ and similarly for u_2. Kendall's τ measures correlation between rank statistics; as such, it is invariant under monotone increasing transformations of random variables.

Copulas are also appropriate to measure dependence between extreme events as *tail dependencies*. Formally, we define the *lower and upper tail dependence coefficients* as the quantities

$$\lambda_l = \lim_{u \to 0} P(\boldsymbol{x}_2 \leq P_2^{-1}(u) | \boldsymbol{x}_1 \leq P_1^{-1}(u)) = \lim_{u \to 0} \frac{C(u,u)}{u},$$
$$\lambda_u = \lim_{u \to 1} P(\boldsymbol{x}_2 > P_2^{-1}(u) | \boldsymbol{x}_1 > P_1^{-1}(u)) = \lim_{u \to 1} \frac{1 - 2u + C(u,u)}{1-u},$$

where P_1^{-1} and P_2^{-1} are the quantile distribution functions of the random variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, respectively. For instance, the upper tail dependence coefficient λ_u measures the probability of $\boldsymbol{x}_1$ exceeding a very large quantile, conditioned on $\boldsymbol{x}_2$ exceeding that same very large quantile. Tail dependency differs from the usual notion of statistical dependence: even a strongly correlated Gaussian distribution exhibits no tail dependency.

4.4.2 Estimation of copulas from data

Having access to n samples $X = \{(x_{1,i}, x_{2,i})\}_{i=1}^n$ drawn iid from the probability distribution $P(\boldsymbol{x}_1, \boldsymbol{x}_2)$, the question of how to estimate a model for the copula of $P(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is of immediate practical interest. The standard way to proceed is

1. Estimate the marginal cdfs $P(\boldsymbol{x}_1)$ and $P(\boldsymbol{x}_2)$ as the marginal ecdfs $P_n(\boldsymbol{x}_1)$ and $P_n(\boldsymbol{x}_2)$.

2. Obtain the copula pseudo-sample

$$U = \{(u_{1,i}, u_{2,i})\}_{i=1}^n := \{(P_n(\boldsymbol{x}_1 = x_{1,i}), P_n(\boldsymbol{x}_2 = x_{2,i})\}_{i=1}^n.$$

3. Choose a parametric copula function C_θ and estimate its parameters θ.

First, the transformation from the distribution sample X to the copula sample U involves learning the marginal cdfs $P(\boldsymbol{x}_1)$ and $P(\boldsymbol{x}_2)$ from data. In practice, the empirical cdf (Definition 2.3) is the tool of choice to obtain nonparametric estimates of univariate cdfs.

Second, when working with $d-$dimensional samples $\{(x_{1,i}, \ldots, x_{d,i})\}_{i=1}^n$, we need to compute d independent ecdfs to unfold the underlying copula sample U. The transformation of each of the components of a random vector to follow an uniform distribution by means of their ecdfs is the *empirical copula transformation*.

Definition 4.2 (Empirical copula transformation)**.** *Let* $\{(x_{1,i}, \ldots, x_{d,i})\}_{i=1}^{n}$, $x_i \in \mathbb{R}^d$*, be an iid sample from a probability density over* $\mathbb{R}^d$ *with continuous marginal cdfs* $P(\boldsymbol{x}_1), \ldots, P(\boldsymbol{x}_d)$*;* $P(\boldsymbol{x}_i) : \mathbb{R} \mapsto [0, 1]$*. Let* $P_{n,1}, \ldots, P_{n,d}$ *be the corresponding ecdfs as in Definition 2.3. The empirical copula transformation of* x *is*

$$u = T_n(x) = [P_{n,1}(x_i), \ldots, P_{n,d}(x_d)] \in \mathbb{R}^d.$$

Given that the d marginal transformations are independent from each other, we can straightforwardly use the result from Theorem 2.1 to obtain a guarantee for the fast convergence rate of the empirical copula transformation to its asymptotic limit as $n \to \infty$.

Corollary 4.1 (Convergence of the empirical copula)**.** *Let*

$$\{(x_{i,1}, \ldots, x_{i,d})\}_{i=1}^{n},$$

$x_i \in \mathbb{R}^d$*, be an iid sample from a probability density over* $\mathbb{R}^d$ *with continuous marginal cdfs* $P(\boldsymbol{x}_1), \ldots, P(\boldsymbol{x}_d)$*. Let* $T(x)$ *be the copula transformation obtained using the true marginals cdfs* $P(\boldsymbol{x}_1), \ldots, P(\boldsymbol{x}_d)$ *and let* $T_n(x)$ *be the empirical copula transformation from Definition 4.2. Then, for any* $\epsilon > 0$

$$\mathbb{P}\left[\sup_{x \in \mathbb{R}^d} \|T(x) - T_n(x)\|_2 > \epsilon\right] \leq 2d \exp\left(-\frac{2n\epsilon^2}{d}\right).$$

Proof. Use Theorem 2.1 taking into account that $\|\cdot\|_2 \leq \sqrt{d}\|\cdot\|_\infty$ in $\mathbb{R}^d$. Then apply the union-bound over the d dimensions (Póczos et al., 2012). □

Third, once we have obtained the copula sample U, we may want to fit a parametric copula model to it. For instance, we could use maximum likelihood estimation to tune the parameters of a parametric copula density. But when considering multiple candidate parametric copula families, this procedure becomes computationally prohibitive. Instead, one exploit the fact that most bivariate copulas with a single scalar parameter share a one-to-one relationship between Kendall's τ and their parameter. This means that given an estimate of Kendall's τ built from the copula sample, one can obtain an estimate of the parameter of a parametric copula by inverting the relationship (4.8). This is an efficient procedure, since the estimation of Kendall's τ from n data takes $O(n \log n)$ time. This is *the inversion method* (Dissmann et al., 2013).

Example 4.1 *(Construction of a parametric bivariate copula)***.** Figure 4.5 illustrates the estimation of a parametric bivariate copula. This process involves 1) computing the two marginal ecdfs, 2) obtaining the copula sample, and 3) fitting a parametric model using Kendall's τ inversion. A density estimate of the probability distribution of the original data is obtained by multiplying the density estimates of the marginals and the density estimate of the copula, as illustrated in Figure 4.5. ◊

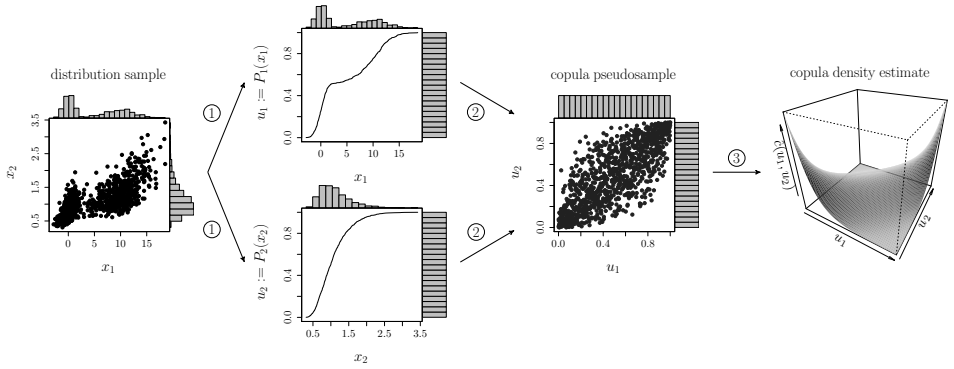


Figure 4.5: Estimation of a parametric bivariate copula

Remark 4.4 *(Wonders and worries of copulas).* Copulas transform each of the variables from our data to follow an uniform distribution. Thus, copula data reveals the essence of the dependencies in data by disentangling the complex shapes of marginal distributions. Copula data also brings a benefit when dealing with outliers: copulas upper bound the influence of outliers by squeezing all data to fit in the unit interval.

Nevertheless, the use of copulas also calls for caution. For instance, copula transformations destroy cluster structures in data. This is because the uniform margins in copulas flatten regions of high density (indications of cluster centers) and fill regions of low density (indications of boundaries between clusters). See for example, in Figure 4.5, how the cluster structure of the data sample is no longer present in the associated copula sample. Moreover, d-dimensional copulas live on the d-dimensional unit hypercube; this may be a bad parametrization for statistical models expecting data with full support on $\mathbb{R}^d$. Lastly, copulas may destroy smoothness: for instance, the copula transformation of a sinusoidal pattern is a saw-like pattern. $\Diamond$

4.4.3 Conditional distributions from copulas

As we will see in Section 4.5.2, conditional distributions play a central role in the construction of multivariate copulas. Using copulas, formulas for conditional distributions can be obtained by partial differentiation.

Definition 4.3 (Copula conditional distributions). *Define the quantity*

$$c_v(u) = P(u|v) = \frac{\partial}{\partial v} C(u, v).$$

For any $u \in [0, 1]$, $c_v(u)$ exists almost surely for all $v \in [0, 1]$, is bounded between 0 and 1, well defined, and nondecreasing almost everywhere on $[0, 1]$. Similar claims follow when conditioning on u (Salvadori et al., 2007).

Schepsmeier and Stöber (2014) provide a collection of closed-form expressions for the partial derivatives of common bivariate parametric copulas.

When conditioning to more than one variable, the previous definition extends recursively:

$$P(u|v) = \frac{\partial\, C(P(u|v_{-j}), P(v_j|v_{-j})|v_{-j})}{\partial P(v_j|v_{-j})}, \tag{4.9}$$

where the copula C is conditioned to v_{-j}, the vector of variable values v with its j-th component removed. Conditional distributions are central to copula sampling algorithms. Sampling from $C(u, v)$ reduces to i) generate $u \sim \mathcal{U}[0, 1]$ and ii) set $v = c_v^{-1}(u)$ (Nelsen, 2006, Thm. 2.2.7.). The univariate function $c_v(t)$ is inverted numerically.

4.4.4 Parametric copulas

There exists a wide catalog of parametric bivariate copulas. For completeness, we review here the most common families and their properties.

Elliptical These are copulas implicitly derived from elliptically contoured (radially symmetric) probability distributions, and represent linear dependence structures (correlations). When coupled with arbitrary marginals (multimodal, heavy-tailed...), they construct a wide-range of distributions.

- The *Gaussian copula* (Table 4.1, #1) with correlation parameter $\theta \in [-1, 1]$ represents the dependence structure underlying a bivariate Gaussian distribution of two random variables with correlation θ. Gaussian copulas exhibit no tail dependence, which makes them a poor choice to model extreme events. In fact, this is one of the reasons why the Gaussian copula has been demonized as one of the causes of the 2007 financial crisis. The family of distributions with Gaussian copula are the *nonparanormals* (Liu et al., 2009). The parameter of a multivariate Gaussian copula is a correlation matrix.

- The *t-Copula* (Table 4.1, #2) represents the dependence structure implicit in a bivariate Student-t distribution. Thus, t-Copulas are parametrized by their correlation $\theta \in [-1, 1]$ and the degrees of freedom $\nu \in (0, \infty)$. t-Copulas exhibit symmetric lower and upper tail dependencies, of strengths

$$\lambda_l(\theta, \nu) = \lambda_u(\theta, \nu) = 2\, t_{\nu+1}\left(-(\sqrt{\nu+1}\sqrt{1-\theta})/\sqrt{1+\theta}\right), \tag{4.10}$$

  where $t_{\nu+1}$ denotes the density of a Student-t distribution with $\nu + 1$ degrees of freedom. This makes t-Copulas suitable models of symmetric extreme events (happening both in the lower and upper quantiles).

To capture asymmetries in tail dependencies, Demarta and McNeil (2005) proposes a variety of skewed t-copulas. The parameters of a multivariate t-Copula are one correlation matrix, and the number of degrees of freedom.

Archimedean These copulas are popular due to their ease of construction. They are not necessarily elliptical, admit explicit constructions and accommodate asymmetric tails. They originated as an extension of the triangle inequality for probabilistic metric spaces (Schweizer and Sklar, 1983). Archimedean copulas admit the representation:

$$C(u,v) = \psi^{[-1]}(\psi(u|\theta) + \psi(v|\theta)|\theta),$$

where the continuous, strictly decreasing and convex function $\psi : [0,1] \times \Theta \to [0,\infty)$ is a *generator function*. The generalized inverse $\psi^{[-1]}$ is

$$\psi^{[-1]}(t|\theta) = \begin{cases} \psi^{-1}(t|\theta) & \text{if } 0 \le t \le \psi(0|\theta) \\ 0 & \text{if } \psi(0|\theta) \le t \le \infty. \end{cases}$$

Archimedean copulas are commutative ($C(u,v) = C(v,u)$), associative ($C(C(u,v),w) = C(u,C(v,w))$), partially ordered (for $u_1 \le u_2$ and $v_1 \le v_2$, $C(u_1,v_1) \le C(u_2,v_2)$) and have convex level curves (Nelsen, 2006).

Choosing different generator functions ψ yields different Archimedean copulas: for common examples, refer to Table 4.1, #6-12.

Extreme-value These copulas are commonly used in risk management and are appropriate to model dependence between rare events, such as natural disasters or large drops in stock markets. They satisfy:

$$C(u^t, v^t) = C^t(u,v), \quad C(u,v) = e^{\ln(u,v)A\left(\frac{\ln v}{\ln uv}\right)}, \quad t \ge 0,$$

where A is a convex *Pickands dependence function* with $\max(t, 1-t) \le A(t) \le 1$ (Nelsen, 2006). Examples are Gumbel (the only Archimedean extreme-value copula), Husler-Reiss, Galambos or Tawn (Table 4.1 #13-16).

Perfect (in)dependence From (4.4) and (4.6), we see that the only copula describing independence has cdf

$$C_\perp(u,v) = uv. \tag{4.11}$$

On the other hand, the copulas describing perfect positive (comonotonicity) or negative (countermonotonicity) dependence are respectively called the lower and upper Fréchet-Hoeffding bounds, and follow the distributions:

$$C_l(u,v) = \max(u+v-1, 0) \quad \text{and} \quad C_u(u,v) = \min(u,v). \tag{4.12}$$

Any copula C lives inside this pyramid, i.e., $C_l(u,v) \le C(u,v) \le C_u(u,v)$.

The Clayton, Gumbel, Gaussian and t-Copula are some examples of *comprehensive copulas*: they interpolate the Fréchet-Hoeffding bounds (4.12) as their parameters vary between extremes.

Combinations of copulas The convex combination or product of copulas densities is a valid copula density (Nelsen, 2006, §3.2.4). Furthermore, if $C(u, v)$ is a copula and $\gamma : [0, 1] \mapsto [0, 1]$ is a concave, continuous and strictly increasing function with $\gamma(0) = 0$ and $\gamma(1) = 1$, then $\gamma^{-1}(C(u, v))$ is also a valid copula (Nelsen, 2006, Thm. 3.3.3).

Table 4.1 summarizes a variety of bivariate parametric copulas. In this table, Φ_2 is the bivariate Gaussian CDF with correlation θ. Φ^{-1} is the univariate Normal quantile distribution function. $t_{2;\nu,\theta}$ is the bivariate Student's t CDF with ν degrees of freedom and correlation θ. t_ν^{-1} is the univariate Student's t quantile distribution function with ν degrees of freedom. D is a Debye function of the first kind. $A(w)$ is a Pickands dependence function. For contour plots of different parametric copulas, refer to (Salvadori et al., 2007).

4.4.5 Gaussian process conditional copulas

The extension of the theory of copulas to the case of conditional distributions is due to Patton (2006). Let us assume, in addition to $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, the existence of a third random variable $\boldsymbol{x}_3$. This third variable influences $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, in the sense that the joint distribution for $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ changes as we condition to different values of $\boldsymbol{x}_3$. The conditional cdf for $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ given $\boldsymbol{x}_3 = x_3$ is $P(\boldsymbol{x}_1, \boldsymbol{x}_2|x_3)$.

We can apply the copula framework to decompose $P(\boldsymbol{x}_1, \boldsymbol{x}_2|x_3)$ into its bivariate copula and one-dimensional marginals. The resulting decomposition is similar to the one shown in (4.5) for the unconditional distribution of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. But, since we are now conditioning to $\boldsymbol{x}_3 = x_3$, both the copula and the marginals of $P(\boldsymbol{x}_1, \boldsymbol{x}_2|x_3)$ depend on the value x_3 taken by the random variable $\boldsymbol{x}_3$. The copula of $P(\boldsymbol{x}_1, \boldsymbol{x}_2|x_3)$ is the *conditional copula* of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, given $\boldsymbol{x}_3 = x_3$ (Patton, 2006).

#	**Name**	**Cumulative Distribution F.** $C(u,v) =$	**Par. Domain**	**Kendall's** $\tau =$	λ_l	λ_v
1	Gaussian	$\Phi_2(\Phi^{-1}(u), \Phi^{-1}(v); \theta)$	$\theta \in [-1,1]$	$\frac{2}{\pi}\arcsin(\theta)$	0	0
2	t-Student	$t_{2;\nu,\theta}(t_\nu^{-1}(u), t_\nu^{-1}(v); \theta)$	$\theta \in [-1,1]$, $\nu > 0$		Equation (4.10)	
3	Independent	uv	—	0	0	0
4	Upper FH bound	$\min(u,v)$	—	1	1	1
5	Lower FH bound	$\max(u+v-1, 0)$	—	-1	0	0
6	**Archimedean**	$\psi^{[-1]}\left(\psi(u;\theta) + \psi(v;\theta); \theta\right)$	$\psi-$dependent	$1 + 4\int_0^1 \frac{\psi(t)}{\psi'(t)}\mathrm{d}t$	ψ-dependent	
7	Ali-Mikhail-Haq	$\frac{uv}{1-\theta(1-u)(1-v)}$	$\theta \in [-1,1]$	$1 - \frac{2(\theta+(1-\theta)^2\log(1-\theta))}{3\theta^2}$	0	0
8	Clayton	$\max\left((u^{-\theta} + v^{-\theta} - 1), 0\right)^{-1/\theta}$	$\theta \geq -1$	$\theta/(\theta+2)$	$2^{-1/\theta}$	0
9	Frank	$\frac{1}{\ln\theta}\ln\left(1 + \frac{(\theta^u-1)(\theta^v-1)}{\theta-1}\right)$	$\theta \geq 0$	$1 + 4(D(\theta) - 1)/\theta$	0	0
10	Gumbel	$\exp\left(-\left((-\ln u)^\theta + (-\ln v)^\theta\right)^{1/\theta}\right)$	$\theta \geq 1$	$(\theta-1)(\theta)$	0	$2 - 2^{1/\theta}$
11	Joe	$1 - \left((1-u)^\theta + (1-v)^\theta - (1-u)^\theta(1-v)^\theta\right)^{1/\theta}$	$\theta \geq 1$	$1 - \sum_{k=1}^{\infty} \frac{4}{k(\theta k+2)(\theta(k-1)+2)}$	0	$2 - 2^{1/\theta}$
12	Kimeldorf-Sampson	$(u^{-\theta} + v^{-\theta} - 1)^{-1/\theta}$	$\theta \geq 0$	$\theta/(\theta+2)$	$2^{-1/\theta}$	0
13	**Extreme-Value**	$\exp\left(\ln u + \ln v\right) A\left(\frac{\ln v}{\ln u + \ln v}\right)$	$A-$dependent	$\int_0^1 \frac{w(1-w)}{A(w)} A''(w)\mathrm{d}w$	A-dependent	
14	Galambos	$uv\exp\left((-\ln u)^\theta + (-\ln v)^{-\theta}\right)^{-1/\theta}$	$\theta \geq 0$		0	$2^{-1/\theta}$
15	Hüsler-Reiss	$e^{\ln(u)\Phi\left(\frac{1}{\theta} + \frac{\theta}{2}\ln\frac{\ln u}{\ln v}\right) + \ln(v)\Phi\left(\frac{1}{\theta} + \frac{\theta}{2}\ln\frac{\ln u}{\ln v}\right)}$	$\theta \geq 0$		0	$2 - 2\Phi(1/\theta)$
16	Tawn	$e^{(1-\alpha)\ln u + (1-\beta)\ln v - ((-\alpha\ln u)^\gamma + (-\beta\ln v)^\gamma)^{1/\gamma}}$	$\alpha, \beta \in [0,1]$, $\gamma \geq 0$		numerical approx.	
17	FGM	$uv(1 + \theta(1-u)(1-v))$	$\theta \in [-1,1]$	$(2\theta)/9$	0	0
18	Marshall-Olkin	$\min\left(u^{1-\alpha}v, uv^{1-\beta}\right)$	$\alpha, \beta \in [0,1]$	$(\alpha\beta)/(2\alpha + 2\beta - \alpha\beta)$	0	$\min(\alpha, \beta)$
19	Plackett	$\frac{1+(\theta-1)(u+v) - \sqrt{(1+(\theta-1)(u+v))^2 - 4\theta(\theta-1)uv}}{2(\theta-1)}$	$\theta \geq 0$, $\theta \neq 1$	numerical approx.	0	0
20	Raftery	$\#4 + \frac{1-\theta}{1+\theta}(uv)^{1/(1-\theta)}(1 - \max(u,v)^{-(1+\theta)/(1-\theta)})$	$\theta \in [0,1]$	$(2\theta)/(3-\theta)$	$\frac{2\theta}{\theta+1}$	0
21	**Nonparametric**	Section 4.4.6				
22	**Conditional**	Section 4.4.5				

Table 4.1: A zoo of copulas

Definition 4.4 (Conditional copula)**.** *The conditional copula of* $P(\boldsymbol{x}_1, \boldsymbol{x}_2|x_3)$ *is the joint distribution of* $\boldsymbol{u}_{1|3} \equiv P(\boldsymbol{x}_1|x_3)$ *and* $\boldsymbol{u}_{2|3} \equiv P(\boldsymbol{x}_1|x_3)$*, where* $P(\boldsymbol{x}_1|x_3)$ *and* $P(\boldsymbol{x}_1|x_3)$ *are the conditional marginal cdfs of* $P(\boldsymbol{x}_1, \boldsymbol{x}_2|x_3)$*.*

Theorem 4.3 (Sklar's theorem for conditional distributions)**.** *Let* $P(\boldsymbol{x}_1, \boldsymbol{x}_2|x_3)$ *be the conditional joint cdf for* $\boldsymbol{x}_1$ *and* $\boldsymbol{x}_2$ *given* $\boldsymbol{x}_3 = x_3$ *and let* $P(\boldsymbol{x}_1|x_3)$ *and* $P(\boldsymbol{x}_2|x_3)$ *be its continuous conditional marginal cdfs. Then, there exists a unique conditional copula* $C(\boldsymbol{x}_1, \boldsymbol{x}_2|x_3)$ *such that*

$$P(x_1, x_2|x_3) = C(P(x_1|x_3), P(x_2|x_3)|x_3) \tag{4.13}$$

for any x_1*,* x_2 *and* x_3 *in the support of* $\boldsymbol{x}_1$*,* $\boldsymbol{x}_2$ *and* $\boldsymbol{x}_3$*, respectively. Conversely, if* $P(\boldsymbol{x}_1|x_3)$ *and* $P(\boldsymbol{x}_1|x_3)$ *are the conditional cdfs of* $\boldsymbol{x}_1$ *and* $\boldsymbol{x}_2$ *given* $\boldsymbol{x}_3 = x_3$ *and* $C(\boldsymbol{x}_1, \boldsymbol{x}_2|x_3)$ *is a conditional copula, then (4.13) is a valid conditional joint distribution with marginals* $P(\boldsymbol{x}_1|x_3)$ *and* $P(\boldsymbol{x}_2|x_3)$ *and dependence structure* $C(\boldsymbol{x}_1, \boldsymbol{x}_2|x_3)$*.*

Proof. See (Patton, 2002). □

In the following, we describe a novel method based on Gaussian processes (Rasmussen and Williams, 2006) to estimate conditional copulas (Lopez-Paz et al., 2013b).

Semiparametric conditional copulas

Let $\mathcal{D}_{1,2} = \{x_{1,i}, x_{2,i}\}_{i=1}^n$ and $\mathcal{D}_3 = \{x_{3,i}\}_{i=1}^n$ form a dataset corresponding to n paired samples of $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ from the joint distribution $P(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$. We want to learn the conditional copula $C(\boldsymbol{x}_1, \boldsymbol{x}_2|x_3)$. For this, we first compute estimates $P_n(\boldsymbol{x}_1|x_3)$ and $P_n(\boldsymbol{x}_2|x_3)$ of the conditional marginal cdfs using the data available in $\mathcal{D}_{1,2}$ and $\mathcal{D}_3$. We can obtain a sample $\mathcal{D}'_{1,2} = \{u_{1,i}, u_{2,i}\}_{i=1}^n$ from $C(\boldsymbol{x}_1, \boldsymbol{x}_2|x_3)$ by mapping the observations for $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ to their corresponding marginal conditional probabilities given the observations for $\boldsymbol{x}_3$:

$$u_{1,i} = P_n(\boldsymbol{x}_1 = x_{1,i}|x_{3,i}), \;\; u_{2,i} = P_n(\boldsymbol{x}_2 = x_{2,i}|x_{3,i}), \;\; \text{for } i = 1, \ldots, n\,.$$

This pair of transformations are computed by i) estimating the marginal cdfs $P(\boldsymbol{x}_1)$ and $P(\boldsymbol{x}_2)$, ii) estimating two parametric copulas $C(P(\boldsymbol{x}_1), P(\boldsymbol{x}_3))$ and $C(P(\boldsymbol{x}_2), P(\boldsymbol{x}_3))$, and iii) estimating the conditional distributions $P(\boldsymbol{x}_1 \,|\, x_3)$ and $P(\boldsymbol{x}_2 \,|\, x_3)$ from such parametric copulas, as explained in Section 4.4.3.

The data in $\mathcal{D}'_{1,2}$ and $\mathcal{D}_3$ can be used to adjust a semiparametric model for $C(\boldsymbol{x}_1, \boldsymbol{x}_2|x_3)$. In particular, we assume that $C(\boldsymbol{x}_1, \boldsymbol{x}_2|x_3)$ follows the shape of a parametric copula, specified in terms of its Kendall's τ statistic, where the value of τ depends on the value of x_3. The parameter θ of the copula can be easily obtained as a function of Kendall's τ. The connection between τ and x_3 is a latent function g, such that $\tau = g(x_3)$. To ease estimation,

g is the composition of an unconstrained function f and a link function σ mapping the values of f to valid Kendall τ values. For example, we can fix $\sigma(x) = 2\Phi(x) - 1$, where Φ is the standard Gaussian cdf: this particular choice maps the real line to the interval $[-1, 1]$. After choosing a suitable σ, our semiparametric model can make use of unconstrained nonlinear functions f. We can learn g by placing a Gaussian process (GP) prior on f and computing the posterior distribution for f given $\mathcal{D}'_{1,2}$ and $\mathcal{D}_3$.

Let $f = (f(x_{3,1}), \ldots, f(x_{3,n}))^\top$ be the n-dimensional vector with the evaluation of f at the available observations from $\boldsymbol{x}_3$. Since f is a sample from a GP, the prior distribution for f given $\mathcal{D}_3$ is the Gaussian:

$$p(f|\mathcal{D}_3) = \mathcal{N}(f|m_0, K)\,,$$

where m_0 is an n-dimensional vector with the evaluation of the mean function m at $\mathcal{D}_3$, that is, $m_0 = (m(x_{3,1}), \ldots, m(x_{3,n}))^\top$ and K is an $n \times n$ kernel matrix generated by the evaluation of the kernel k at $\mathcal{D}_3$, that is, $k_{i,j} = k(x_{3,i}, x_{3,j})$. We select a constant function for m and the Gaussian kernel k:

$$\begin{aligned} m(x) &= m_0, \\ k(x_i, x_j) &= \sigma^2 \exp\left\{-(x_i - x_j)^2 \lambda^{-2}\right\} + \sigma_0^2\,, \end{aligned} \tag{4.14}$$

where m_0, σ^2, λ and σ_0^2 are hyper-parameters. The posterior distribution for f given $\mathcal{D}'_{1,2}$ and $\mathcal{D}_3$ is

$$p(f|\mathcal{D}'_{1,2}, \mathcal{D}_3) = \frac{\left[\prod_{i=1}^n c(u_{1,i}, u_{2,i}|\sigma(f_i))\right] p(f|\mathcal{D}_3)}{p(\mathcal{D}'_{1,2}|\mathcal{D}_3)}\,. \tag{4.15}$$

In the equation above, $c(u_{1,i}, u_{2,i}|\sigma(f_i)]$ is the density function of the parametric copula model with $\tau = \sigma(f_i)$. Given $\mathcal{D}_{1,2}$, $\mathcal{D}_3$ and a particular assignment $\boldsymbol{x}_3 = x_3^\star$, we can make predictions for the conditional distribution of $u_1^\star = P(x_1^\star|x_3^\star)$ and $u_2^\star = P(x_2^\star|x_3^\star)$, where $x_1^\star$ and $x_2^\star$ are samples from $p(x_1, x_2|x_3^\star)$. In particular, we have that

$$p(u_1^\star, u_2^\star|x_3^\star) = \int c(u_1^\star, u_2^\star|\sigma(f^\star)) p(f^\star|f) p(f|\mathcal{D}'_{1,2}, \mathcal{D}_3)\, df\, df^\star\,, \tag{4.16}$$

where $f^\star = f(x_3^\star)$, $p(f^\star|f) = \mathcal{N}(f^\star|k_\star^\top K^{-1} f, k_{\star,\star} - k_\star^\top K^{-1} k)$, k is an n-dimensional vector with the prior covariances between $f(x_3^\star)$ and $\{f(x_3^{(i)})\}_{i=1}^n$ and $k_{\star,\star} = k(x_3^\star, x_3^\star)$. Unfortunately, the exact computation of (4.15) and (4.16) is intractable. To circumvent this issue, we resort to the use of the *expectation propagation algorithm* (Minka, 2001), one alternative to efficiently compute approximations to (4.15) and (4.16).

Approximating the posterior with expectation propagation

We use expectation propagation (EP) (Minka, 2001) to obtain tractable approximations to the exact posterior (4.15) and predictive distributions

(4.16). The posterior $p(f|\mathcal{D}'_{1,2}, \mathcal{D}_3)$ is, up to a normalization constant, the product of factors

$$p(f|\mathcal{D}'_{1,2}, \mathcal{D}_3) \propto \left[\prod_{i=1}^{n} h_i(f_i)\right] h_{n+1}(f)\,, \tag{4.17}$$

where $h_i(f_i) = c_{x_1,x_2|x_3}[u_{1,i}, u_{2,i}|\sigma(f_i)]$ and $h_{n+1}(f) = \mathcal{N}(f|m_0, K)$. EP approximates (4.17) with a simpler distribution $q(f) \propto [\prod_{i=1}^{n} \tilde{h}_i(f_i)]h_{n+1}(f)$, obtained by replacing each non-Gaussian factor h_i in (4.17) with an approximate factor $\tilde{h}_i$ that is Gaussian, but unnormalized:

$$\tilde{h}_i(f_i) = c_i \exp\{-\frac{1}{2}a_i f_i^2 + b_i f_i\}\,,$$

where c_i is a positive constant and a_i and b_i are the natural parameters of the Gaussian factor $\tilde{h}_i$. Since h_{n+1} in (4.17) is already Gaussian, there is no need for its approximation. Since the Gaussian distribution belong to the exponential family of distributions, they are closed under the product and division operations, and therefore q is Gaussian with natural parameters equal to the sum of the natural parameters of the Gaussian factors $\tilde{h}_1, \ldots, \tilde{h}_n$ and h_{n+1}.

Initially all the approximate factors $\tilde{h}_i$ are uninformative or uniform, that is, $a_i = 0$ and $b_i = 0$ for $i = 1, \ldots, n$. EP iteratively updates each $\tilde{h}_i$ by first computing the *cavity* distribution $q^{\setminus i}(f) \propto q(x)/\tilde{h}_i(f_i)$ and then minimizing the Kullback-Liebler (KL) divergence between $h_i(f_i)q^{\setminus i}(f)$ and $\tilde{h}_i(f_i)q^{\setminus i}(f)$ (Minka, 2001). To achieve this, EP matches the first two moments of $h_i(f_i)q^{\setminus i}(f)$ and $\tilde{h}_i(f_i)q^{\setminus i}(f)$, with respect to f_i, after marginalizing out all the other entries in f (Seeger, 2005). In our implementation of EP, we follow Van Gerven et al. (2010) and refine all the $\tilde{h}_i$ in parallel. For this, we first compute the n-dimensional vectors $m = (m_1, \ldots, m_n)^\top$ and $v = (v_1, \ldots, v_n)^\top$ with the marginal means and variances of q, respectively. In particular,

$$\begin{aligned} v &= \mathrm{diag}\left\{\left(K^{-1} + \mathrm{diag}(a)\right)^{-1}\right\}, \\ m &= \left(K^{-1} + \mathrm{diag}(a)\right)^{-1}\left(b + K^{-1}m_0\right)\,, \end{aligned} \tag{4.18}$$

where $a = (a_1, \ldots, a_n)^\top$ and $b = (b_1, \ldots, b_n)^\top$ are n-dimensional vectors with the natural parameters of the approximate factors $\tilde{h}_1, \ldots, \tilde{h}_n$. After this, we update all the approximate factors. For this, we obtain, for $i = 1, \ldots, n$, the marginal mean $m^{\setminus i}$ and the marginal variance $v^{\setminus i}$ of f_i with respect to the cavity distribution $q^{\setminus i}$. This leads to

$$\begin{aligned} v^{\setminus i} &= (v_i^{-1} - a_i)^{-1}, \\ m^{\setminus i} &= v^{\setminus i}(m_i v_i^{-1} - b_i)\,. \end{aligned}$$

We then compute, for each approximate factor $\tilde{h}_i$, the new marginal mean and marginal variance of q with respect to f_i after updating that factor. In particular, we compute

$$m_i^{\text{new}} = \frac{1}{Z_i} \int f_i h_i(f_i) \mathcal{N}(f_i | m^{\setminus i}, v^{\setminus i}) \, df_i \,,$$
$$v_i^{\text{new}} = \frac{1}{Z_i} \int (f_i - m_i^{\text{new}})^2 h_i(f_i) \mathcal{N}(f_i | m^{\setminus i}, v^{\setminus i}) \, df_i \,.$$

where $Z_i = \int h_i(f_i) \mathcal{N}(f_i | m^{\setminus i}, v^{\setminus i}) \, df_i$ is a normalization constant. These integrals are not analytic, so we approximate them using numerical integration. The new values for a_i and b_i are

$$a_i^{\text{new}} = [v_i^{\text{new}}]^{-1} - [v^{\setminus i}]^{-1} \,, \tag{4.19}$$
$$b_i^{\text{new}} = m_i^{\text{new}} [v_i^{\text{new}}]^{-1} - m^{\setminus i} [v^{\setminus i}]^{-1} \,, \tag{4.20}$$

Once we have updated a_i and b_i, we can update the marginal mean and the marginal variance of f_i in q, namely,

$$v_i^{\text{new}} = ([v^{\setminus i}]^{-1} + a_i)^{-1},$$
$$m_i^{\text{new}} = v_i^{\text{new}} (m^{\setminus i} [v^{\setminus i}]^{-1} + b_i) \,.$$

Finally, we update c_i to be

$$\log c_i^{\text{new}} = \log Z_i + \frac{1}{2} \log v^{\setminus i} - \frac{1}{2} \log v_i^{\text{new}} + \frac{[m^{\setminus i}]^2}{2 v^{\setminus i}} - \frac{[m_i^{\text{new}}]^2}{2 v^{\text{new}}} \,. \tag{4.21}$$

This completes the operations required to update all the approximate factors $\tilde{h}_1, \ldots, \tilde{h}_n$. Once EP has updated all these factors using (4.19), (4.20) and (4.21), a new iteration begins. EP stops when the change between two consecutive iterations in the marginal means and variances of q, as given by (4.18), is less than 10^{-3}. To improve the convergence of EP and avoid numerical problems related to the parallel updates (Van Gerven et al., 2010), we damp the EP update operations. When damping, EP replaces (4.19) and (4.20) with

$$a_i^{\text{new}} = (1 - \epsilon) a_i^{\text{old}} + \epsilon \left\{ [v_i^{\text{new}}]^{-1} - [v^{\setminus i}]^{-1} \right\} \,,$$
$$b_i^{\text{new}} = (1 - \epsilon) b_i^{\text{old}} + \epsilon \left\{ m_i^{\text{new}} [v_i^{\text{new}}]^{-1} - m^{\setminus i} [v^{\setminus i}]^{-1} \right\} \,,$$

where a_i^{old} and b_i^{old} are the parameters values before the EP update. The parameter $\epsilon \in [0, 1]$ controls the amount of damping. When $\epsilon = 1$, we recover the original EP updates. When $\epsilon = 0$, the parameters of the approximate factor $\tilde{h}_i$ are not modified. We use an annealed damping scheme: we start with $\epsilon = 1$ and, after each EP iteration, we scale down ϵ by 0.99.

Some of the parameters a_i may become negative during the execution of EP. These negative variances in $\tilde{h}_1, \ldots, \tilde{h}_n$ may result in a covariance matrix $V = \left(K^{-1} + \text{diag}(a)\right)^{-1}$ for f in q that is not positive definite. Whenever this happens, we first restore all the $\tilde{h}_1, \ldots \tilde{h}_n$, to their previous value, reduce the damping parameter ϵ by scaling it by 0.5 and repeat the update of all the approximate factors with the new value of ϵ. We repeat this operation until V is positive definite.

EP can also approximate the normalization constant of the exact posterior distribution (4.15), that is, $p(\mathcal{D}'_{1,2}|\mathcal{D}_3)$. For this, note that $p(\mathcal{D}'_{1,2}|\mathcal{D}_3)$ is the integral of $[\prod_{i=1}^n h_i(f_i)]h_{n+1}(f)$. We can then approximate $p(\mathcal{D}'_{1,2}|\mathcal{D}_3)$ as the integral of $[\prod_{i=1}^n \tilde{h}_i(f_i)]h_{n+1}(f)$ once all the $\tilde{h}_i$ factors have been adjusted by EP. Since all the $\tilde{h}_i$ and h_{n+1} are Gaussian, this integral can be efficiently computed. In particular, after taking logarithms, we obtain

$$\log p(\mathcal{D}'_{1,2}|\mathcal{D}_3) \approx$$
$$\sum_{i=1}^n \log c_i - \frac{1}{2}\log|K| - \frac{1}{2}m_0^\top K^{-1} m_0 + \frac{1}{2}\log|V| + \frac{1}{2}m^\top V^{-1} m\,, \qquad (4.22)$$

where V is the covariance matrix for f in q and m is the mean vector for f in q as given by (4.18). The EP approximation to $p(\mathcal{D}'_{1,2}|\mathcal{D}_3)$ is also a proxy to adjust the hyper-parameters m_0, σ^2, σ_0^2 and λ of the mean function and the covariance function of the GP (4.14). In particular, we can obtain a type-II maximum likelihood estimate of these hyper-parameters by maximizing $\log p(\mathcal{D}'_{1,2}|\mathcal{D}_3)$ (Bishop, 2006). To solve this maximization, descend along the gradient of $\log p(\mathcal{D}'_{1,2}|\mathcal{D}_3)$ with respect to m_0, σ^2, σ_0^2 and λ. Fortunately, the right-hand side of (4.22) approximates this gradient well, if we treat the parameters of the approximate factors a_i, b_i and c_i, for $i = 1, \ldots, n$ as constants (Seeger, 2005).

Finally, the EP solution is also useful to approximate the predictive distribution (4.16). For this, we first replace $p(f|\mathcal{D}'_{1,2}, \mathcal{D}_3)$ in (4.16) with the EP approximation to this exact posterior distribution, that is, q. After marginalizing out f, we have

$$\int p(f^\star|f)p(f|\mathcal{D}'_{1,2}, \mathcal{D}_3)\,df \approx \int p(f^\star|f)q(f)\,df = \mathcal{N}(f^\star|m^\star, v^\star)$$

where

$$m^\star = k_\star^\top (K + \tilde{V})^{-1}\tilde{m}\,,$$
$$v^\star = k_{\star,\star} - k_\star^\top (K + \tilde{V})^{-1} k_\star\,,$$

$k_\star$ is an n-dimensional vector with the prior covariances between $f_\star$ and $f_1, \ldots, f_n$, $k_{\star,\star}$ is the prior variance of $f_\star$, $\tilde{m}$ is an n-dimensional vector whose i-th entry is b_i/a_i and $\tilde{V}$ is an $n \times n$ diagonal matrix whose i-th entry in the

diagonal is $1/a_i$. Once we have computed $m^\star$ and $v^\star$, we approximate the integral $\int c_{x_1,x_2|x_3}[u_1^\star, u_2^\star|\sigma(f^\star)]\mathcal{N}(f^\star|m^\star, v^\star)\, df^\star$ by Monte Carlo. For this, draw N samples $f^\star_{(1)}, \ldots, f^\star_{(N)}$ from $\mathcal{N}(f^\star|m^\star, v^\star)$ and approximate (4.16) by

$$p(u_1^\star, u_2^\star|x_3^\star) \approx \frac{1}{N}\sum_{i=1}^{N} c_{x_1,x_2|x_3}[u_1^\star, u_2^\star|\sigma(f^\star_{(i)})]\,.$$

Speeding up the computations with GPs

The computational cost of the previous EP algorithm is $O(n^3)$, due to the computation of the inverse of a kernel matrix of size $n \times n$. To reduce this cost, we use the FITC approximation for Gaussian processes described by Snelson and Ghahramani (2005). The FITC approximation replaces the $n \times n$ covariance matrix K with the low-rank matrix $K' = Q + \mathrm{diag}(K - Q)$, where $Q = K_{n,n_0}K_{n_0,n_0}^{-1}K_{n,n_0}^\top$ is a low-rank matrix, K_{n_0,n_0} is the $n_0 \times n_0$ covariance matrix generated by evaluating the covariance function k in (4.14) between some n_0 training points or pseudo-inputs and K_{n,n_0} is the $n \times n_0$ matrix with the covariances between all training points $x_{3,1}, \ldots, x_{3,n}$ and pseudo-inputs. This approximate EP algorithm has cost $O(nn_0^2)$.

4.4.6 Nonparametric copulas

In search for a higher degree of flexibility than the one provided by the parametric copulas of Section 4.4.4, one could try to perform kernel density estimation to estimate copula densities, for instance by placing a bivariate Gaussian kernel on each copula sample (u_i, v_i). However, the resulting kernel density estimate would have support on $\mathbb{R}^2$, while the support of any bivariate copula is the unit square. A workaround to this issue is to 1) transform each copula marginal distribution to have full support and 2) perform kernel density estimation on such transformed sample. Following this rationale, this section studies copula estimates of the form

$$\hat{c}(u, v) = \frac{\hat{p}_{ab}(\hat{P}_a^{-1}(u), \hat{P}_b^{-1}(v))}{\hat{p}_a(\hat{P}_a^{-1}(u))\hat{p}_b(\hat{P}_b^{-1}(v))}, \tag{4.23}$$

where

$$\hat{p}_a(u) = \sum_{i=1}^{k_a} m_{a,i}\,\mathcal{N}(u; \mu_{a,i}, \sigma_{a,i}^2),$$

$$\hat{p}_b(v) = \sum_{i=1}^{k_b} m_{b,i}\,\mathcal{N}(v; \mu_{b,i}, \sigma_{b,i}^2),$$

and

$$\hat{p}_{ab}(u, v) = \sum_{i=1}^{k_{ab}} m_{ab,i}\,\mathcal{N}(u, v; \mu_{ab,i}, \Sigma_{ab,i}^2),$$

with

$$\sum_{i=1}^{k_a} m_{a,i} = \sum_{i=1}^{k_b} m_{b,i} = \sum_{i=1}^{k_{ab}} m_{ab,i} = 1.$$

Since Gaussian mixture models are dense in the set of all probability distributions, the model in (4.23) can model a wide range of distributions. Rank (2007) set p_a and p_b to be Normal distributions, $k_{ab} = m_{ab,i}^{-1} = n$ and $\mu_{ab,i} = (P_a^{-1}(u_i), P_b^{-1}(v_i))$ yielding the so-called *nonparametric copula*, equivalent to a Gaussian kernel density estimate on the transformed sample $\{(\Phi^{-1}(u_i), \Phi^{-1}(v_i))\}_{i=1}^n$, where Φ^{-1} is the Normal inverse cdf.

Conditional distributions

The conditional distribution $P(u|v)$ for the copula model in (4.23) is

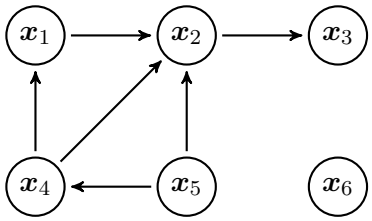
$$\begin{aligned}\hat{P}(u|v) &= \int_0^u \hat{c}(x,v)\mathrm{d}x = \int_0^u \frac{\hat{p}_{ab}(\hat{P}_a^{-1}(x), \hat{P}_b^{-1}(v))}{\hat{p}_a(\hat{P}_a^{-1}(x))\hat{p}_b(\hat{P}_b^{-1}(v))}\mathrm{d}x \\ &= \sum_{i=1}^{k_{ab}} \frac{m_{ab,i}}{\hat{p}_b(\hat{P}_b^{-1}(v))} \int_0^u \frac{\mathcal{N}(\hat{P}_a^{-1}(x), \hat{P}_b^{-1}(v); \mu_{ab,i}, \Sigma_{ab,i})}{\sum_{i=1}^{k_a} m_{a,i}\mathcal{N}(\hat{P}_a^{-1}(x); \mu_{a,i}, \sigma_{a,i}^2)}\mathrm{d}x. \end{aligned} \tag{4.24}$$

Unfortunately, the integral in (4.24) has no analytical solution for arbitrary mixtures (p_a, p_b). Let us instead restrict ourselves to the case where $p_a(x) = p_b(x) := \mathcal{N}(x; 0, 1)$. By denoting $z_i := \mu_{ab,i}^{(1)}$, $w_i := \mu_{ab,i}^{(2)}$, Lopez-Paz et al. (2012) derives

$$\begin{aligned}\hat{P}(u|v) &= \int_0^u \hat{c}(x,v)\mathrm{d}x = \int_0^u \frac{\hat{p}_{ab}(\Phi^{-1}(x), \Phi^{-1}(v))}{\phi(\Phi^{-1}(x))\phi(\Phi^{-1}(v))}\mathrm{d}x \\ &= \sum_{i=1}^{k_{ab}} \frac{m_{ab,i}}{\phi(\Phi^{-1}(v))} \int_0^u \frac{\mathcal{N}(\Phi^{-1}(x), \Phi^{-1}(v); \mu_{ab,i}, \mathbf{\Sigma}_{ab,i})}{\phi(\Phi^{-1}(u))}\mathrm{d}x \\ &= \sum_{i=1}^{k_{ab}} \frac{m_{ab,i}}{\phi(\Phi^{-1}(v))}\mathcal{N}(\Phi^{-1}(v); \mu_{ab,i}^{(2)}, \sigma_{wi}^2)\,\Phi\left(\frac{\Phi^{-1}(u) - \mu_{z_i|w_i}}{\sigma_{z_i|w_i}^2}\right),\end{aligned}$$

where $\Sigma_{ab,i} = \begin{pmatrix} \sigma_{z_i}^2 & \gamma_i \\ \gamma_i & \sigma_{w_i}^2 \end{pmatrix}$, $\mu_{z_i|w_i} = z_i + \frac{\sigma_{z_i}}{\sigma_{w_i}}\gamma_i(w - w_i)$ and $\sigma_{z_i|w_i}^2 = \sigma_{z_i}^2(1-\gamma_i^2)$, for some correlations $-1 \leq \gamma_i \leq 1$ and $1 \leq i \leq k_{ab}$. Setting $k_{ab} = m_{ab,i}^{-1} = n$, $\mu_{ab,i} = (\Phi^{-1}(u_i), \Phi^{-1}(v_i))$ and $\Sigma_{ab,i} = \begin{pmatrix} \sigma_z^2 & \gamma_i \\ \gamma_i & \sigma_w^2 \end{pmatrix}$ produces similar expressions for the nonparametric copula.

The previous are closed-form expressions for our nonparametric copula model and their exact conditional distributions. Therefore, these formulas can be used to construct vine copulas (presented in Section 4.5.2) in a consistent manner.



$$p(\boldsymbol{x} = x) = p(x_6)p(x_5)p(x_4 \,|\, x_5)p(x_1 \,|\, x_4)p(x_2 \,|\, x_1, x_4, x_5)p(x_3 \,|\, x_2)$$

Figure 4.6: A Bayesian network and its factorization.

4.5 Product models

Product models exploit the conditional probability rule

$$p(\boldsymbol{x} = x, \boldsymbol{y} = y) = p(\boldsymbol{y} = y \,|\, \boldsymbol{x} = x)p(\boldsymbol{x} = x)$$

and the conditional independence rule

$$p(\boldsymbol{x} = x, \boldsymbol{y} = y \,|\, \boldsymbol{z} = z) = p(\boldsymbol{x} = x \,|\, \boldsymbol{z} = z)p(\boldsymbol{y} = y \,|\, \boldsymbol{z} = z)$$

to express high-dimensional joint probability density function as the product of low-dimensional conditional probability density functions. As opposed to mixture models, which implement the "`OR`" operation between their components, product models implement the "`AND`" operation between their factors. The most prominent example of product models are Bayesian networks.

4.5.1 Bayesian networks

Bayesian networks (Pearl, 1985) are probabilistic graphical models that represent the joint probability distribution of a set of random variables as a Directed Acyclic Graph (DAG). In this DAG, each node represents a random variable, and each edge represents a conditional dependence between two variables. Using this graphical representation, Bayesian networks factorize probability distributions in one factor per node, equal to the conditional distribution of the variable associated with that node, when conditioned on all its parents in the graph. Figure 4.6 illustrates a Bayesian network on six random variables, and the resulting factorization of the six-dimensional density $p(\boldsymbol{x} = x)$.

The arrows in the DAG of a Bayesian network are a mathematical representation of conditional dependence: this notion has nothing to do with causation between variables. We will devote Chapter 6 to extend Bayesian networks to the language of causation.

Remark 4.5 *(Other product models).* Other product models include Markov networks and factor graphs. In contrast to Bayesian networks, Markov networks and factor graphs rely on undirected, possibly cyclic graphs. Bayesian networks and Markov networks are complimentary, in the sense that each of them can represent dependencies that the other can not. The Hammersley-Clifford theorem establishes that factor graphs can represent both Bayesian Networks and Markov networks. ◊

4.5.2 Vine copulas

We now extend the framework of copulas (Section 4.4) to model $d-$dimensional probability density functions $p(\boldsymbol{x})$ as the product of its one-dimensional marginal densities $p(\boldsymbol{x}_i)$ and its dependence structure or copula c:

$$p(\boldsymbol{x}) = \underbrace{\left[\prod_{i=1}^{d} p(\boldsymbol{x}_i)\right]}_{\text{marginals}} \cdot \underbrace{c(P(\boldsymbol{x}_1), \ldots, P(\boldsymbol{x}_d))}_{\text{dependence structure}},$$

where $P(\boldsymbol{x}_i)$ denotes the marginal cdf of $\boldsymbol{x}_i$, for all $1 \leq i \leq n$. To learn p, we first estimate each of the d marginals p_i independently, and then estimate the multivariate copula c. However, due to the curse of dimensionality, directly learning c from data is a challenging task. One successful approach to deal with this issue is to further factorize c into a product of bivariate, *parametric*, *unconditional* copulas. This is the approach of *vine decompositions* (Bedford and Cooke, 2001).

Vine copulas (Bedford and Cooke, 2001) are hierarchical graphical models that factorize a d-dimensional copula into the product of $d(d-1)/2$ bivariate copulas. Vines are flexible models, since each of the bivariate copulas in the factorization can belong to a different parametric family (like the ones described in Section 4.4.4). Multiple types of vines populate the literature; we here focus on regular vine copula distributions, since they are the most general kind (Aas et al., 2009; Kurowicka, 2011).

Remark 4.6 *(History of vine copulas).* Vines are due to Joe (1996) and Bedford and Cooke (2001). Aas et al. (2009) and Kurowicka (2011) offer two monographs on vines. Vines enjoy a mature theory, including results in sampling (Bedford and Cooke, 2002), characterization of assumptions (Haff et al., 2010; Acar et al., 2012; Lopez-Paz et al., 2013b), model selection (Kurowicka, 2011; Dissmann et al., 2013), extensions to discrete distributions (Panagiotelis et al., 2012) and identification of equivalences to other well known models, such as Bayesian belief networks and factor models (Kurowicka, 2011).

Vines have inherited the wide range of applications that copulas have enjoyed, including time series prediction, modeling of financial returns, comorbidity analysis, and spatial statistics (Kurowicka, 2011). Initial applications

in machine learning include semisupervised domain adaptation (Lopez-Paz et al., 2012) and Gaussian process conditional distribution estimation (Lopez-Paz et al., 2013b). ◊

A vine $\mathcal{V}$ is a hierarchical collection of $d-1$ undirected trees $T_1, \ldots, T_{d-1}$. Each tree T_i owns a set of nodes N_i and a set of edges E_i, and each edge in each tree will later correspond to a different bivariate copula in the vine factorization. The copulas derived from the edges of the first tree are unconditional. On the other hand, the copulas derived from the edges of the trees $T_2, \ldots, T_{d-1}$ will be conditioned to some variables.

Therefore, the edges of a vine $\mathcal{V}$ specify the factorization of a d-dimensional copula density $c(u_1, \ldots, u_d)$ into the product of bivariate copula densities, that we write using the notation

$$c(u_1, \ldots, u_d) = \prod_{T_i \in \mathcal{V}} \prod_{e_{ij} \in E_i} c_{ij|D_{ij}}(P_{u_{ij}|D_{ij}}(u_{ij}|D_{ij}), P_{v_{ij}|D_{ij}}(v_{ij}|D_{ij})|D_{ij}),$$

where $e_{ij} \in E_i$ is the j-th edge from the i-th tree T_i, corresponding to a bivariate copula linking the two variables u_{ij} and v_{ij} when conditioned to the set of variables D_{ij}. The set of variables $\{u_{ij}, v_{ij}\}$ is *the conditioned set* of e_{ij}, and the set D_{ij} is *the conditioning set* of e_{ij}. The elements of these sets for each edge e_{ij} are constant during the construction of the vine, and detailed in Definition 4.6.

Three rules establish the hierarchical relationships between the trees forming a vine.

Definition 4.5 (Regular vine structure)**.** *The structure of a $d-$dimensional regular vine $\mathcal{V}$ is a sequence of $d-1$ trees $T_1, \ldots, T_{d-1}$ satisfying:*

1. *T_1 has node set $N_1 = \{1, \ldots, d\}$ and edge set E_1.*
2. *T_i has node set $N_i = E_{i-1}$ and edge set E_i, for $2 \leq i \leq d-1$.*
3. *For $\{a, b\} \in E_i$, with $a = \{a_1, a_2\}$ and $b = \{b_1, b_2\}$, it must hold that $\#(a \cap b) = 1$ (proximity condition). That is, the edges a and b must share a common node.*

Define by $C(e_{ij}) := \{u_{ij}, v_{ij}\}$ and $D(e_{ij}) := D_{ij}$ the conditioned and conditioning sets of the edge e_{ij}, respectively. These two sets, for each edge, specify each bivariate copula in the vine factorization. To construct these sets a third and auxiliary set, the *constraint set*, is necessary. In the following definition we show how to obtain the conditioned and conditioning sets in terms of the constraint sets.

Definition 4.6 (Constraint, conditioning and conditioned vine sets)**.** *An edge $e = \{a, b\} \in E_i$, with $a, b \in E_{i-1}$ owns:*

1. *its constraint set*

$$N(e) = \{n \in N_1 : \exists\, e_j \in E_j, j = 1, \ldots, d-1, \\ \textit{with } n \in e_1 \in e_2 \in \ldots \in e\} \subset N_1.$$

2. *its conditioning set* $D(e) = N(a) \cap N(b)$.

3. *its conditioned set* $C(e) = \{N(a) \setminus D(e), N(b) \setminus D(e)\}$.

The constraint set $N(e)$ contains all the nodes in N_1 reachable from nested structure of edges contained in e. For example, consider the edge $e = \{\{1,2\},\{2,3\}\} \in E_2$, with $1,2,3 \in N_1$. Then, e has constraint set $N(e) = \{1,2,3\}$, conditioned set $C(e) = \{1,3\}$ and conditioning set $D(e) = \{2\}$. Therefore, the edge e will later correspond to the bivariate copula $c_{1,3|2}(P_{1|2}(u_{1|2}), P_{3|2}(u_{3|2})|u_2)$ in the resulting vine factorization.

Estimation of vines from data

The structure of a vine is determined by the particular spanning trees chosen at each level of the hierarchy. There exists $\frac{d!}{2}2^{\binom{d-2}{2}}$ different vine structures to model a $d-$dimensional copula function (Kurowicka, 2011). Therefore, to estimate a vine decomposition from data, one must first decide on a particular structure for its trees. One common alternative is to use the greedy algorithm of Dissmann et al. (2013). This algorithm selects maximum spanning trees after giving each edge e a weight corresponding to the empirical estimate of Kendall's τ between each variable in $C(e)$ when conditioned to $D(e)$.

Example 4.2 *(Construction of a four-dimensional regular vine).* Assume access to a sample $U = \{(u_{1,i}, u_{2,i}, u_{3,i}, u_{4,i})\}_{i=1}^n$, drawn iid from some copula $c_{1234}(u_1, u_2, u_3, u_4)$.

1. Our starting point is the four-dimensional complete graph, denoted by G_1. The graph G_1 has set of nodes N_1, one node per random variable $\boldsymbol{u}_i$, and one edge per bivariate unconditional copula $c_{ij}(u_i, u_j)$. See Figure 4.7, left.

2. To construct the first tree in the vine, T_1, we give each edge in G_1 a weight equal to an empirical estimate of Kendall's τ between the variables connected by the edge. For example, we assign the edge e_{11} in G_1 a weight of $\hat{\tau}(u_1, u_3)$. Using the edge weights, we infer the maximum spanning tree T_1. Assume that $E_1 = \{e_{11}, e_{12}, e_{13}\}$ are the edges of the maximum spanning tree T_1. Then

   - $e_{11} = \{1,3\}$ owns $N(e_{11}) = \{1,3\}$, $D(e_{11}) = \emptyset$ and $C(e_{11}) = \{1,3\}$, and produces the copula $c_{13}(u_1, u_3)$ in the vine factorization.

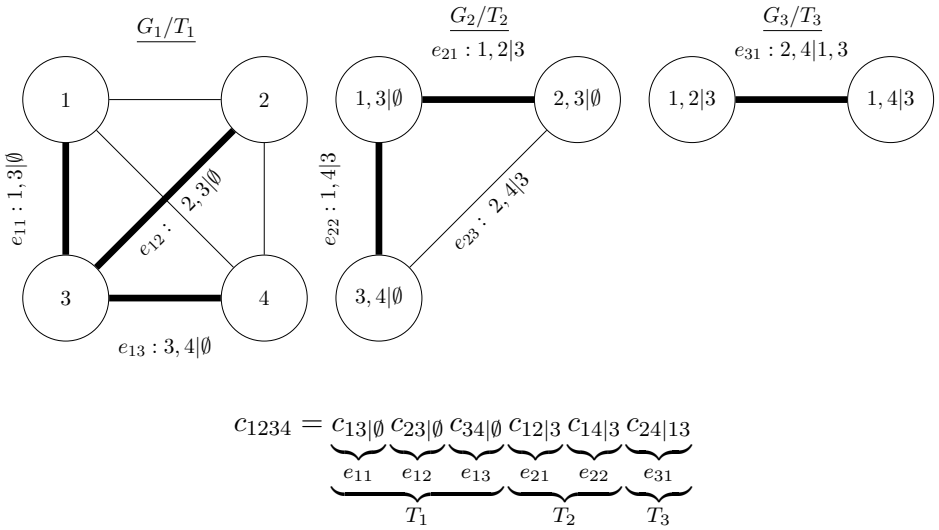


Figure 4.7: Example of the hierarchical construction of a vine factorization of a copula density $c(u_1, u_2, u_3, u_4)$. The edges selected to form each tree are highlighted in bold. Conditioned and conditioning sets for each node and edge are shown as $C(e)|D(e)$.

- $e_{12} = \{2, 3\}$ owns $N(e_{12}) = \{2, 3\}$, $D(e_{12}) = \emptyset$ and $C(e_{12}) = \{2, 3\}$, and produces the copula $c_{23}(u_2, u_3)$ in the vine factorization.
- $e_{13} = \{3, 4\}$ owns $N(e_{13}) = \{3, 4\}$, $D(e_{13}) = \emptyset$ and $C(e_{13}) = \{3, 4\}$, and produces the copula $c_{34}(u_3, u_4)$ in the vine factorization.

The edges in E_1 are highlighted in bold in the left-hand side of Figure 4.7. The parametric copulas c_{13}, c_{23} and c_{34} can belong to any of the families presented in Section 4.4.4, and their parameters can be chosen via maximum likelihood or Kendall's τ inversion on the available data.

3. The next tree T_2 is be the maximum spanning tree of a graph G_2, constructed by following the rules in Definition 4.5. That is, G_2 has node set $N_1 := E_1$ and set of edges formed by pairs of edges in E_1 sharing a common node from N_1.

4. To assign a weight to the edges $\{e_{21}, e_{22}, e_{23}\}$ in G_2, we need samples of the conditional variables $\{\boldsymbol{u}_{1|3}, \boldsymbol{u}_{2|3}, u_{4|3}\}$, where $\boldsymbol{u}_{i|j} = P(\boldsymbol{u}_i|\boldsymbol{u}_j)$. These samples can be obtained using the original sample U and the recursive equation (4.9). For instance, to obtain the samples for $\boldsymbol{u}_{1|3}$,

use

$$u_{1|3,i} = \frac{\partial C_{1,3}(u_{1,i}, u_{3,i})}{\partial u_{3,i}}.$$

Once we have computed the empirical estimate of Kendall's τ on these new conditioned samples, we can assign a weight to each of the edges in G_2 and infer a second maximum spanning tree T_2. Let us assume that $E_2 = \{e_{21}, e_{22}\}$ are the edges forming the maximum T_2 (Figure 4.7, middle).

5. The edges of T_2 represent bivariate *conditional* copulas. Using Definition 4.6, we obtain

   - $e_{21} = \{e_{11}, e_{12}\}$ owns $N(e_{21}) = \{1, 2, 3\}$, $D(e_{21}) = \{3\}$ and $C(e_{21}) = \{1, 2\}$ and produces the copula $c_{1,2|3}(u_{1|3}, u_{2|3})$ in the vine factorization.
   - $e_{22} = \{e_{11}, e_{13}\}$ owns $N(e_{22}) = \{1, 3, 4\}$, $D(e_{22}) = \{3\}$ and $C(e_{22}) = \{1, 4\}$ and produces the copula $c_{1,4|3}(u_{1|3}, u_{4|3})$ in the vine factorization.

6. We repeat this procedure until we have built $d-1$ trees. In our example, we compute a third and last graph G_3, from which we estimate a third and last maximum spanning tree T_3 (Figure 4.7, right-hand side). The corresponding conditional copula ($c_{24|13}$) is the final factor of the overall vine factorization, as depicted in the bottom part of Figure 4.7.

$\Diamond$

Model truncation

In the presence of high-dimensional data, it may be computationally prohibitive to build the $d-1$ trees and $d(d-1)/2$ bivariate copulas that form a complete vine decomposition and specify the full copula density. Similarly, when using finite samples, the curse of dimensionality calls for a large amount of data to efficiently model the higher-order dependencies described in the copulas from the last trees of the factorization.

We can address both of these issues by truncating the vine structure, that is, stopping the construction process after building $d' < d-1$ trees. A truncated vine with d' trees assumes independence in the conditional interactions described by the ignored trees $T_{d'+1}, \dots T_{d-1}$. A truncated vine has a valid density function because the density of the independent copula is constant and equal to one (Equation 4.11). This allows to control the complexity of vine density estimates given a computational budget, dimensionality of the modeled random variable, and size of its sample. This is an attractive property of product models that contrasts mixture models: by structure, mixture models necessarily model all the dependencies at once.

When should we truncate a vine? This is yet another model selection task, which can be addressed by monitoring the log-likelihood on some validation data as we add more trees to the vine hierarchy. For example, we can discard the last built tree if the validation log-likelihood does not improve when adding the corresponding copulas to the vine factorization.

Model limitations and extensions

We now identify two major limitations of vine models, and propose novel solutions to address them based on the material introduced in Sections 4.4.5 and 4.4.6.

Simplification of conditional dependencies Because of the challenges involved in estimating conditional copulas, the literature on vines has systematically ignored the effect of the conditioning variables on the bivariate copulas participating in the vine factorization (Bedford and Cooke, 2001, 2002; Aas et al., 2009; Kurowicka, 2011; Dissmann et al., 2013). This means that the influence of the variables in the conditioning set D_{ij} on each copula $c_{ij|D_{ij}}$ is only incorporated through the conditional cdfs $P_{u_{ij}|D_{ij}}$ and $P_{v_{ij}|D_{ij}}$. That is, the dependence of the copula function $c_{ij|D_{ij}}$ on D_{ij} is ignored. This results in the simplified densities

$$c_{ij|D_{ij}}(P_{u_{ij}|D_{ij}}(u_{ij}|D_{ij}), P_{v_{ij}|D_{ij}}(v_{ij}|D_{ij})|D_{ij}) \approx c_{ij}(P_{u_{ij}|D_{ij}}(u_{ij}|D_{ij}), P_{v_{ij}|D_{ij}}(v_{ij}|D_{ij}))\,.$$

This approximation is the *vine simplifying assumption* (Haff et al., 2010). Acar et al. (2012) argues that this approximation may be too crude when modeling real-world phenomena, and proposes a solution to incorporate the conditioning influence of scalar random variables in the second tree of a vine. However, the question of how to generally describe conditional dependencies across all the trees of a vine remains open.

To address this issue, we propose to model vine conditional dependencies using the novel Gaussian process conditional copulas described in Section 4.4.5. The same ideas apply to the construction of *conditional vine models*, that is, vines conditioned to some set of exogenous variables. In Section 4.6.1 we conduct a variety of experiments that demonstrate the advantages of modeling the previously ignored conditional dependencies in vine decompositions.

Strong parametric assumptions Throughout the literature, vines restrict their bivariate copulas to belong to a parametric family. This has the negative consequence that vines are not universal density estimators like, for example, Gaussian mixture models. To address this issue, we propose to use the described nonparametric bivariate copulas and its novel conditional

distribution rules from Section 4.4.6 to construct more flexible vine distributions. Section 4.6.2 uses the proposed nonparametric vines to address semisupervised domain adaptation problems on a variety of real-world data.

4.6 Numerical simulations

We present two series of numerical experiments. First, we evaluate the improvements obtained by incorporating conditional dependencies into the copulas forming a vine. For this we use the extension proposed in Section 4.4.5. In the second series of experiments, we analyze vine density estimates when we allow nonparametric copulas to participate in the factorization, built as in Section 4.4.6. We illustrate this by using nonparametric vines to address the problem of semisupervised domain adaptation.

4.6.1 Conditional density estimation

We evaluate the performance of the proposed method for the estimation of vine copula densities with full conditional dependencies, as described in Section 4.4.5. Because our method relies on Gaussian processes, we call it GPRV. We compare with two other methods: SRV, a vine model based on the simplifying assumption which ignores conditional dependencies in the bivariate copulas, and NNRV, a vine model based on the nearest-neighbour method of Acar et al. (2012). This latter model can only handle conditional dependencies with respect to a single scalar variable. Therefore, we can only evaluate the performance of NNRV in vine models with two trees, since additional trees would require to account for multivariate conditional dependencies.

In all the experiments, we use 20 pseudo-inputs in the sparse Gaussian process approximation described in Section 4.4.5. The Gaussian processes kernel parameters and pseudo-input locations are tuned using approximate Bayesian model selection, that is, by maximizing the EP estimate of the marginal likelihood. The mean of the GP prior is set to be constant and equal to $\Phi^{-1}((\hat{\tau}_{MLE}+1)/2)$, where $\hat{\tau}_{MLE}$ is the maximum likelihood estimate of τ given the training data. In NNRV, the bandwidth of the Epanechnikov kernel is selected by running a leave-one-out cross validation search using a 30-dimensional log-spaced grid ranging from 0.05 to 10. To simplify the experimental setup, we focus on regular vines formed by bivariate Gaussian copulas. The extension of the proposed approach to incorporate different parametric families of bivariate copulas is straightforward (Hernández-Lobato et al., 2013). We use the empirical copula transformation to obtain data with uniform marginal distributions, as described in Section 4.4.2.

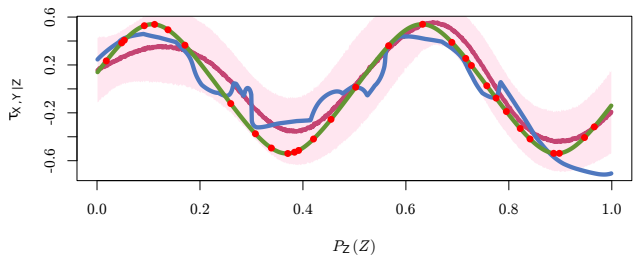


Figure 4.8: In green, the true function g that maps u_3 to τ. In red, the GPRV approximation. In blue, the NNRV approximation. In red, the uncertainty of the GPRV prediction, plus-minus one standard deviation. In red dots, the training samples of u_3.

Synthetic data

We sample synthetic scalar variables $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ according to the following generative process. First, we sample $\boldsymbol{z}$ uniformly from the interval $[-6, 6]$ and second, we sample $\boldsymbol{x}$ and $\boldsymbol{y}$ given $\boldsymbol{z}$ from a bivariate Gaussian distribution with zero mean and covariance matrix given by $\mathrm{Var}(\boldsymbol{x}) = \mathrm{Var}(\boldsymbol{y}) = 1$ and $\mathrm{Cov}(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{z}) = 3/4 \sin(\boldsymbol{z})$. We sample a total of 1000 data points and choose 50 subsamples of size 100 to infer a vine model for the data using SRV, NNRV and GPRV. The first row of the left-hand side of Figure 4.9 shows the average test log-likelihoods on the remaining data points. In these experiments, GPRV shows the best performance.

Figure 4.8 displays the true value of the function g that maps u_3 to the Kendall's τ value of the conditional copula $c_{12|3}(P(u_1|u_3), P(u_2|u_3)|u_3)$, where u_1, u_2 and u_3 are the empirical cumulative probability levels of the samples generated for $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$, respectively. We also show the approximations of g generated by GPRV and NNRV. In this case, GPRV does a better job than NNRV at approximating the true g.

Real data

We further compare the performance of SRV, NNRV and GPRV on an array of real-world datasets. For a detailed description of the datasets, consult Lopez-Paz et al. (2013b). For each dataset, we generate 50 random partitions of the data into training and test sets, each containing half of the available data. Here, each method learns from each training set, and evaluate its log-likelihood on the corresponding test set (higher is better). Table 4.2 shows the test log-likelihood for SRV and GPRV, when using up to T trees in the vine, $1 \leq T \leq d-1$, where d is the number of variables in the data. In general, taking into account conditional dependencies in bivariate copulas leads to superior predictive performance. Also, we often find that improvements

data	SRV	NNRV	GPRV
synthetic	-0.005 ± 0.012	0.101 ± 0.162	$\mathbf{0.298 \pm 0.031}$
uranium	0.006 ± 0.006	0.016 ± 0.026	$\mathbf{0.022 \pm 0.012}$
cloud	8.899 ± 0.334	9.013 ± 0.600	$\mathbf{9.335 \pm 0.348}$
glass	1.206 ± 0.259	0.460 ± 1.996	$\mathbf{1.264 \pm 0.303}$
housing	3.975 ± 0.342	4.246 ± 0.480	$\mathbf{4.487 \pm 0.386}$
jura	2.134 ± 0.164	2.125 ± 0.177	$\mathbf{2.151 \pm 0.173}$
shuttle	2.552 ± 0.273	2.256 ± 0.612	$\mathbf{3.645 \pm 0.427}$
weather	0.789 ± 0.159	0.771 ± 0.890	$\mathbf{1.312 \pm 0.227}$
stocks	$\mathbf{2.802 \pm 0.141}$	2.739 ± 0.155	2.785 ± 0.146

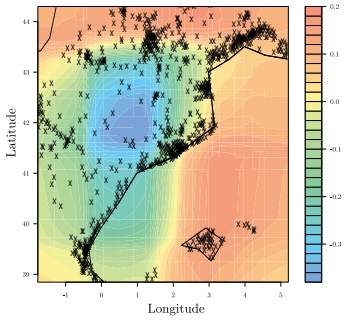


Figure 4.9: Left: Average test log-likelihood and standard deviations for all methods and datasets when limited to 2 trees in the vine (higher is better). Right: Kendall's τ correlation between *atmospheric pressure* and *cloud percentage cover* (color scale) when conditioned to *longitude* and *latitude*.

get larger as we increase the number of trees in the vines. However, in the "stocks" and "jura" datasets the simplifying assumption seems valid. The left-hand side of Figure 4.9 shows a comparison between NNRV and GPRV, when restricted to vines of two trees. In these experiments, NNRV is most of the times outperformed by GPRV. Figure 4.9 shows the use of GPRV to discover scientifically interesting features, revealed by learning spatially varying correlations. In this case, the blue region in the plot corresponds to the Pyrenees mountains. Furthermore, one could examine the learned Gaussian process models to interpret the shape and importance of each of the estimated conditional dependencies.

4.6.2 Vines for semisupervised domain adaptation

We study the use of nonparametric bivariate copulas in single-tree regular vines, using their novel conditional distributions from Section 4.4.6. We call this model Non-Parametric Regular Vine (NPRV). The density estimates in this section are the product of the one-dimensional marginal densities and a vine copula decomposition

$$p(\boldsymbol{x}) = \prod_{i=1}^{d} p_i(x_i) \prod_{T_i \in \mathcal{V}} \prod_{e_{ij} \in E_i} c_{ij}(P_{u_{ij}|D_{ij}}(u_{ij}|D_{ij}), P_{v_{ij}|D_{ij}}(v_{ij}|D_{ij})). \quad (4.25)$$

Remark 4.7 *(Domain adaptation problems).* *Domain adaptation* (Ben-David et al., 2010) aims at transferring knowledge between different but related learning tasks. Generally, the goal of domain adaptation is to improve the learning performance on a *target task*, by using knowledge obtained when solving a different but related *source task*. ◊

In the following, we assume access to large amounts of data sampled from some source distribution p_s. However, a much scarcer sample is available

data	T	SRV	GPRV
cloud	1	**7.860 ± 0.346**	**7.860 ± 0.346**
	2	8.899 ± 0.334	**9.335 ± 0.348**
	3	9.426 ± 0.363	**10.053 ± 0.397**
	4	9.570 ± 0.361	**10.207 ± 0.415**
	5	9.644 ± 0.357	**10.332 ± 0.440**
	6	9.716 ± 0.354	**10.389 ± 0.459**
	7	9.783 ± 0.361	**10.423 ± 0.463**
	8	9.790 ± 0.371	**10.416 ± 0.459**
	9	9.788 ± 0.373	**10.408 ± 0.460**
glass	1	**0.827 ± 0.150**	**0.827 ± 0.150**
	2	1.206 ± 0.259	**1.264 ± 0.303**
	3	1.281 ± 0.251	**1.496 ± 0.289**
	4	1.417 ± 0.251	**1.740 ± 0.308**
	5	1.493 ± 0.291	**1.853 ± 0.318**
	6	1.591 ± 0.301	**1.936 ± 0.325**
	7	1.740 ± 0.282	**2.000 ± 0.345**
	8	1.818 ± 0.243	**2.034 ± 0.343**
jura	1	**1.887 ± 0.153**	**1.887 ± 0.153**
	2	2.134 ± 0.164	**2.151 ± 0.173**
	3	2.199 ± 0.151	**2.222 ± 0.173**
	4*	2.213 ± 0.153	**2.233 ± 0.181**
	5*	2.209 ± 0.153	**2.215 ± 0.185**
	6*	**2.213 ± 0.155**	2.197 ± 0.189
shuttle	1	**1.487 ± 0.256**	**1.487 ± 0.256**
	2	2.188 ± 0.314	**2.646 ± 0.349**
	3	2.552 ± 0.273	**3.645 ± 0.427**
	4	2.782 ± 0.284	**4.204 ± 0.551**
	5	3.092 ± 0.353	**4.572 ± 0.567**
	6	3.284 ± 0.325	**4.703 ± 0.492**
	7	3.378 ± 0.288	**4.763 ± 0.408**
	8	3.417 ± 0.257	**4.761 ± 0.393**
	9	3.426 ± 0.252	**4.755 ± 0.389**

data	T	SRV	GPRV
weather	1	**0.684 ± 0.128**	**0.684 ± 0.128**
	2	0.789 ± 0.159	**1.312 ± 0.227**
	3	0.911 ± 0.178	**2.081 ± 0.341**
	4	1.017 ± 0.184	**2.689 ± 0.368**
	5	1.089 ± 0.188	**3.078 ± 0.423**
	6	1.138 ± 0.181	**3.326 ± 0.477**
	7	1.170 ± 0.169	**3.473 ± 0.467**
	8	1.177 ± 0.170	**3.517 ± 0.465**
stocks	1	**2.776 ± 0.142**	2.776 ± 0.142
	2*	**2.799 ± 0.142**	2.785 ± 0.146
	3	**2.801 ± 0.142**	2.764 ± 0.151
	4	**2.802 ± 0.143**	2.742 ± 0.158
	5	**2.802 ± 0.141**	2.721 ± 0.159
housing	1	**3.409 ± 0.354**	**3.409 ± 0.354**
	2	3.975 ± 0.342	**4.487 ± 0.386**
	3	4.128 ± 0.363	**4.953 ± 0.425**
	4	4.250 ± 0.376	**5.307 ± 0.458**
	5	4.386 ± 0.380	**5.541 ± 0.498**
	6	4.481 ± 0.399	**5.691 ± 0.516**
	7	4.576 ± 0.422	**5.831 ± 0.529**
	8	4.666 ± 0.412	**5.934 ± 0.536**
	9	4.768 ± 0.399	**6.009 ± 0.516**
	10	4.838 ± 0.382	**6.084 ± 0.520**
	11	4.949 ± 0.362	**6.113 ± 0.525**

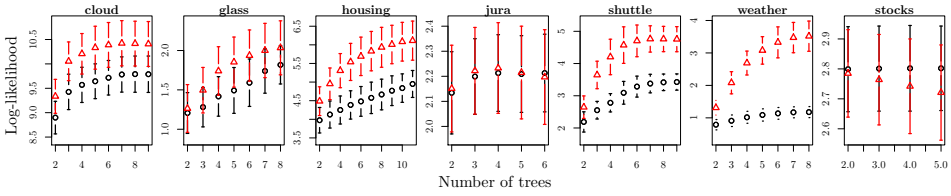


Table 4.2: Top: Average test log-likelihood and standard deviations for SRV and GPRV on real-world datasets (higher is better). Asterisks denote results not statistically significant with respect to a paired Wilcoxon test with p–value = 10^{-3}. Same information depicted as a plot; red triangles correspond to GPRV, black circles to SRV. For results comparing GPRV to NNRV, see Table 4.9.

to estimate the target density p_t. Given the data available for both tasks, our objective is to build a good estimate for the density p_t. To do so, we assume that p_t is a modified version of p_s. In particular, we assume that the transformation from p_s to p_t takes two steps. First, p_s follows a vine factorization, as in Equation 4.25. Second, we modify a small subset of the factors in p_s, either marginals or bivariate copulas, to obtain p_t. This is equivalent to assuming that only a small amount of marginal distributions or dependencies in the joint distribution change across domains, while the structure of the trees forming the vine remains constant.

All we need to address the adaptation across domains is to reconstruct the vine representation of p_s using data from the source task, and then identify which of the factors forming p_s changed to produce p_t. These factors are re-estimated using data from the target task, when available. Note that this is a general domain adaptation strategy (subsuming *covariate shift*, among others), and works in the unsupervised or semisupervised scenario (where target data has missing variables, do not update factors related to those variables). To decide whether to re-estimate a given univariate marginal or bivariate copula when adapting p_s to p_t, we use the *Maximum Mean Discrepancy* test, or MMD (Gretton et al., 2012a).

We analyze NPRV in a series of domain adaptation nonlinear regression problems on real data. For a more detailed description about the experimental protocol and datasets, consult the supplementary material of Lopez-Paz et al. (2012). During our experiments, we compare NPRV with different benchmark methods. The first two methods, GP-Source and GP-All, are baselines. They are two Gaussian Process (GP) methods, the first one trained only with data from the source task, and the second one trained with the normalized union of data from both source and target problems. The other five methods are state-of-the-art domain adaptation techniques: including Daume (Daumé III, 2009), SSL-Daume (Daumé III et al., 2010), ATGP (Cao et al., 2010), Kernel Mean Matching or KMM (Huang et al., 2006), and Kernel unconstrained Least-Squares Importance Fitting or KuLSIF (Kanamori et al., 2012). Besides NPRV, we also include in the experiments its unsupervised variant, UNPRV, which ignores any labeled data from the target task and adapts only vine factors depending on the input features. For training, we randomly sample 1000 data points for both source and target tasks, where all the data in the source task and 5% of the data in the target task have labels. The test set contains 1000 points from the target task. Table 4.3 summarizes the average test normalized mean square error (NMSE) and corresponding standard deviations for each method in each dataset across 30 random repetitions of the experiment. The proposed methods obtain the best results in 5 out of 6 cases. The two last two rows in Table 4.3 show the average number of factors (marginals or bivariate copulas) updated from source to target task, according to the MMD test.

data	**wine**	**sarcos**	**rocks-mines**	**hill-valleys**	**axis-slice**	**isolet**
No. of variables	**12**	**21**	**60**	**100**	**386**	**617**
GP-Source	0.86 ± 0.02	1.80 ± 0.04	0.90 ± 0.01	1.00 ± 0.00	1.52 ± 0.02	1.59 ± 0.02
GP-All	0.83 ± 0.03	1.69 ± 0.04	1.10 ± 0.08	0.87 ± 0.06	1.27 ± 0.07	1.58 ± 0.02
Daume	0.97 ± 0.03	0.88 ± 0.02	0.72 ± 0.09	0.99 ± 0.03	0.95 ± 0.02	0.99 ± 0.00
SSL-Daume	0.82 ± 0.05	0.74 ± 0.08	0.59 ± 0.07	0.82 ± 0.07	0.65 ± 0.04	0.64 ± 0.02
ATGP	0.86 ± 0.08	0.79 ± 0.07	**0.56 ± 0.10**	0.15 ± 0.07	1.00 ± 0.01	1.00 ± 0.00
KMM	1.03 ± 0.01	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
KuLSIF	0.91 ± 0.08	1.67 ± 0.06	0.65 ± 0.10	0.80 ± 0.11	0.98 ± 0.07	0.58 ± 0.02
NPRV	**0.73 ± 0.07**	**0.61 ± 0.10**	0.72 ± 0.13	**0.15 ± 0.07**	0.38 ± 0.07	0.46 ± 0.09
UNPRV	0.76 ± 0.06	0.62 ± 0.13	0.72 ± 0.15	0.19 ± 0.09	**0.37 ± 0.07**	**0.42 ± 0.04**
Av. Ch. Mar.	10	1	38	100	226	89
Av. Ch. Cop.	5	8	49	34	155	474

Table 4.3: NMSE for all domain adaptation algorithms and datasets.

Chapter 5

Discriminative dependence

This chapter contains novel material. Section 5.1 presents a framework for nonlinear component analysis based on random features (Lopez-Paz et al., 2014), called Randomized Component Analysis (RCA). We exemplify RCA by proposing Randomized Principal Component Analysis (RPCA, Section 5.1.1), and Randomized Canonical Correlation Analysis (RCCA, Section 5.1.2). Based on RCA and the theory of copulas, we introduce a measure of dependence termed the Randomized Dependence Coefficient (RDC, Section 5.2.3, Lopez-Paz et al. (2013a)). We give theoretical guarantees for RPCA, RCCA, and RDC by using recent matrix concentration inequalities. We illustrate the effectiveness of the proposed methods in a variety of numerical simulations (Section 5.4).

The previous chapter studied generative models of dependence: those that estimate the entire dependence structure of some multivariate data, and are able to synthesize new samples from the data generating distribution. We have seen that generative modeling is intimately linked to density estimation, which is a general but challenging learning problem. General, because one can solve many other tasks of interest, such as regression and classification, as a byproduct of density estimation. Challenging, because it requires the estimation of all the information contained in data.

Nevertheless, in most situations we are not interested in describing the whole dependence structure cementing the random variables under study, but in summarizing some particular aspects of it, which we believe useful for subsequent learning tasks. Let us give three examples. First, *component analysis* studies how to boil down the variables in some high-dimensional data to a small number of explanatory components. These explanatory components throw away some of the information from the original data, but retain directions containing most of the variation in data. Second, *dependence measurement*, given two random variables, quantifies to what degree they depend on each other. Third, *two-sample-testing* asks: given two random samples, were they drawn from the same distribution? These three tasks do

not require estimating the density of the data in its entirety. Instead, these *discriminative dependence methods* summarize the dependence structure of a multivariate data set into a low-dimensional statistic that answers the question at hand.

Let us examine the state-of-the-art more concretely. Two of the most popular discriminative dependence methods are Principal Component Analysis (PCA) by Pearson (1901) and Canonical Correlation Analysis (CCA) by Hotelling (1936). Both have played a crucial role in multiple applications since their conception over a century ago. Despite their great successes, an impediment of these classical discriminative methods for modern data science is that they only reveal linear relationships between the variables under study. But linear component analysis methods, such as PCA and CCA, operate in terms of inner products between the examples contained in the data at hand. This makes kernels one elegant way to extend these algorithms to capture nonlinear dependencies. Examples of these extensions are Kernel PCA or KPCA (Schölkopf et al., 1997), and Kernel CCA or KCCA (Lai and Fyfe, 2000; Bach and Jordan, 2002). Unfortunately, when working on n data, kernelized discriminative methods require the construction and inversion of $n \times n$ kernel matrices. Performing these operations takes $O(n^3)$ time, a prohibitive computational requirement when analyzing large data.

In this chapter, we propose the use of random features (Section 3.2.2) to overcome the limits of linear component analysis algorithms and the computational burdens of their kernelized extensions. We exemplify the use of random features in three discriminative dependence tasks: component analysis (Section 5.1), dependence measurement (Section 5.2), and two-sample testing (Section 5.3). The algorithms presented in this chapter come with learning rates and consistency guarantees (Section 5.5), as well as a performance evaluation on multiple applications and real-world data (Section 5.4). Since our framework and its extensions are based on random features, we call it Randomized Component Analysis, or RCA.

Before we start, let us introduce the main actors of this chapter in the following definition.

Definition 5.1 (Assumptions on discriminative dependence)**.** *As usual, we consider data* $\{x_1, \ldots, x_n\} \sim P^n(\boldsymbol{x})$*, where* $x_i \in \mathcal{X}$ *for all* $1 \leq i \leq n$*. Using this data, the central object of study throughout this chapter is the spectral norm* $\|\hat{K} - K\|$*, where* $K \in \mathbb{R}^{n \times n}$ *is a full-rank kernel matrix, and* $\hat{K} \in \mathbb{R}^{n \times n}$ *is a rank-*m *approximation of* K *(Section 2.1.2). The full-rank kernel matrix* K *has entries* $K_{i,j} = k(x_i, x_j)$*, where* k *is a real-valued, shift-invariant* $(k(x, x') = k(x - x', 0))$*, and* L_k*-Lipschitz kernel,*

$$|k(\delta, 0) - k(\delta', 0)| \leq L_k \|\delta - \delta'\|,$$

also satisfying the boundedness *condition* $|k(x, x')| \leq 1$ *for all* $x, x' \in \mathcal{X}$*. On*

the other hand, the approximate kernel matrix $\hat{K}$ *is*

$$z_i = \sqrt{\frac{2}{m}} \left(\cos(\langle w_i, x_1\rangle + b_i), \ldots, \cos(\langle w_i, x_n\rangle + b_i)\right)^\top \in \mathbb{R}^n, \tag{5.1}$$
$$\hat{K}_i = z_i z_i^\top,$$
$$\hat{K} = \frac{1}{m}\sum_{i=1}^{m} z_i z_i^\top,$$

where $z_i \in \mathbb{R}^{n\times 1}$*, with* $\|z_i\|^2 \leq B$*, is the* i*-th random feature of the* n *examples contained in our training data (Section 3.2.2). We call* $Z \in \mathbb{R}^{n\times m}$ *the matrix with rows* $z_1, \ldots, z_m$*.*

Finally, some parts of this chapter will consider data from two random variables

$$\{(x_1, y_1), \ldots, (x_n, y_n)\} \sim P^n(\boldsymbol{x}, \boldsymbol{y}).$$

When this is the case, we will build full rank kernel matrices K_x *and* K_y *for each of the two random variables, and their respective rank-*m *approximations* $\hat{K}_x$ *and* $\hat{K}_y$*.*

5.1 Randomized Component analysis

Component analysis relates to the idea of *dimensionality reduction*: summarizing a large set of variables into a small set of factors able to explain key properties about the original variables. Some examples of component analysis algorithms include "principal component analysis, factor analysis, linear multidimensional scaling, Fishers linear discriminant analysis, canonical correlations analysis, maximum autocorrelation factors, slow feature analysis, sufficient dimensionality reduction, undercomplete independent component analysis, linear regression, and distance metric learning", all of these eloquently reviewed in (Cunningham and Ghahramani, 2015). Component analysis is tightly related to transformation generative models, in the sense that both methods aim at extracting a set of explanatory factors from data. Dimensionality reduction algorithms differ in what they call "the important information to retain about the original data". During the remainder of this section, we review two of the most widely used linear dimensionality reduction methods, PCA and CCA, and extend them to model nonlinear dependencies in a theoretically and computationally sustained way.

Remark 5.1 *(Prior work on randomized component analysis)***.** Achlioptas et al. (2002) pioneered the use of randomized techniques to approximate kernelized component analysis, by suggesting three sub-sampling strategies to speed up KPCA. Avron et al. (2014) used randomized Walsh-Hadamard transforms to adapt linear CCA to large datasets. McWilliams et al. (2013) applied the Nyström method to CCA on the problem of semisupervised learning. ◊

5.1.1 Principal component analysis

Principal Component Analysis or PCA (Pearson, 1901) is the orthogonal transformation of a set of n observations of d variables $X \in \mathbb{R}^{n \times d}$ into a set of n observations of d uncorrelated *principal components* XF (also known as factors or latent variables). Principal components owe their name to the following property: the first principal component captures the maximum amount of variations due to linear relations in the data; successive components account for the maximum amount of remaining variance in dimensions orthogonal to the preceding ones. PCA is commonly used for dimensionality reduction, assuming that the $d' < d$ principal components capture the core properties of the data under study. For a centered matrix of n samples and d dimensions $X \in \mathbb{R}^{n \times d}$, PCA requires computing the singular value decomposition $X = U\Sigma F'$ (Section 2.1.2). The top d' principal components are $XF_{1:d',:}$, where $F_{1:d',:}$ denotes the first d' rows of F. PCA seeks to retain linear variations in the data, in the sense that it minimizes the reconstruction error of the linear transformation from the d' principal components back to the original data.

Remark 5.2 *(History of PCA).* Principal component analysis was first formulated over a century ago by Pearson (1901). The method was independently discovered and advanced to its current form by Hotelling (1933), who is also responsible for coining the term *principal components*. PCA has found a wide range of successful applications, including finance, chemistry, computer vision, neural networks, and biology, to name some. Jolliffe (2002) offers a modern account on PCA and its applications. ◊

One of the limitations of the PCA algorithm is that the recovered principal components can only account for linear variations in data. This is a limiting factor, as it may be the case that there exists interesting nonlinear patterns hidden in the data, not contributing to the linear variance that PCA seeks to retain. To address these limitations, Schölkopf et al. (1997) introduced Kernel PCA or KPCA, an algorithm that leverages the kernel trick (Section 3.1) to extract linear components in some high-dimensional and nonlinear representation of the data. Computationally speaking, KPCA performs the eigendecomposition of a $n \times n$ kernel matrix when analyzing data sets of n examples. Unfortunately, these operations require $O(n^3)$ computation, a prohibitive complexity for large data. But there is hope: in words of Joel Tropp, "large data sets tend to be redundant, so the kernel matrix also tends to be redundant. This manifests in the kernel matrix being close to a low-rank matrix". This quote summarizes the motivation of RPCA, the first example of the RCA framework, proposed next.

To extend the PCA algorithm to discover nonlinear principal components, while avoiding the computational burden of KPCA, we propose Randomized PCA or RPCA (Lopez-Paz et al., 2014). In particular, RPCA proceeds by

1. maps the original data $X := (x_1, \ldots, x_n) \in \mathbb{R}^{n \times d}$, into the random feature data $Z := (z_1, \ldots, z_n) \in \mathbb{R}^{n \times m}$.

2. performs PCA on the data Z.

Therefore, RPCA approximates KPCA when the random features used by the former approximate the kernel function used by the latter. The principal components obtained with RPCA are no longer linear transformations of the data, but approximations to nonlinear transformations of the data living in the reproducing kernel Hilbert Space $\mathcal{H}$. Computationally, approximating the covariance matrix of $Z \in \mathbb{R}^{n \times m}$ dominates the time complexity of RPCA. This operation has a time complexity $O(m^2 n)$ in the typical regime $n \gg m$, which is competitive with the linear PCA complexity $O(d^2 n)$.

Since RPCA approximates KPCA, and the solution of KPCA relates to the spectrum of K, we will study the convergence rate of $\hat{K}$ to K in *operator norm* as m grows (see Definition 5.1). A bound about $\|\hat{K} - K\|$ is quite valuable to our purposes: such bound simultaneously controls the error in every linear projection of the approximation, that is:

$$\|\hat{K} - K\| \leq \varepsilon \Rightarrow |\mathrm{tr}(\hat{K}X) - \mathrm{tr}(KX)| \leq \varepsilon,$$

where $\|X\|_{S_1} \leq 1$ and $\|\cdot\|_{S_1}$ is the Schatten 1-norm (Tropp, 2015). Such bound also controls the whole spectrum of singular values of our approximation $\hat{K}$, that is:

$$\|\hat{K} - K\| \leq \varepsilon \Rightarrow |\sigma_j(\hat{K}) - \sigma_j(K)| \leq \varepsilon,$$

for all $j = 1, \ldots, n$ (Tropp, 2015).

Theorem 5.1 (Convergence of RPCA)**.** *Consider the assumptions from Definition 5.1. Then,*

$$\mathbb{E}\Big[\|\hat{\boldsymbol{K}} - K\|\Big] \leq \sqrt{\frac{3B\|K\| \log n}{m}} + \frac{2B \log n}{m}. \tag{5.2}$$

Proof. See Section 5.5.1. □

Theorem 5.1 manifests that RPCA approximates KPCA with a small amount of random features whenever the intrinsic dimensionality $n/\|K\|$ of the exact kernel matrix K is small.

Remark 5.3 *(Similar algorithms to RPCA)***.** Spectral clustering uses the spectrum of K to perform dimensionality reduction before applying k-means (Von Luxburg, 2007). Therefore, the analysis of RPCA inspires a randomized and nonlinear variant of spectral clustering. ◊

Remark 5.4 *(Compression and intelligence)***.** Dimensionality reduction, and more generally unsupervised learning, relates to data *compression*. In fact,

compression is possible because of the existence of patterns in data, as these patterns allow to recover some variables from others. In the absence of patterns, data would be independent noise, and the best compression would be the data itself. Compression amounts to finding the simplest descriptions of objects, a task considered intimate to intelligence. ◊

5.1.2 Canonical correlation analysis

Canonical Correlation Analysis or CCA (Hotelling, 1936) estimates the correlation between two multidimensional random variables. Given two paired samples $X \in \mathbb{R}^{n\times p}$ and $Y \in \mathbb{R}^{n\times q}$, CCA computes pairs of *canonical bases* $f_i \in \mathbb{R}^p$ and $g_i \in \mathbb{R}^q$ such that they maximize the correlation between the transformed samples Xf_i and Yg_i, for all $1 \leq i \leq \min(p, q)$. Graphically, CCA finds a pair of linear transformations, XF from X and YG from Y, such that the dimensions of XF and YG (also known as canonical variables) are maximally correlated. This is an useful manipulation when we learn from two different views of the same data. Consider for instance of document translation (Vinokourov et al., 2002), where the training data is a collection of documents in two different languages, let us say English and Spanish. In this task, we could use CCA to transform the documents into a representation that correlates their English version and their Spanish version, and then exploit these correlations to predict translations.

More formally, let C_{xy} be the empirical covariance matrix between X and Y. Thus CCA maximizes

$$\rho_i^2 := \rho^2(Xf_i, Yg_i) = \frac{f_i^\top C_{xy} g_i}{\sqrt{f_i^\top C_{xx} f_i}\sqrt{g_i^\top C_{yy} g_i}},$$

for $1 \leq i \leq r = \min(\text{rank}(X), \text{rank}(Y))$, subject to

$$\rho^2(Xf_i, Yg_j) = \rho^2(Xf_i, Xf_j) = \rho^2(Yg_i, Yg_j) = 0,$$

for all $i \neq j$ and $1 \leq j \leq r$. We call the quantities ρ_i^2 the *canonical correlations*. Analogous to principal components, we order the *canonical variables* (Xf_i, Yg_i) with respect to their cross-correlation, that is, $\rho_1^2 \geq \cdots \geq \rho_r^2$. The canonical correlations $\rho_1^2, \ldots, \rho_r^2$ and canonical bases $f_1, \ldots, f_r \in \mathbb{R}^p$, $g_1, \ldots, g_r \in \mathbb{R}^q$ are the solutions of the generalized eigenvalue problem (Bach and Jordan, 2002, Equation (2)):

$$\begin{pmatrix} 0 & C_{xy} \\ C_{yx} & 0 \end{pmatrix} \begin{pmatrix} f \\ g \end{pmatrix} = \rho^2 \begin{pmatrix} C_{xx} & 0 \\ 0 & C_{yy} \end{pmatrix} \begin{pmatrix} f \\ g \end{pmatrix},$$

Said differently, CCA processes two different views of the same data (speech audio signals and paired speaker video frames) and returns their maximally correlated linear transformations. This is particularly useful when

the two views of the data are available at training time, but only one of them is available at test time (Kakade and Foster, 2007; Chaudhuri et al., 2009; Vapnik and Vashist, 2009).

Remark 5.5 *(History of CCA).* Hotelling (1936) introduced CCA to measure the correlation between two multidimensional random variables. CCA has likewise found numerous applications, including multi-view statistics (Kakade and Foster, 2007) and learning with missing features (Chaudhuri et al., 2009; Lopez-Paz et al., 2014). Hardoon et al. (2004) offers a monograph on CCA with a review on applications. ♢

The main limitation of the CCA algorithm is that the recovered canonical variables only extract linear patterns in the analyzed pairs of data. To address these limitations, Lai and Fyfe (2000); Bach and Jordan (2002) introduce Kernel CCA or KCCA, an algorithm that leverages the kernel trick (Section 3.1) to extract linear components in some high-dimensional nonlinear representation of the data. Computationally speaking, KCCA performs the eigendecomposition of the matrix

$$M := \begin{pmatrix} I & M_{12} \\ M_{21} & I \end{pmatrix}, \tag{5.3}$$

where

$$M_{12} = (K_x + n\lambda I)^{-1} K_x K_y (K_y + n\lambda I)^{-1},$$
$$M_{21} = (K_y + n\lambda I)^{-1} K_y K_x (K_x + n\lambda I)^{-1},$$

and $\lambda > 0$ is a regularization parameter necessary to avoid spurious perfect correlations (Bach and Jordan, 2002, Equation (16)). Unfortunately, these operations require $O(n^3)$ computations, a prohibitive complexity for large data.

To extend the CCA algorithm to extract nonlinear canonical variables while avoiding the computational burdens of KCCA, we propose RCCA (Lopez-Paz et al., 2014), the second example of our framework RCA. In particular, RCCA

1. maps the original data $X := (x_1, \ldots, x_n) \in \mathbb{R}^{n \times p}$, into the random feature data $Z_x := (z_1^{(x)}, \ldots, z_n^{(x)}) \in \mathbb{R}^{n \times m_x}$,
2. maps the original data $Y := (y_1, \ldots, y_n) \in \mathbb{R}^{n \times q}$, into the randomized feature vectors $Z_y := (z_1^{(y)}, \ldots, z_n^{(y)}) \in \mathbb{R}^{n \times m_y}$ and
3. performs CCA on the pair of datasets Z_x and Z_y.

Thus, RCCA approximates KCCA when the random features of the former approximate the kernel function of the latter. The canonical variables in RCCA are no longer linear transformations of the original data; they are

approximations of nonlinear transformations living in the Hilbert Spaces $(\mathcal{H}_x, \mathcal{H}_y)$, induced by the kernels k_x and k_y. The computational complexity of RCCA is $O((m_x^2 + m_y^2)n)$, which is competitive when compared to the computational complexity $O((p^2+q^2)n)$ of linear CCA for a moderate number of random features.

As with PCA, we will study the convergence rate of RCCA to KCCA in operator norm, as m_x and m_y grow. Let $\hat{K}_x$ and $\hat{K}_y$ be the approximations to the kernel matrices K_x and K_y, obtained by using m_x and m_y random features on the data X and Y as in Definition 5.1. Then, RCCA approximates the KCCA matrix (5.3) with

$$\hat{M} := \begin{pmatrix} I & \hat{M}_{12} \\ \hat{M}_{21} & I \end{pmatrix}. \tag{5.4}$$

where

$$\hat{M}_{12} = (\hat{K}_x + n\lambda I)^{-1}\hat{K}_x\hat{K}_y(\hat{K}_y + n\lambda I)^{-1},$$
$$\hat{M}_{21} = (\hat{K}_y + n\lambda I)^{-1}\hat{K}_y\hat{K}_x(\hat{K}_x + n\lambda I)^{-1}.$$

The solution of RCCA is the eigendecomposition of (5.4). The following theorem allows to phrase the convergence rate of RCCA to KCCA, as a function of the convergence rate of RPCA to KPCA.

Theorem 5.2 (Norm of kernel matrices bound norm of CCA)**.** *Let M and $\hat{M}$ be as in Equations* (5.3) *and* (5.4)*, respectively. Then,*

$$\|\hat{M} - M\| \leq \left\{\frac{3}{n}\left(\frac{1}{\lambda^2} + \frac{1}{\lambda}\right)\right\}\left(\|\hat{K}_x - K_x\| + \|K_y - \hat{K}_y\|\right) \leq 1.$$

Proof. See Section 5.5.2. □

The following result characterizes the convergence rate of RCCA to KCCA, in terms of the number of random features m.

Corollary 5.1 (Convergence of RCCA)**.** *Consider the assumptions from Definition 5.1, and Equations 5.3-5.4. Then,*

$$\mathbb{E}\left[\|\hat{\boldsymbol{M}} - M\|\right] \leq \left\{\frac{6}{n}\left(\frac{1}{\lambda^2} + \frac{1}{\lambda}\right)\right\}\left(\sqrt{\frac{3B\|K\|\log n}{m}} + \frac{2B\log n}{m}\right) \leq 1, \tag{5.5}$$

where $\|K\| = \max(\|K_x\|, \|K_y\|)$, and $m = \min(m_x, m_y)$.

Proof. Combine Theorem 5.1 and Theorem 5.2. □

Remark 5.6 *(Similar algorithms to RCCA)***.** *Linear discriminant analysis* seeks a linear combination of the features of the data $X \in \mathbb{R}^{n\times d}$ such that the samples become maximally separable with respect to a paired labeling y

with $y_i \in \{1, \dots, c\}$. LDA solves CCA(X, T), where $T_{ij} = \mathbb{I}\{y_i = j\}$ (De Bie et al., 2005). Therefore, a similar analysis to the one of RCCA applies to study randomized and nonlinear variants of LDA. $\lozenge$

Remark 5.7 *(Extensions and improvements to component analysis).* Component analysis reduces our data into a number of explanatory factors smaller than the number of original variables. However, in some situations it may be the case that the observed variables are the summary of a larger number of explanatory factors. In this case, component analysis would aim at estimating a number of explanatory factors larger than the number of observed variables. We refer to this kind of component analysis as *overcomplete component analysis*. Random features allow overcomplete component analysis, by setting $m \gg d$, and regularizing properly.

There are two straightforward ways to improve the speed of the algorithms presented in this chapter. First, distributing the computation of the random feature covariance matrices over multiple processing units. Second, using computing only the top k singular values of the random feature matrix Z, if we are only interested in extracting the top $k \ll m$ components. $\lozenge$

5.2 Measures of dependence

Measuring the extent to which two random variables depend on each other is a fundamental question in statistics and applied sciences (*What genes are responsible for a particular phenotype?*). Mathematically, given the sample

$$\mathcal{Z} = \{(x_i, y_i)\}_{i=1}^n \sim P^n(\boldsymbol{x}, \boldsymbol{y}),$$

the question of whether the two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ are independent is to estimate whether the equality

$$p(x, y) = p(x)p(y)$$

holds for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. One way to prove that two random variables are independent is to check if their mutual information

$$I(\boldsymbol{x}, \boldsymbol{y}) = \int_{\mathcal{Y}} \int_{\mathcal{X}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \mathrm{d}x\mathrm{d}y.$$

is zero. Unfortunately, estimating the mutual information is a challenging task, since it requires the estimation of the densities $p(\boldsymbol{x})$, $p(\boldsymbol{y})$, and $p(\boldsymbol{x}, \boldsymbol{y})$.

Let us take one step back, and simplify the problem by asking if the random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ are *correlated*, that is, related by a *linear* dependence. Answering this question is much simpler; two random variables are not correlated if and only if their Pearson's correlation coefficient

$$\hat{\rho}(\{(x_i, y_i)\}_{i=1}^n) = \frac{\sum_{i=1}^n (x_i - \hat{\mu}_x)(y_i - \hat{\mu}_y)}{\sqrt{\sum_{i=1}^n (x_i - \hat{\mu}_x)} \sqrt{\sum_{i=1}^n (y_i - \hat{\mu}_y)}}$$

converges to zero, as $n \to \infty$, where $\hat{\mu}_x = n^{-1} \sum_i x_i$, and similarly for $\hat{\mu}_y$.

Correlation measures to what extent the relationship between two variables is a *straight line*. There are multiple ways to extend the concept of correlation to slightly more general situations. For example, Spearman's ρ and Kendall's τ measure to what extent we can express the relationship between two random variables as a *monotone* function. But what should a general measure of dependence satisfy?

5.2.1 Renyi's axiomatic framework

Half a century ago, Alfréd Rényi (1959) argued that a general measure of dependence $\rho^* : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ between two nonconstant random variables $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$ should satisfy seven fundamental properties:

1. $\rho^*(\boldsymbol{x}, \boldsymbol{y})$ is defined for any pair of random variables $\boldsymbol{x}$ and $\boldsymbol{y}$.
2. $\rho^*(\boldsymbol{x}, \boldsymbol{y}) = \rho^*(\boldsymbol{y}, \boldsymbol{x})$
3. $0 \leq \rho^*(\boldsymbol{x}, \boldsymbol{y}) \leq 1$
4. $\rho^*(\boldsymbol{x}, \boldsymbol{y}) = 0$ iff $\boldsymbol{x}$ and $\boldsymbol{y}$ are statistically independent.
5. For bijective Borel-measurable $f, g : \mathbb{R} \to \mathbb{R}$, $\rho^*(\boldsymbol{x}, \boldsymbol{y}) = \rho^*(f(\boldsymbol{x}), g(\boldsymbol{y}))$.
6. $\rho^*(\boldsymbol{x}, \boldsymbol{y}) = 1$ if for Borel-measurable f or g, $\boldsymbol{y} = f(\boldsymbol{x})$ or $\boldsymbol{x} = g(\boldsymbol{y})$.
7. If $(\boldsymbol{x}, \boldsymbol{y}) \equiv \mathcal{N}(\mu, \Sigma)$, then $\rho^*(\boldsymbol{x}, \boldsymbol{y}) = |\rho(\boldsymbol{x}, \boldsymbol{y})|$, where ρ is the correlation coefficient.

In the same work, Rényi also showed that the *Hirschfeld-Gebelein-Rényi Maximum Correlation Coefficient* (HGR) satisfies all these properties. HGR, introduced by Gebelein (1941) is the suprema of Pearson's correlation coefficient ρ over all Borel-measurable functions f, g of finite variance:

$$\mathrm{HGR}(\boldsymbol{x}, \boldsymbol{y}) = \sup_{f,g} \rho(f(\boldsymbol{x}), g(\boldsymbol{y})). \tag{5.6}$$

Unfortunately, the suprema in (5.6) is NP-hard to compute. In the following, we review different computable alternatives to approximate the HGR statistic.

5.2.2 Kernel measures of dependence

Kernel measures of dependence measure the dependence between two random variables as their correlation when mapped to some RKHS. This is just another clever use of the kernel trick: in Chapter 3, kernels allowed us to phrase nonlinear regression as linear regression in RKHS (recall Example 3.1). In this section, they will allow us to phrase dependence as correlation in RKHS.

In the following, consider the kernel matrix $K \in \mathbb{R}^{n \times n}$ with entries $K_{i,j} = k_x(x_i, x_j)$, and the kernel matrix $L \in \mathbb{R}^{n \times n}$ with entries $L_{i,j} = k_y(y_i, y_j)$, for all $1 \leq i \leq n$ and $1 \leq j \leq n$. Also, consider the centered kernel matrices $\tilde{K} = HKH$ and $\tilde{L} = HLH$, built using the centering matrix $H = I_n - n^{-1} 1_n 1_n^\top$. We review three important measures of dependence based on kernels.

First, the COnstrained COvariance or COCO (Gretton et al., 2005a) is the largest singular value of the cross-covariance operator associated with the reproducing kernel Hilbert spaces $\mathcal{H}_{k_x}$ and $\mathcal{H}_{k_y}$:

$$\text{COCO}(\mathcal{Z}, k_x, k_y) = \frac{1}{n} \sqrt{\|\tilde{K}\tilde{L}\|_2} \in [0, \infty).$$

As a covariance, the COCO statistic is nonnegative and unbounded.

Second, the Hilbert-Schmidt Independence Criterion or HSIC (Gretton et al., 2005b) borrows the same ideas from COCO, but uses the entire spectrum of the cross-covariance operator instead of only its largest singular value, that is,

$$\text{HSIC}(\mathcal{Z}, k_x, k_y) = \frac{1}{n^2} \text{tr}(\tilde{K}\tilde{L}) \in [0, \infty).$$

HSIC relates to COCO, but was shown superior on several benchmarks (Gretton et al., 2005b).

Third, the Kernel Canonical Correlation or KCC (Bach and Jordan, 2002) is the largest kernel canonical correlation, that is,

$$\text{KCC}(\mathcal{Z}, k_x, k_y, \lambda_x, \lambda_y) = \sqrt{\|\hat{M}\|_2} \in [0, 1],$$

where $\hat{M}$ is the matrix in Equation 5.4. As an absolute correlation, the KCC statistic is bounded between zero and one. Since KCC relies on KCCA, we require the use of two regularization parameters $\lambda_x, \lambda_y > 0$ to avoid spurious perfect correlations.

When the kernels k_x, k_y are characteristic kernels (Sriperumbudur et al., 2011) the COCO, HSIC, and KCC statistics converge to zero as $n \to \infty$ if and only if the random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ are independent.

From a computational point of view, these three kernel measures of dependence rely on the computation and eigendecomposition of $n \times n$ matrices. These are operations taking $O(n^3)$ computations, a prohibitive running time for large data. In the following, we propose the Randomized Dependence Coefficient or RDC (Lopez-Paz et al., 2013a), the third example within our framework RCA, which approximates HGR in $O(n \log n)$ time, while being invariant with respect to changes in marginal distributions.

5.2.3 The randomized dependence coefficient

Estimating the dependence between two random samples $X \in \mathbb{R}^{n \times p}$ and $Y \in \mathbb{R}^{n \times q}$ with RDC involves three steps. First, RDC maps the two input

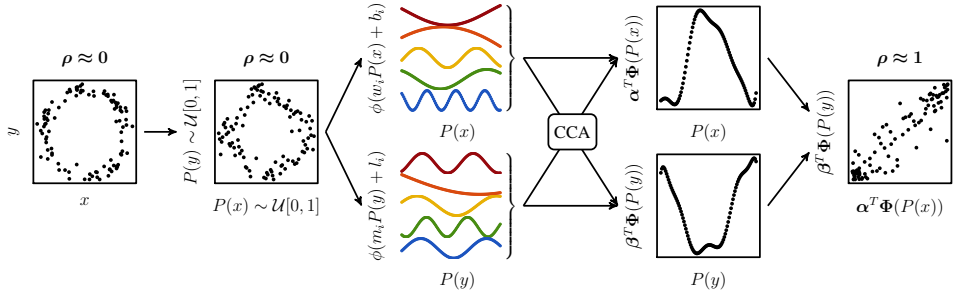


Figure 5.1: RDC computation for the sample $\{(x_i, y_i)\}_{i=1}^{100}$ drawn from a noisy circular pattern.

samples to their respective *empirical copula transformations*

$$X = (x_1, \ldots, x_n)^\top \mapsto T_{\boldsymbol{x},n}(X) := (T_{\boldsymbol{x},n}(x_1), \ldots, T_{\boldsymbol{x},n}(x_n))^\top,$$
$$Y = (y_1, \ldots, y_n)^\top \mapsto T_{\boldsymbol{y},n}(Y) := (T_{\boldsymbol{y},n}(y_1), \ldots, T_{\boldsymbol{y},n}(y_n))^\top,$$

which have uniformly distributed marginals.

Working with copulas makes RDC invariant with respect to transformations on the marginal distributions, as requested by Rényi's fifth property. Second, RDC maps the copula data to a randomized feature representation $\phi_{\boldsymbol{x},m} : \mathbb{R}^{n\times p} \mapsto \mathbb{R}^{n\times m_x}$ and $\phi_{\boldsymbol{y},m} : \mathbb{R}^{n\times q} \mapsto \mathbb{R}^{n\times m_y}$, constructed as in (5.1). For simplicity, let $m_x = m_y = m$. That is, we compute:

$$T_{\boldsymbol{x},n}(X) \mapsto \phi_{\boldsymbol{x},m}(T_{\boldsymbol{x},n}(X))^\top := (\phi_{\boldsymbol{x},m}(T_{\boldsymbol{x},n}(x_1)), \ldots, \phi_{\boldsymbol{x},m}(T_{\boldsymbol{x},n}(x_n)))^\top,$$
$$T_{\boldsymbol{y},n}(Y) \mapsto \phi_{\boldsymbol{y},m}(T_{\boldsymbol{y},n}(Y))^\top := (\phi_{\boldsymbol{y},m}(T_{\boldsymbol{y},n}(y_1)), \ldots, \phi_{\boldsymbol{y},m}(T_{\boldsymbol{y},n}(y_n)))^\top.$$

Third, RDC is the largest canonical correlation between the previous two maps

$$\mathrm{RDC}(X, Y) = \sup_{\alpha,\beta} \rho(\langle \phi_{\boldsymbol{x},m}(T_{\boldsymbol{x},n}(X)),\, \alpha\rangle, \langle \phi_{\boldsymbol{y},m}(T_{\boldsymbol{y},n}(Y)),\, \beta\rangle), \tag{5.7}$$

where $\alpha, \beta \in \mathbb{R}^{m\times 1}$. Figure 5.1 offers a sketch of this process.

In another words, RDC is the largest canonical correlation as computed by RCCA on random features of the copula transformations of two random samples.

Properties of RDC

RDC enjoys some attractive properties. First, its computational complexity is $O((p + q)n \log n + m^2 n)$, that is, log-linear with respect to the sample size. This cost is due to the estimation of two copula transformations and the largest RCCA eigenvalue. Second, RDC is easy to implement. Third, RDC compares well with the state-of-the-art. Table 5.1 summarizes, for

Dependence coefficient	**Nonlinear measure**	**Multidim. inputs**	**Marginal invariant**	**Renyi's axioms**	**Coeff.** $\in [0,1]$	**# Par.**	**Comp. Cost**
Pearson's ρ	×	×	×	×	✓	0	n
Spearman's ρ	×	×	✓	×	✓	0	$n \log n$
Kendall's τ	×	×	✓	×	✓	0	$n \log n$
CCA	×	✓	×	×	✓	0	n
KCCA	✓	✓	×	×	✓	1	n^3
ACE	✓	×	×	✓	✓	1	n
MIC	✓	×	×	×	✓	1	$n^{1.2}$
dCor	✓	✓	×	×	✓	1	n^2
HSIC	✓	✓	×	×	×	1	n^2
CHSIC	✓	✓	✓	×	×	1	n^2
RDC	✓	✓	✓	✓	✓	2	$n \log n$

Table 5.1: Comparison of measures of dependence.

a selection of well-known measures of dependence, whether they allow for general nonlinear dependence estimation, handle multidimensional random variables, are invariant with respect to changes in the one-dimensional marginal distributions of the variables under analysis, return a statistic in $[0, 1]$, satisfy Rényi's properties, and their number of parameters. As parameters, we here count the kernel function for kernel methods, the basis function and number of random features for RDC, the stopping tolerance for ACE (Breiman and Friedman, 1985) and the grid size for MIC. The table lists computational complexities with respect to sample size.

RDC can prescind from the use of copulas. In that case, RDC would no longer be scale-invariant, but would avoid potential pitfalls related to the misuse of copulas (see Remark 4.4).

Fourth, RDC is consistent with respect to KCCA. In particular, we are interested in how quickly does RDC converge to KCCA when the latter is performed on the true copula transformations of the pair of random variables under study. For that, consider the matrix

$$\tilde{Q} := \begin{pmatrix} I & \tilde{Q}_{12} \\ \tilde{Q}_{21} & I \end{pmatrix}. \tag{5.8}$$

with blocks

$$\tilde{Q}_{12} = (\tilde{K}_x + n\lambda I)^{-1} \tilde{K}_x \tilde{K}_y (\tilde{K}_y + n\lambda I)^{-1},$$
$$\tilde{Q}_{21} = (\tilde{K}_y + n\lambda I)^{-1} \tilde{K}_y \tilde{K}_x (\tilde{K}_x + n\lambda I)^{-1},$$

where $\tilde{K}_{x,i,j} = k(T_n(x_i), T_n(x_j))$ are true kernel evaluations on the empirical copula of the data. The matrix $\hat{Q}$ has the same structure, but operates on the empirical copula of the data and random features. The matrix Q has the same structure, but operates on the true copula of the data and the true kernel. The following theorem provides an specific rate on the convergence of RDC to the largest copula kernel canonical correlation.

Theorem 5.3 (Convergence of RDC)**.** *Consider the definitions from the previous paragraph, and the assumptions from Definition 5.1. Then,*

$$\begin{aligned}\mathbb{E}\left[\|\hat{\boldsymbol{Q}} - Q\|\right] &\leq \left\{\frac{6}{n}\left(\frac{1}{\lambda^2} + \frac{1}{\lambda}\right)\right\} \\ &\times \left(\sqrt{\frac{3B\|K\|\log n}{m}} + \frac{2B\log n}{m} + 2L_k\sqrt{nd}\left(\sqrt{\pi} + \sqrt{\log 2d}\right)\right) \\ &\leq 1,\end{aligned}$$

where $\|K\| = \max(\|K_x\|, \|K_y\|)$*, and* $m = \min(m_x, m_y)$*.*

Proof. See Secton 5.5.3. □

As it happened with KCCA, regularization is necessary in RDC to avoid spurious perfect correlations. However, (5.7) lacks regularization. This is because, as we will see in our numerical simulations, using a small number of random features (smaller than the number of samples) provides an implicit regularization that suffices for good empirical performance.

5.2.4 Conditional RDC

In some situations, including the causal inference problems studied in the second part of this thesis, we will study the statistical dependence of two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ when conditioned to the effects of a third random variable $\boldsymbol{z}$. Mathematically, $\boldsymbol{x}$ and $\boldsymbol{y}$ are conditionally independent given $\boldsymbol{z}$ if the equality

$$p(x, y \,|\, z) = p(x \,|\, z)p(y \,|\, z)$$

holds for all $x \in \mathcal{X}$, $y \in \mathcal{Y}$, and $z \in \mathcal{Z}$. Measuring conditional dependence using RDC relies on partial CCA (Rao, 1969), a variant of CCA designed to measure the correlation between two multidimensional random samples X and Y after eliminating the effects of a third sample Z. Partial canonical correlations are the solutions of the following generalized eigenvalue problem:

$$\begin{aligned}C_{xy|z}C_{yy|z}^{-1}C_{yx|z}f &= \rho^2 C_{xx|z}f \\ C_{yx|z}C_{xx|z}^{-1}C_{xy|z}g &= \rho^2 C_{yy|z}g,\end{aligned}$$

where $C_{ij|z} = C_{iz}C_{zz}^{-1}C_{zj}$, for $i, j \in \{x, y\}$. In this case, computing the conditional RDC is as follows. First, we map the three random samples $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ to a randomized nonlinear representation of their copula transformations. Second, we compute the conditional RDC as the largest partial canonical correlation between these three random feature maps.

Remark 5.8 *(A general recipe for measures of conditional dependence)***.** There is a common recipe to measure conditional dependence using unconditional measures of dependence an nonlinear regression methods:

1. Estimate the regression residuals $\boldsymbol{r}_x = \boldsymbol{x} - \mathbb{E}[\boldsymbol{x} \mid z]$.
2. Estimate the regression residuals $\boldsymbol{r}_y = \boldsymbol{y} - \mathbb{E}[\boldsymbol{y} \mid z]$.
3. Estimate the dependence between $\boldsymbol{r}_x$ and $\boldsymbol{r}_y$.

$\lozenge$

Hypothesis testing with RDC

Consider the hypothesis "the two sets of nonlinear projections are mutually uncorrelated". Under normality assumptions and large sample sizes, Bartlett's approximation (Mardia et al., 1979) approximates the null-distribution of RCCA as

$$\left(\frac{2k+3}{2} - n\right) \log \prod_{i=1}^{k} (1 - \rho_i^2) \sim \chi^2_{k^2},$$

which can be easily adapted for approximate RDC hypothesis testing.

Alternatively, we could use bootstrapping to obtain nonparametric estimates of the null-distribution of RDC. Figure 5.2 shows the null-distribution of RDC for unidimensional random samples and different sample sizes n, as estimated from $100,000$ pairs of independent random samples. The Beta distribution (dashed lines in the figure) is a good approximation to the empirical null-distribution of RDC (solid lines). The parameters of the Beta distribution vary smoothly as the sample size n increases. For scalar random variables, the marginal distributions of the random samples under measurement do not have any effect on the null-distribution, thanks to the scale invariance provided by the empirical copula transformation. Therefore, tables for the null-distribution of RDC can be efficiently pre-computed for one-dimensional random variables.

5.2.5 Model selection and hypothesis testing

All the measures of dependence presented in this section have tunable parameters: their regularizers, kernel functions, parameters of these kernel functions, and so forth. This is not a novel nuisance for us, as tunable parameters populated our discussions in previous chapters about data representation and density estimation. All these parameters were tuned by monitoring the objective function of the problem at hand in some *held out* validation data.

To some extent, the measures of dependence from this section follow the same techniques for model selection (Sugiyama et al., 2012). After all, these algorithms aim at extracting the largest amount of patterns from data, as long as those patterns are not hallucinated from noise. Therefore, cross-validation is of use to avoid overfitting. If possible, such cross-validation

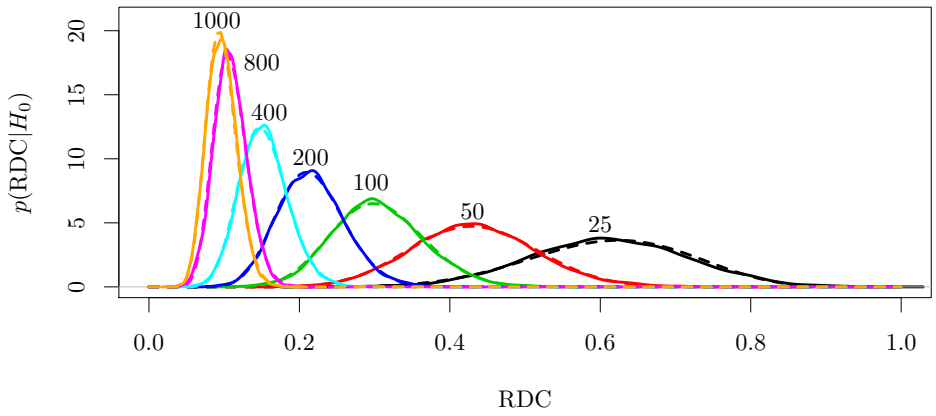


Figure 5.2: Empirical null-distribution of RDC for unidimensional random variables (solid lines), and corresponding Beta-approximations (dashed lines). Sample sizes on the top of each associated curve.

should aim at directly maximizing the power[1] of the dependence statistic (Gretton et al., 2012b).

Model selection is more subtle when performed for hypothesis testing. In dependence testing, our null hypothesis H_0 means "the random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ are independent". Therefore, a type-I error (false positive) is to conclude that a pair of independent random variables is dependent, and a type-II error (false negative) is to conclude that a pair of dependent random variables is independent. Regarding parameters, simultaneously avoiding type-I and type-II errors are two conflicting interests: parameters providing flexible measures of dependence (for instance, RDC with large number of random features) will tend to make type-I errors (overfit, low bias, high variance), and rigid measures of dependence will tend to make type-II errors (underfit, high bias, low variance). Nevertheless, in some applications, one of the two errors is more severe. For instance, it is worse to tell a patient suffering from cancer that he is healthy, rather than diagnosing a healthy patient with cancer. Mathematically, given a measure of dependence with parameters θ, model selection could maximize the objective

$$\arg\min_{\theta} \underbrace{(1-\lambda)\text{dep}(\mathcal{Z},\theta)}_{\text{avoids type-II error}} + \underbrace{\lambda\text{dep}(\mathcal{Z}_\pi,\theta)}_{\text{avoids type-I error}} ,$$

where $\mathcal{Z}_\pi$ is a copy of $\mathcal{Z}$ where the samples of the second random variable have been randomly permuted, and $\lambda \in [0,1]$ is a parameter that balances the importance between type-I and type-II errors, and depends on the problem at hand.

[1] The power of a dependence test is the probability that the test rejects the independence hypothesis when analyzing dependent random variables.

5.3 Two-sample tests

The problem of two-sample testing addresses the following question:

Given two samples $\{x_i\}_{i=1}^n \sim P^n$ *and* $\{y_i\}_{i=1}^n \sim Q^n$*, is* $P = Q$*?*

One popular nonparametric two-sample test is the *Maximum Mean Discrepancy* or MMD (Gretton et al., 2012a). Given a kernel function k, the empirical MMD statistic is

$$\mathrm{MMD}^2(\mathcal{Z}, k) = \frac{1}{n_x^2} \sum_{i,j=1}^{n_x^2} k(x_i, x_j) - \frac{2}{n_x n_y} \sum_{i,j=1}^{n_x, n_y} k(x_i, y_j) + \frac{1}{n_y^2} \sum_{i,j=1}^{n_y} k(y_i, y_j), \tag{5.9}$$

When k is a characteristic kernel and $n \to \infty$, the MMD statistic is zero if and only if $P = Q$. For simplicity, let $n_x = n_y = n$; then, computing the MMD statistic takes $O(n^2)$ operations. By making use of the random features introduced in Section 3.2.2, we can define an approximate, randomized version of MMD

$$\begin{aligned} \mathrm{RMMD}^2(\mathcal{Z}, k) &= \left\| \frac{1}{n_x} \sum_{i=1}^{n_x} \hat{\phi}_k(x_i) - \frac{1}{n_y} \sum_{i=1}^{n_y} \hat{\phi}_k(y_i) \right\|_{\mathbb{R}^m}^2 \\ &= \frac{1}{n_x^2} \sum_{i,j=1}^{n_x^2} \hat{k}(x_i, x_j) - \frac{2}{n_x n_y} \sum_{i,j=1}^{n_x, n_y} \hat{k}(x_i, y_j) + \frac{1}{n_y^2} \sum_{i,j=1}^{n_y} \hat{k}(y_i, y_j), \end{aligned} \tag{5.10}$$

where $\hat{\phi}$ is a random feature map and $\hat{k}$ is the induced approximate kernel. RMMD can be computed in $O(nm)$ operations, and is the fourth example of our framework RCA.

Theorem 5.4 (Convergence of RMMD)**.** *Let the* MMD *and* RMMD *be as in* (5.9) *and* (5.10)*, respectively. Let the data be* d*-dimensional and live in a compact set* $\mathcal{S}$ *of diameter* $|\mathcal{S}|$*, let* RMMD *use* m *random features corresponding to a shift invariant kernel, and let* $n = n_x = n_y$*. Then,*

$$\mathbb{P}\left(|\mathrm{MMD}(\mathcal{Z}, k) - \mathrm{RMMD}(\mathcal{Z}, k)| \geq \frac{4(h(d, |\mathcal{S}|, c_k) + \sqrt{2t})}{\sqrt{m}} \right) \leq \exp(-t),$$

where the function h *is defined as in* (3.13)*.*

Proof. See Section 5.5.4. □

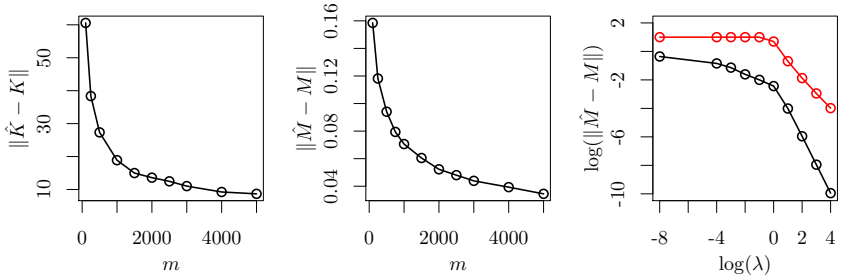


Figure 5.3: Matrix Bernstein inequality error norms.

5.4 Numerical simulations

We evaluate the performance of a selection of the RCA methods introduced in this chapter throughout a variety of experiments, on both synthetic and real-world data. In particular, we organize our numerical simulations as follows. Section 5.4.1 validates the Bernstein bounds from Theorem 5.1 and Corollary 5.1. Section 5.4.3 evaluates the performance of RCCA on the task of learning shared representations between related datasets. Section 5.4.4 explores the use of RCCA in Vapnik's *learning using privileged information* setup. Section 5.4.2 exemplifies the use of RPCA as an scalable, randomized strategy to train autoencoder neural networks. Finally, Section 5.4.5 offers a variety of experiments to study the capabilities of RDC to measure statistical dependence between multivariate random variables. The Gaussian random features used throughout these experiments are like the ones from Equation 3.11. The bandwidth parameter γ is adjusted using the median heuristic, unless stated otherwise.

5.4.1 Validation of Bernstein bounds

We now validate empirically the Bernstein bounds obtained in Theorem 5.1 and Corollary 5.1. To do so, we perform simulations in which we separately vary the values of the two tunable parameters in RPCA and RCCA: the number of random projections m, and the regularization parameter λ. We use synthetic data matrices $X \in \mathbb{R}^{1000\times 10}$ and $Y \in \mathbb{R}^{1000\times 10}$, formed by iid normal entries. When not varying, the parameters are fixed to $m = 1000$ and $\lambda = 10^{-3}$.

Figure 5.3 depicts the value of the norms from equations (5.2, 5.5), as the parameters $\{m, \lambda\}$ vary, when averaged over a total of 100 random data matrices X and Y. The simulations agree with the presented theoretical analysis: the number of random features m has an inverse square root effect in both RPCA and RCCA, and the effect of the regularization parameter is upper bounded by the theoretical bound $\min(1, \lambda^{-1} + \lambda^{-2})$ (depicted in red) in RCCA.

Figure 5.4: Autoencoder reconstructions of unseen test images for the MNIST (top) and CIFAR-10 (bottom) datasets.

5.4.2 Principal component analysis

One use for RPCA is the scalable training of nonlinear autoencoders (for a review on autoencoders, see Section 5.6). The process involves i) mapping the observed data $X \in \mathbb{R}^{n \times D}$ into the latent factors $Z \in \mathbb{R}^{n \times d}$ using the top d nonlinear principal components from RPCA, and ii) reconstructing X from Z using D nonlinear regressors. Figure 5.4 shows the reconstruction of *unseen* MNIST and CIFAR-10 images from the RPCA compressions. Here, the number random projections is $m = 2000$, the number of latent dimensions is $d = 20$ for MNIST, and $d = 40$ (first row) or $d = 100$ (second row) for CIFAR-10. Training took under 200 seconds on a 1.8GhZ processor for each dataset of 50000 samples.

5.4.3 Canonical correlation analysis

We compare three variants of CCA on the task of learning correlated features from two modalities of the same data: linear CCA, Deep CCA (Andrew et al., 2013) and the proposed RCCA. Deep CCA (DCCA), the current state-of-the-art, feeds the pair of input samples through a deep neural network, and learns its weights by solving a nonconvex optimization problem with gradient descent. We were unable to run exact KCCA on the proposed datasets due to its cubic complexity. Instead, we offer a comparison to a low-rank approximation based on the Nyström method (see Section 3.2.1). We replicate the two experiments from Andrew et al. (2013). The task is to measure the test correlation between canonical variables computed on some training data. The participating datasets are MNIST and XRMB (Andrew et al., 2013).

For the MNIST dataset, we learn correlated representations between the left and right halves of the MNIST images (LeCun et al., 1998b). Each image

Table 5.2: Sum of largest test canonical correlations and running times by all CCA variants in the MNIST and XRMB datasets.

RCCA on **MNIST** (50 largest canonical correlations)

m_x, m_y	Fourier		Nyström	
	corr.	minutes	corr.	minutes
1000	36.31	5.55	41.68	5.29
2000	39.56	19.45	43.15	18.57
3000	40.95	41.98	43.76	41.25
4000	41.65	73.80	44.12	75.00
5000	41.89	112.80	44.36	115.20
6000	42.06	153.48	**44.49**	156.07

RCCA on **XRMB** (112 largest canonical correlations)

m_x, m_y	Fourier		Nyström	
	corr.	minutes	corr.	minutes
1000	68.79	2.95	81.82	3.07
2000	82.62	11.45	93.21	12.05
3000	89.35	26.31	98.04	26.07
4000	93.69	48.89	100.97	50.07
5000	96.49	79.20	103.03	81.6
6000	98.61	120.00	**104.47**	119.4

	linear CCA		DCCA	
	corr.	minutes	corr.	minutes
MNIST	28.0	0.57	39.7	787.38
XRMB	16.9	0.11	92.9	4338.32

has a width and height of 28 pixels; therefore, each of the two views of CCA consists on 392 features. We use 54000 random samples for training, 10000 for testing and 6000 to cross-validate the parameters of CCA and DCCA. For the X-Ray Microbeam Speech (XRMB) dataset, we learn correlated representations of simultaneous acoustic and articulatory speech measurements (Andrew et al., 2013). The articulatory measurements describe the position of the speaker's lips, tongue and jaws for seven consecutive frames, yielding a 112-dimensional vector at each point in time; the acoustic measurements are the MFCCs for the same frames, producing a 273-dimensional vector for each point in time. We use 30000 random samples for training, 10000 for testing and 10000 to cross-validate the parameters of CCA and DCCA.

Table 5.2 shows the sum of the largest canonical correlations (corr.) in the test sets of both MNIST and XRMB, obtained by each CCA variant, as well as their running times (minutes, single 1.8GHz core). For RCCA, we use representations based on Nyström and Mercer random features (see Section 3.2).[2] Given enough random projections ($m = m_x = m_y$), RCCA is able to extract the most test correlation while running drastically faster than DCCA. Moreover, when using random Mercer features, the number of parameters of the RCCA model is up to two orders of magnitude lower than for DCCA.

We tune no parameters for RCCA: the kernel widths were set using the median heuristic (see Section 3.1.2), and CCA regularization is implicitly provided by the use of random features (and thus set to 10^{-8}). On the contrary, DCCA has ten parameters (two autoencoder parameters for pre training, number of hidden layers, number of hidden units and CCA regularizers for each view), which were cross-validated using the grids described in Andrew et al. (2013). Cross-validating RCCA parameters did not improve

[2]The theorems presented in this chapter only apply to Mercer random features.

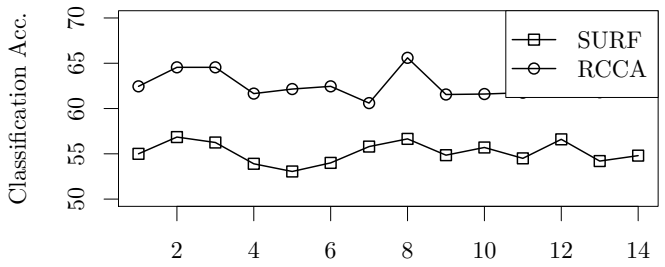


Figure 5.5: Results for the LUPI experiments.

our results.

5.4.4 Learning using privileged information

In Vapnik's *Learning Using Privileged Information* (LUPI) paradigm (Vapnik and Vashist, 2009) the learner has access to a set of *privileged* features or information $X_\star$, exclusive of training time, that he would like to exploit to obtain a better classifier for test time. Although we will discuss the problem of learning using privileged information in Section 8.1, we now test the capabilities of RCCA to address this problem. To this end, we propose the use RCCA to construct a highly correlated subspace between the regular features X and the privileged features $X_\star$, accessible at test time through a nonlinear transformation of X.

We experiment with the *Animals-with-Attributes* dataset[3]. In this dataset, the regular features X are the SURF descriptors of 30000 pictures of 35 different animals; the privileged features $X_\star$ are 85 high-level binary attributes associated with each picture (such as *eats-fish* or *can-fly*). To extract information from $X_\star$ at training time, we build a feature space formed by the concatenation of the 85, five-dimensional top canonical variables associated with $\text{RCCA}(X, [X_\star^{(i)}, y])$, $i \in \{1, \ldots, 85\}$. The vector y denotes the training labels.

We perform 14 random training/test partitions of 1000 samples each. Each partition groups a random subset of 10 animals as class "0" and a second random subset of 10 animals as class "1". Hence, each experiment is a different, challenging binary classification problem. Figure 5.5 shows the test classification accuracy of a linear SVM when using as features the images' SURF descriptors or the RCCA "semiprivileged" features. As a side note, directly using the high-level attributes yields 100% accuracy. The cost parameter of the linear SVM is cross-validated on the grid $[10^{-4}, \ldots, 10^4]$. We observe an average improvement of 14% in classification when using the RCCA basis instead of the image features alone. Results are statistically significant respect to a paired Wilcoxon test on a 95% confidence interval.

[3] `http://attributes.kyb.tuebingen.mpg.de/`

The SVM+ algorithm (Vapnik and Vashist, 2009) did not improve our results when compared to regular SVM using SURF descriptors.

5.4.5 The randomized dependence coefficient

We perform experiments on both synthetic and real-world data to validate the empirical performance of RDC as a measure of statistical dependence.

Concerning parameter selection, for RDC we set the number of random features to $k = 20$ for both random samples, and observed no significant improvements for larger values. The random feature bandwidth γ is set to a linear scaling of the input variable dimensionality d. Note that the stability of RDC can be improved by allowing a larger amount of random features, and regularizing the RCCA step using cross-validation (Section 5.2.5). In all our experiments $\gamma = \frac{1}{6d}$ worked well. On the other hand, HSIC and CHSIC (HSIC on copula) use Gaussian kernels $k(z, z') = exp(-\gamma\|z - z'\|_2^2)$ with γ set using the median heuristic. For MIC, the search-grid size is $B(n) = n^{0.6}$, as recommended in (Reshef et al., 2011). The tolerance of ACE is $\epsilon = 0.01$, the default value in the R package `acepack`.

Resistance to additive noise

We define the *power* of a measure of dependence as its ability to discern between dependent and independent samples that share equal marginal distributions. We follow the experiments of Simon and Tibshirani[4], and choose 8 bivariate association patterns, depicted inside boxes in Figure 5.6. For each of the 8 association patterns, we generate 500 repetitions of 500 samples, in which the input sample is uniformly distributed on the unit interval. Next, we regenerated the input sample randomly, to generate independent versions of each sample with equal marginals. Figure 5.6 shows the power for the discussed nonlinear measures of dependence as the variance of some zero-mean Gaussian additive noise increases from 1/30 to 3. RDC shows worse performance in the linear association pattern due to overfitting, and in the step-function due to the smoothness prior induced by the Gaussian random features. On the other hand, RDC shows good performance in nonfunctional patterns. As a future research direction, it would be interesting to analyze the separate impact of copulas and CCA in the performance of RDC (see a similar discussion by Gretton et al. (2005b)), and to cross-validate the parameters of all the competing measures of dependence.

Statistic semantics

Figure 5.7 shows RDC, ACE, dCor, MIC, Pearson's ρ, Spearman's rank and Kendall's τ dependence estimates for 14 different associations of two

[4] `http://www-stat.stanford.edu/~tibs/reshef/comment.pdf`

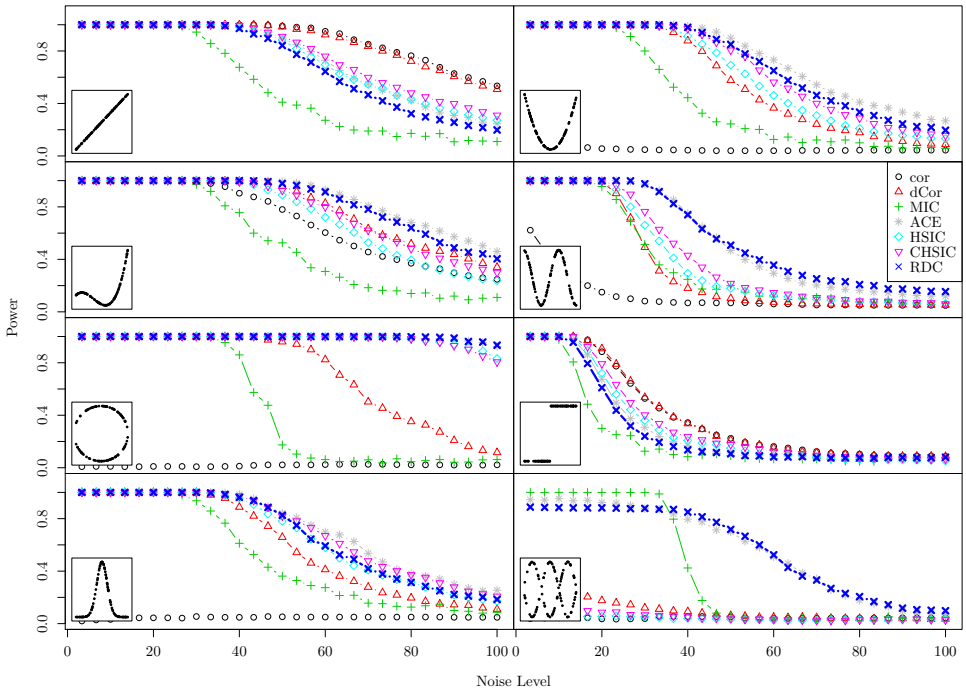


Figure 5.6: Power of discussed measures on example bivariate association patterns as noise increases. Insets show the noise-free form of each association pattern.

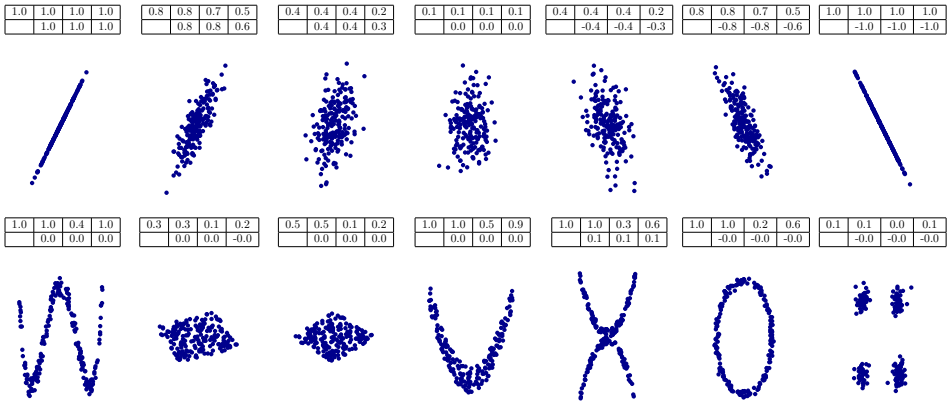


Figure 5.7: RDC, ACE, dCor, MIC, Pearson's ρ, Spearman's rank and Kendall's τ estimates (numbers in tables above plots, in that order).

Table 5.3: Average running times (in seconds) for measures of dependence on vs sample sizes.

sample size	**Pearson's ρ**	**RDC**	**ACE**	**KCCA**	**dCor**	**HSIC**	**CHSIC**	**MIC**
1,000	0.0001	0.0047	0.0080	0.402	0.3417	0.3103	0.3501	1.0983
10,000	0.0002	0.0557	0.0782	3.247	59.587	27.630	29.522	—
100,000	0.0071	0.3991	0.5101	43.801	—	—	—	—
1,000,000	0.0914	4.6253	5.3830	—	—	—	—	—

scalar random samples. RDC is close to one on all the proposed dependent associations, and is close to zero for the independent association, depicted last. When the associations are Gaussian (first row), RDC is close to the absolute value Pearson's correlation coefficient, as requested by the seventh property of Rényi.

Computational complexity

Table 5.3 shows running times for the considered nonlinear measures of dependence on scalar, uniformly distributed, independent samples of sizes $\{10^3, \ldots, 10^6\}$, when averaged over 100 runs. We cancelled all simulations running over ten minutes. The implementation of Pearson's ρ, ACE, dCor (Székely et al., 2007), KCCA (Bach and Jordan, 2002) and MIC is in C, and the one of RDC, HSIC and CHSIC is in R.

Feature selection in real-world data.

We performed greedy feature selection via dependence maximization (Song et al., 2012) on real-world datasets. More specifically, we aim at constructing the subset of features $\mathcal{G} \subset \mathcal{X}$ that minimizes the Normalized Mean Squared

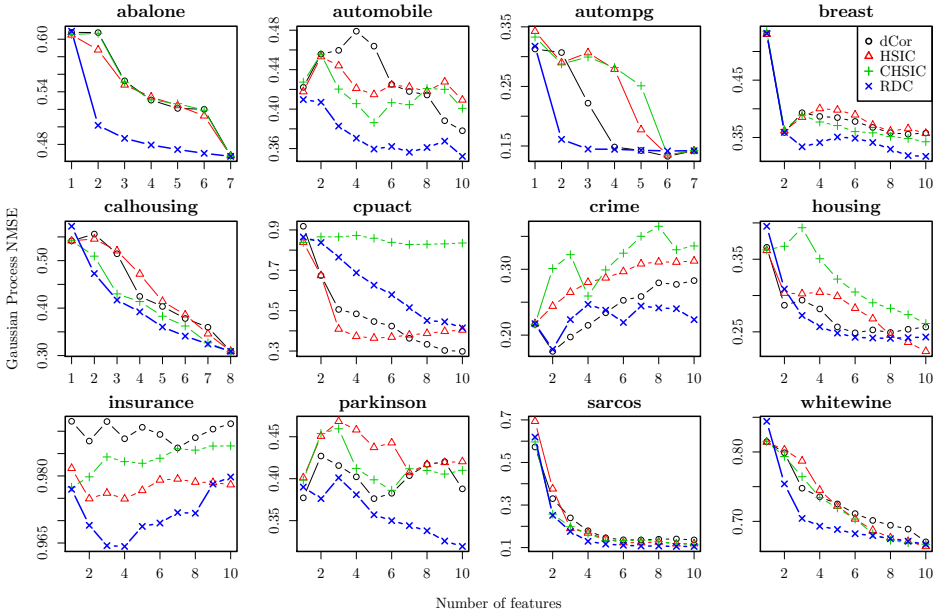


Figure 5.8: Feature selection experiments on real-world datasets.

Error (NMSE) of a Gaussian process. We do so by selecting the feature $x_{:,i}$ maximizing dependence between the feature set $\mathcal{G}_i = \{\mathcal{G}_{i-1}, x_{:,i}\}$ and the target variable y at each iteration $i \in \{1, \dots 10\}$, such that $\mathcal{G}_0 = \{\emptyset\}$ and $x_{:,i} \notin \mathcal{G}_{i-1}$.

We considered 12 heterogeneous datasets, obtained from the UCI dataset repository[5], the Gaussian process web site Data[6] and the Machine Learning data set repository[7]. All random training and test partitions are disjoint and of equal size.

Since $\mathcal{G}$ can be multi-dimensional, we compare RDC to the multivariate methods dCor, HSIC and CHSIC. Given their quadratic computational demands, dCor, HSIC and CHSIC use up to 1000 points when measuring dependence. This constraint only applied on the `sarcos` and `abalone` datasets. Results are averages over 20 random training/test partitions.

Figure 5.8 summarizes the results for all datasets and algorithms as the number of selected features increases. RDC performs best in most datasets, using a much lower running time than its contenders. In some cases, adding more features damages test accuracy. This is because the added features may be irrelevant, and the corresponding increase in dimensionality turns the learning problem harder.

[5] `http://www.ics.uci.edu/~mlearn`
[6] `http://www.gaussianprocess.org/gpml/data/`
[7] `http://www.mldata.org`

5.5 Proofs

5.5.1 Theorem 5.1

Proof. Observe that $\mathbb{E}\left[\hat{\boldsymbol{K}}\right] = K$, and that $\hat{K}$ is the sum of the m independent matrices $\hat{K}_i$, where the randomness is over random feature sampling. This is because the random features are independently and identically distributed, and the data matrix X is constant. Consider the error matrix

$$E = \hat{K} - K = \sum_{i=1}^{m} E_i, \text{ where } E_i = \frac{1}{m}(\hat{K}^{(i)} - K),$$

and $\mathbb{E}[\boldsymbol{E}_i] = 0$ for all $1 \leq i \leq m$. Since we are using bounded kernels and features (Definition 5.1), it follows that there exists a constant B such that $\|z\|^2 \leq B$. Thus,

$$\begin{aligned}\|E_i\| &= \frac{1}{m}\|z_i z_i^\top - \mathbb{E}\left[zz^\top\right]\| \\ &\leq \frac{1}{m}(\|z_i\|^2 + \mathbb{E}\left[\|\boldsymbol{z}\|^2\right]) \leq \frac{2B}{m},\end{aligned}$$

because of the triangle inequality on the norm and Jensen's inequality on the expected value. To bound the variance of E, bound first the variance of each of its summands E_i and observe that $\mathbb{E}\left[\boldsymbol{z}_i \boldsymbol{z}_i^\top\right] = K$:

$$\begin{aligned}\mathbb{E}\left[\boldsymbol{E}_i^2\right] &= \frac{1}{m^2}\mathbb{E}\left[(\boldsymbol{z}_i \boldsymbol{z}_i^\top - K)^2\right] \\ &= \frac{1}{m^2}\mathbb{E}\left[\|\boldsymbol{z}_i\|^2 \boldsymbol{z}_i \boldsymbol{z}_i^\top - \boldsymbol{z}_i \boldsymbol{z}_i^\top K - K \boldsymbol{z}_i \boldsymbol{z}_i^\top + K^2\right] \\ &\preceq \frac{1}{m^2}\left[BK - 2K^2 + K^2\right] \preceq \frac{BK}{m^2}.\end{aligned}$$

Next, taking all summands E_i together we obtain

$$\|\mathbb{E}\left[\boldsymbol{E}^2\right]\| \leq \left\|\sum_{i=1}^{m} \mathbb{E}\left[\boldsymbol{E}_i^2\right]\right\| \leq \frac{1}{m} B\|K\|,$$

where the first inequality follows by Jensen. We can now invoke the matrix Bernstein inequality (Theorem 2.9) on $E - \mathbb{E}[\boldsymbol{E}]$ and obtain the bound:

$$\mathbb{E}\left[\|\hat{\boldsymbol{K}} - K\|\right] \leq \sqrt{\frac{3B\|K\| \log n}{m}} + \frac{2B \log n}{m}.$$

□

5.5.2 Theorem 5.2

Proof. We are looking after an upper bound on the norm of the matrix

$$\begin{pmatrix} 0 & (\hat{K}_x + n\lambda I)^{-1}\hat{K}_x\hat{K}_y(\hat{K}_y + n\lambda I)^{-1} \\ (\hat{K}_y + n\lambda I)^{-1}\hat{K}_y\hat{K}_x(\hat{K}_x + n\lambda I)^{-1} & 0 \end{pmatrix} - \begin{pmatrix} 0 & (K_x + n\lambda I)^{-1}K_xK_y(K_y + n\lambda I)^{-1} \\ (K_y + n\lambda I)^{-1}K_yK_x(K_x + n\lambda I)^{-1} & 0 \end{pmatrix}, \tag{5.11}$$

where the identity matrices have canceled out. The norm of this matrix is upper bounded by the sum of the norms of each block, due to the triangle inequality. Therefore, we first bound the norm of

$$(\hat{K}_y+n\lambda I)^{-1}\hat{K}_y\hat{K}_x(\hat{K}_x+n\lambda I)^{-1}-(K_y+n\lambda I)^{-1}K_yK_x(K_x+n\lambda I)^{-1}. \tag{5.12}$$

The other block is bounded analogously. We follow a similar argument to Fukumizu et al. (2007). Start by observing that (5.12) equals

$$\left[(\hat{K}_y + n\lambda I)^{-1} - (K_y + n\lambda I)^{-1}\right]\hat{K}_y\hat{K}_x(\hat{K}_x + n\lambda I)^{-1} \tag{5.13}$$

$$+ (K_y + n\lambda I)^{-1}(\hat{K}_y\hat{K}_x - K_yK_x)(\hat{K}_x + n\lambda I)^{-1} \tag{5.14}$$

$$+ (K_y + n\lambda I)^{-1}K_yK_x\left[(\hat{K}_x + n\lambda I)^{-1} - (K_x + n\lambda I)^{-1}\right]. \tag{5.15}$$

Next, use the identity

$$A^{-1} - B^{-1} = \left[B^{-1}(B^2 - A^2) + (A - B)\right]A^{-2}$$

to develop (5.13) as

$$\left\{(K_y + n\lambda I)^{-1}\left[(K_y + n\lambda I)^2 - (\hat{K}_y + n\lambda I)^2\right] + (\hat{K}_y - K_y)\right\}(\hat{K}_y + n\lambda I)^{-1}\times \tag{5.16}$$

$$(\hat{K}_y + n\lambda I)^{-1}\hat{K}_y\hat{K}_x(\hat{K}_x + n\lambda I)^{-1}. \tag{5.17}$$

The norm of (5.16) can be upper-bounded using the fact that

$$\begin{aligned} \|A^2 - B^2\| &= \frac{\|(A + B)(A - B) + (A - B)(A + B)\|}{2} \\ &\leq \|(A + B)(A - B)\| \\ &\leq \|A + B\|\|A - B\| \\ &\leq (\|A\| + \|B\|)\,\|A - B\|, \end{aligned}$$

to obtain

$$\begin{aligned} &\frac{1}{n^2\lambda^2}\left(\left\|\hat{K}_y + n\lambda I\right\| + \|K_y + n\lambda I\|\right)\left\|\hat{K}_y - K_y\right\| + \frac{1}{n\lambda}\|\hat{K}_y - K_y\| \\ &\leq \left(\frac{2}{n\lambda^2} + \frac{3}{n\lambda}\right)\left\|\hat{K}_y - K_y\right\| \\ &\leq \frac{3}{n}\left(\frac{1}{\lambda} + \frac{1}{\lambda^2}\right)\left\|\hat{K}_y - K_y\right\|, \end{aligned} \tag{5.18}$$

In the previous, the second line uses the triangle inequalities $\|K_y + n\lambda I\| \leq \|K_y\| + \|n\lambda I\|$ and $\|\hat{K}_y + n\lambda I\| \leq \|\hat{K}_y\| + \|n\lambda I\|$, the boundedness of our kernel function and random features (Definition 5.1) to obtain $\|K_y\| \leq n$ and $\|\hat{K}_y\| \leq n$, and the fact that $\|n\lambda I\| \leq n\lambda$.

Since the norm of (5.17) is upper-bounded by 1, Equation 5.18 is also an upper-bound for (5.13). Similarly, upper-bound the norm of (5.15) by

$$\frac{3}{n}\left(\frac{1}{\lambda} + \frac{1}{\lambda^2}\right)\left\|\hat{K}_x - K_x\right\|. \tag{5.19}$$

Finally, an upper-bound for (5.14) is

$$\begin{aligned}
&\left\|(K_y + n\lambda I)^{-1}(\hat{K}_y\hat{K}_x - K_yK_x)(\hat{K}_x + n\lambda I)^{-1}\right\| \\
&\quad\leq \frac{1}{n^2\lambda^2}\left\|\hat{K}_y\hat{K}_x - K_yK_x\right\| \\
&\quad= \frac{1}{n^2\lambda^2}\left\|\hat{K}_y\hat{K}_x - K_yK_x + \hat{K}_yK_x - \hat{K}_yK_x\right\| \\
&\quad= \frac{1}{n^2\lambda^2}\left\|\hat{K}_y(\hat{K}_x - K_x) - (K_y - \hat{K}_y)K_x\right\| \\
&\quad\leq \frac{1}{n^2\lambda^2}\left(\left\|\hat{K}_y(\hat{K}_x - K_x)\right\| + \left\|(K_y - \hat{K}_y)K_x\right\|\right) \\
&\quad\leq \frac{1}{n\lambda^2}\left(\left\|\hat{K}_x - K_x\right\| + \left\|K_y - \hat{K}_y\right\|\right).
\end{aligned} \tag{5.20}$$

Equations (5.18), (5.19), and (5.20) upper-bound the norm of (5.12) as

$$\left\{\frac{3}{n}\left(\frac{1}{\lambda^2} + \frac{1}{\lambda}\right)\right\}\left(\left\|\hat{K}_x - K_x\right\| + \left\|K_y - \hat{K}_y\right\|\right).$$

Observing that this same quantity upper-bounds the norm of the upper-right block of (5.11) produces the claimed result. □

5.5.3 Theorem 5.3

Proof. We bound the two approximations (kernel and copula) separately, using the triangle inequality:

$$\|\hat{Q} - Q\| \leq \|\hat{Q} - \tilde{Q}\| + \|\tilde{Q} - Q\|,$$

where

- Q operates on the true kernel and the true copula,
- $\tilde{Q}$ operates on the true kernel and the empirical copula,
- $\hat{Q}$ operates on random features and the empirical copula.

Therefore, the overall bound will be

$$\mathbb{E}\Big[\|\hat{\boldsymbol{Q}} - Q\|\Big] \leq \left\{\frac{3}{n}\left(\frac{1}{\lambda^2} + \frac{1}{\lambda}\right)\right\}\left(\mathbb{E}\Big[\|\hat{\boldsymbol{K}}_a - \tilde{K}_a\|\Big] + \mathbb{E}\Big[\|\tilde{\boldsymbol{K}}_b - K_b\|\Big]\right),$$

where $a, b \in \{x, y\}$ are chosen to produce the worst-case upper-bound. The term $\mathbb{E}\Big[\|\hat{\boldsymbol{K}}_a - \tilde{K}_a\|\Big]$ is bounded as in Theorem 5.2. To bound $\mathbb{E}\Big[\|\tilde{\boldsymbol{K}}_b - K_b\|\Big]$, follow

$$\begin{aligned}
\mathbb{E}\Big[\|\tilde{\boldsymbol{K}}_b - K_b\|\Big] &\leq \mathbb{E}\left[\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\left(k(\tilde{\boldsymbol{u}}_i, \tilde{\boldsymbol{u}}_j) - k(u_i, u_j)\right)^2}\right] \\
&\leq \mathbb{E}\left[\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\left(L_k\|\tilde{\boldsymbol{u}}_i - u_i\| + L_k\|\tilde{\boldsymbol{u}}_j - u_j\|\right)^2}\right] \\
&\leq 2L_k n\,\mathbb{E}\left[\sup_{1\leq i\leq n}\|\tilde{\boldsymbol{u}}_i - u_i\|\right] \\
&\leq 2L_k n\,\mathbb{E}\left[\sup_{x\in\mathcal{X}}\|T_n(x) - T(x)\|\right] \\
&\leq 2L_k\sqrt{nd}\left(\sqrt{\pi} + \sqrt{\log 2d}\right).
\end{aligned}$$

where the inequalities follow from the Frobenius norm dominating the operator norm, the L_k-Lipschitzness of the kernel function, the analysis of the worst difference, the generalization of the worst difference to the whole input domain, and applying the expectation of Bernstein's inequality (Theorem 2.7) to Corollary 4.1. □

5.5.4 Theorem 5.4

Proof. Unfold the definitions of MMD and RMMD as

$$\begin{aligned}
|\mathrm{MMD}(\mathcal{Z}, k) - \mathrm{RMMD}(\mathcal{Z}, k)| \leq \frac{1}{n^2}\sum_{i,j=1}^{n^2}|k(x_i, x_j) - \hat{k}(x_i, x_j)| \\
+ \frac{2}{n^2}\sum_{i,j=1}^{n^2}|k(x_i, y_j) - \hat{k}(x_i, y_j)| \\
+ \frac{1}{n^2}\sum_{i,j=1}^{n^2}|k(y_i, y_j) - \hat{k}(y_i, y_j)|
\end{aligned}$$

where the upper bound follows by applying the triangle inequality. The claim follows by noticing that our data lives in a compact set $\mathcal{S}$ of diameter $|\mathcal{S}|$, and by applying Equations 3.12 and 3.13. □

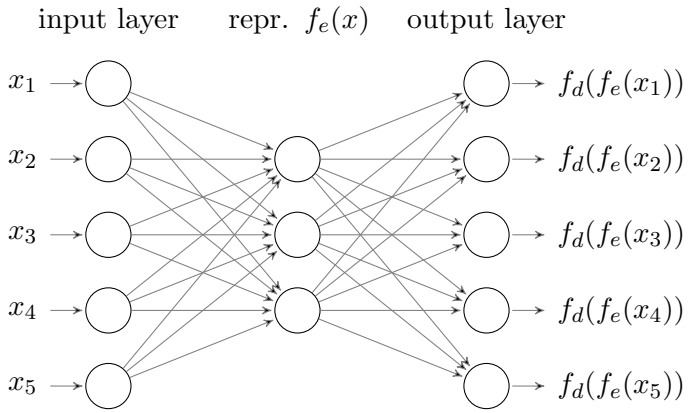


Figure 5.9: An autoencoder.

5.6 Appendix: Autoencoders and heteroencoders

Autoencoders (Baldi and Hornik, 1989; Kramer, 1991; Hinton and Salakhutdinov, 2006) are neural networks that learn to produce their own input. Autoencoders are the extension of component analysis methods to the language and tools of neural networks. Autoencoders are the composition of two functions: one encoder f_e, which maps the observed variables into the latent explanatory factors, and one decoder f_d, which maps the latent explanatory factors back into the observed variables. To learn the encoder and the decoder functions, one minimizes the reconstruction error

$$L(f_e, f_d; x) = \frac{1}{n} \sum_{i=1}^{n} \|x_i - f_d(f_e(x_i))\|,$$

where f_e and f_d are often parametrized as deep fully-connected neural networks. If the weights of the encoder neural network are equal to the transpose of the weights of the decoder neural network, we say that the encoder and the decoder have *tied* weights. Figure 5.9 illustrates an autoencoder neural network of one hidden layer, which reduces five variables into three explanatory components.

If unconstrained, autoencoders may learn to reconstruct their inputs by implementing the trivial identity map $f_d(f_e(x)) = x$. The following are some alternatives to avoid this trivial case, each of them favouring one kind of representation over another.

1. *Bottleneck* autoencoders have representations $f_e(x)$ of lower dimensionality than the one of the inputs x. The transformation computed by a linear autoencoder with a bottleneck of size $r < d$ is the projection into the subspace spanned by the first r principal components of the training data (Baldi and Hornik, 1989).

2. *Sparse* autoencoders promote sparse representations $f_e(x)$ for all x.

3. *Denoising* autoencoders (Vincent et al., 2008) corrupt the data x before passing it to the encoder, but force the decoder to reconstruct the original, clean data x. Linear denoising autoencoders are one special case of heteroencoders, which solve the CCA problem (Roweis and Brody, 1999).

4. *Contractive* autoencoders (Rifai et al., 2011) penalize the norm of the Jacobian of the encoding transformation. This forces the encoder to be contractive in the neighborhood of the data, resulting into a focused representation that better captures the directions of variation of data and ignores all others.

5. *Variational* autoencoders (Kingma and Welling, 2013) use variational inference (Section 4.2.2) to learn probabilistic encoder and decoder functions. Variational autoencoders are also generative models, as they allow the estimation of new samples from the data generating distribution.

All the previous autoencoder regularization schemes allow for overcomplete component analysis, except for bottleneck autoencoders.

Part III

Causation

Chapter 6

The language of causation

This chapter is a review of well-known results.

Chapters 4 and 5 studied the concept of *statistical dependence*. There, we learned that when two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ are *statistically dependent*, we may predict expected values for $\boldsymbol{y}$ given values for $\boldsymbol{x}$ using the *conditional expectation*

$$\mathbb{E}[\boldsymbol{y} \,|\, \boldsymbol{x} = x] \,.$$

Using the same statistical dependence, we may predict expected values for $\boldsymbol{x}$ given values for $\boldsymbol{y}$ using the opposite conditional expectation

$$\mathbb{E}[\boldsymbol{x} \,|\, \boldsymbol{y} = y] \,.$$

So, statistical dependence is a *symmetric* concept: if $\boldsymbol{x}$ is dependent to $\boldsymbol{y}$, then $\boldsymbol{y}$ is also dependent to $\boldsymbol{x}$. Like the tides in the sea and the orbit of the Moon, the luminosity and the warmth of a star, the area and radius of a circle, and the price of butter and cheese.

Yet, statistical dependences often arise due to a most fundamental *asymmetric* relationship between entities. To see this, consider the positive dependence between high levels of blood cholesterol and heart disease. This dependence arises because higher levels of blood cholesterol lead to higher chances of suffering from heart disease, but not vice versa. In everyday language, we say that "blood cholesterol *causes* heart disease". In causal relations, variations in the cause lead to variations in the effect, but variations in the effect do not lead to variations in the cause. Thus, causal relations are *asymmetric*, but all we observe in statistics are symmetric dependencies. How can we tell the difference between dependence and causation? And the difference between cause and effect?

Remark 6.1 *(Dependence does not imply causation!).* When facing two dependent random variables, it is tempting to conclude that one causes the other. The scientific literature is full of statistical dependencies misinterpreted as causal relationships.

Messerli (2012) observed a strong positive correlation between the chocolate consumption and the amount of Nobel laureates from a given country. When explaining his finding, Messerli claimed that chocolate consumption causes the sprouting of Nobel laureates. A more reasonable explanation is due to the existence of a common cause, responsible for the increase in both chocolate consumption and research budget in a given country. For instance, the socioeconomic status of the said country.

In *Nature*, Quinn et al. (1999) claimed that sleeping with intense ambient light causes the development of myopia in children. This is in fact not a causal relationship. On the contrary, a common cause, the parents of the children having myopia, is responsible for the observed association. If the parents of the child have myopia, they tend to leave the lights on at night and, at the same time, their child tends to inherit myopia.

More generally, spurious correlations occur between any two monotonically increasing or decreasing time series. One famous example is the positive association between the price of British bread and the level of Venetian seas (Sober, 2001). A dependence that, when conditioned on time, would most likely vanish. ◊

6.1 Seeing versus doing

The conditional expectation $\mathbb{E}[\boldsymbol{y} \,|\, \boldsymbol{x} = x]$ is a summary of the conditional probability distribution $P(\boldsymbol{y} \,|\, \boldsymbol{x} = x)$. We estimate this conditional expectation in two steps. First, we observe samples $S = \{(x_i, y_i)\}_{i=1}^{n}$ drawn from the joint probability distribution $P(\boldsymbol{x}, \boldsymbol{y})$. Second, we select or smooth the samples $S_x \subseteq S$ compatible with the assignment $\boldsymbol{x} = x$, and use S_x to compute the empirical average of $\boldsymbol{y}$. An analogous procedure applies to compute $\mathbb{E}[\boldsymbol{x} \,|\, \boldsymbol{y} = y]$. In both cases, the procedure is *observational*: as a *passive* agent, we *see* and filter data, from which we compute statistics.

But, there is a difference between *seeing* and *doing*. To illustrate this difference, let us now consider the case where, instead of observing one system and summarizing its behaviour whenever $\boldsymbol{x} = x$ happens, we *intervene* on the system and *actively* force $\boldsymbol{x} = x$. We denote this *intervention* by the *interventional distribution*

$$P(\boldsymbol{y} \,|\, \mathrm{do}(\boldsymbol{x} = x)). \tag{6.1}$$

The interventional distribution (6.1) is in general different from the *observational distribution* $P(\boldsymbol{y} \,|\, \boldsymbol{x} = x)$. Intuitively, the passive filtering used to compute the observational distribution does not control for the values that the common causes of $\boldsymbol{x}$ and $\boldsymbol{y}$ take. The distribution of this uncontrolled values will in turn induce a bias, which translates into differences between observational and interventional distributions. However, these biases vanish when we actively intervene on the system.

In principle, the differences between interventional and observational distributions can be arbitrarily large, even under arbitrarily small interventions. The bridge between observational and interventional distributions will be a set of assumptions about the causal structure between the random variables under study. These assumptions will, in some cases, allow us to infer properties about interventional distributions from observational distributions. This is the power of causal inference. Reasoning, just by *seeing*, the consequences of *doing*. In another words, causation allows to estimate the behaviors of a system under varying or unseen environments. We will do so by placing causal assumptions that will allow us to use observational distributions to access aspects of interventional distributions.

Example 6.1 *(The difference between seeing and doing).* Consider

$$\begin{aligned} z_i &\sim \mathcal{N}(0,1), \\ x_i &\leftarrow 5z_i, \\ y_i &\leftarrow x_i + 5z_i. \end{aligned}$$

If we draw 10^6 samples from this model, we can estimate that

$$\mathbb{E}[\boldsymbol{y} \,|\, \boldsymbol{x} = 1] \approx 2.$$

This an observational expectation. We have passively observed samples drawn from the model, and used a regression method to estimate the mean of $\boldsymbol{y}$. On the contrary, we now put our finger in the system, and perform the intervention $\mathrm{do}(\boldsymbol{x} = 1)$. Then, the intervened generative model is

$$\begin{aligned} z_i &\sim \mathcal{N}(0,1), \\ x_i &\leftarrow 1, \\ y_i &\leftarrow x_i + 5z_i, \end{aligned}$$

If we draw again 10^6 samples, we can estimate that

$$\mathbb{E}[\boldsymbol{y} \,|\, \mathrm{do}(\boldsymbol{x} = 1)] \approx 1.$$

The interventional and observational conclusions differ! ◊

Remark 6.2 *(Counterfactual reasoning).* We can read interventions like (6.1) as *contrary-to-fact* or *counterfactual* questions:

"What would have been the distribution of $\boldsymbol{y}$ had $\boldsymbol{x} = x$?"

Lewis (1974) introduced the concept of counterfactuals. Philosophically, counterfactuals assume the existence of a parallel world where everything is the same, except for the hypothetical intervention and its effects. For example, the counterfactual "had I called Paula, I would be dating her" describes an alternative world, where everything is the same as in ours,

except that I called Paula, and the effects of that call unfolded. By definition, counterfactuals are never observed, so their validity is never verified. This is a source of criticism (Dawid, 2000). In any case, counterfactuals are one concise way to state causal hypothesis. ◊

Pearl (2009a) does a great job at summarizing the distinction between statistics and causal analysis:

> "... causal analysis goes one step further; its aim is to infer not only beliefs or probabilities under static conditions, but also the dynamics of beliefs under changing conditions, for example, changes induced by treatments or external interventions. [..] An associational concept is any relationship that can be defined in terms of a joint distribution of observed variables, and a causal concept is any relationship that cannot be defined from the distribution alone. Examples of associational concepts are: correlation, regression, dependence, conditional independence, likelihood. [...] Examples of causal concepts are randomization, influence, effect, confounding, "holding constant", disturbance, spurious correlation, intervention, explanation, attribution."

In a nutshell, causation is one tool to describe the statistical behaviour of a system in changing environments, where we do not necessarily observe data from all possible environments. The question is, how can we formalize, identify, and exploit causation in learning? The answer, presented throughout the rest of this chapter, will come as a extension of the theory of probability.

Remark 6.3 *(Philosophy of causation).* In *Metaphysics*, Aristotle (384-322 BC) categorizes the causes of phenomena into material causes (what something is made of), formal causes (the form or archetype of something), efficient causes (the source of change and rest in something), and final causes (the reason why something is done). In *Novum Organum*, Francis Bacon (1606-1625) rejects the Aristotelian view, regarding it as nonscientific. Instead, the Baconian scientific method searches for conditions in which the phenomena under study occurs, does not occur, and occur in different degrees. Then, the method strips down these conditions to necessary and sufficient causes for the phenomena.

David Hume (1711-1776) had an skeptic view on causal knowledge, as described in his *A Treatise of Human Nature*. For Hume, causal relations are one form of induction from the experience of constant conjunction of events (nearby events of type A are usually followed by events of type B). But induction, from a Humean perspective, is not logically justified. Immanuel Kant (1724-1804) challenges Hume by considering causation a synthetic, objective, a priori knowledge not acquired by experience. For Kant, this a priori type of knowledge, which includes causal knowledge, is intrinsically true and shapes the world to be what it is.

Francis Galton (1822-1911) and his student Karl Pearson (1857-1936) hinted the relation between dependence and causation. When studying the relationship between the size of the human forearm and head, Galton wrote that "co-relation must be the consequence of the variations of the two organs being partly due to common causes". Hans Reichenbach (1891-1953) sharpened the relation between dependence and causation in his *Principle of Common Cause*, described in the next section.

To learn more about the philosophy of causation, we recommend the reader to consult the monograph (Beebee et al., 2009). ◊

6.2 Probabilistic causation

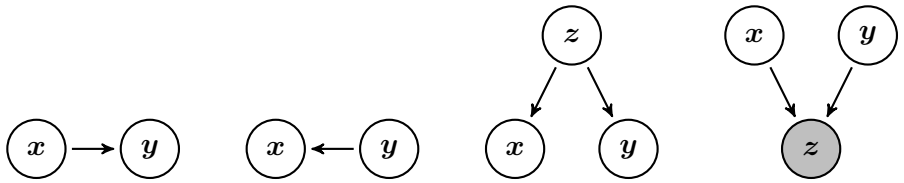
Fortunately, not all people with high levels of cholesterol suffer from heart disease. Although high levels of cholesterol increase the risk of heart disease, a number of other factors such as smoking, diet, genetics, and so forth determine experiencing a cardiovascular failure or not. This situation is easily described using a probabilistic account of causation: causes modify the *probability* of their effects happening.

The main proposition of probabilistic causation is due to Reichenbach (1956). The cornerstone of his theory is the *Principle of Common Cause* (PCC), which states that, when two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ are dependent, this is because either

1. $\boldsymbol{x}$ causes $\boldsymbol{y}$,
2. $\boldsymbol{y}$ causes $\boldsymbol{x}$,
3. there exists a third random variable $\boldsymbol{z}$ which is a common cause of $\boldsymbol{x}$ and $\boldsymbol{y}$, or
4. there exists a third random variable $\boldsymbol{z}$ which is a common effect of $\boldsymbol{x}$ and $\boldsymbol{y}$, upon which the observations are conditioned.

Figure 6.1 illustrates the four cases of the PCC. We refer to common causes as *confounders*. When confounders are unobserved, we call them *unobserved confounders*. Often, spurious correlations are due to the existence of unobserved confounders. Even worse, if the functions mapping confounders to their common effects are rich enough, hidden confounding can reproduce any observed dependence pattern.

Remark 6.4 *(Other interpretations of causation).* Probabilistic causation is not free from criticism. In (Beebee et al., 2009, Chapter 9), Jon Williamson is reluctant to model logical relationships between variables as probabilistic cause-effect relations. For example, in $\boldsymbol{z} = \mathrm{XOR}(\boldsymbol{x}, \boldsymbol{y})$, the random variables $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ are jointly independent, although both $\boldsymbol{x}$ and $\boldsymbol{y}$ are causes of $\boldsymbol{z}$. This complicates the application of the PCC. In opposition, Williamson offers



(a) (b) (c) (d)

Figure 6.1: According to Reichenbach's principle, dependencies between random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ arise because either (a) $\boldsymbol{x}$ causes $\boldsymbol{y}$, (b) $\boldsymbol{y}$ causes $\boldsymbol{x}$, (c) $\boldsymbol{x}$ and $\boldsymbol{y}$ share a common cause $\boldsymbol{z}$, (d) $\boldsymbol{x}$ and $\boldsymbol{y}$ share a common effect $\boldsymbol{z}$ on which the observations are conditioned.

an *epistemic* account of causation: causal relations are how we interpret the world, and have nothing to do with a world free from interpretation. For other interpretations of causation (and questions on the primitivism, pluralism, and dispositionalism of causation), we refer the reader to the accessible and short introduction (Mumford and Anjum, 2013). ◊

In the following, we extend language of probability theory to describe causal structures underlying high-dimensional dependence structures.

6.3 Structural equation models

This section introduces the use of structural equation models to describe causal relationships (Pearl, 2009b).

The following is a bottom-up exposition of these concepts, divided in five parts. First, we introduce the necessary notations to describe the structure of directed graphs. Second, we enumerate assumptions to link directed graphs and probability distributions defined on their nodes, to form graphical models. Third, we introduce a generalization of graphical models, termed structural equation models. Fourth, we describe the necessary assumptions to link structural equation models and the causal relationships in the real world. Fifth and last, we describe how to manipulate structural equation models to reason about the outcome of interventions and answer counterfactual questions.

6.3.1 Graphs

We borrow some of the following from (Peters, 2012, Definition 2.1).

1. A *directed graph* $G = (\mathcal{V}, \mathcal{E})$ is a set of nodes $\mathcal{V} = \{v_1, \ldots, v_d\}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V}^2$.

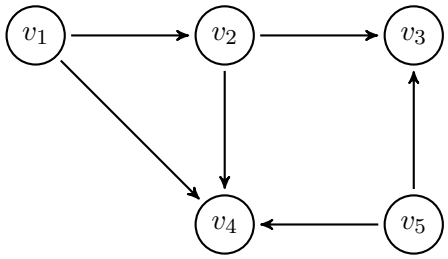


Figure 6.2: A directed acyclic graph.

2. For all $v_i, v_j \in \mathcal{V}$, $v_i \neq v_j$, we say that v_i is a *parent* of v_j if $(i,j) \in \mathcal{E}$, and we write $v_i \to v_j$. A pair of nodes (v_i, v_j) are *adjacent* if either $v_i \to v_j$ or $v_j \to v_i$, and we write $v_i - v_j$.

3. For all $v_j \in \mathcal{V}$, $\mathrm{Pa}(v_j) = \{v_i \,|\, v_i \to v_j\}$ is the set of all parents of v_j.

4. The *skeleton* of G is the set of all edges (i,j) such that $v_i \to v_j$ or $v_j \to v_i$.

5. Three nodes form a *v-structure* or *immorality* if one of them is the child of the two others, which themselves are not adjacent.

6. A *path* in G is a sequence $v_{i_1}, \ldots, v_{i_n}$ such that $v_{i_k} \to v_{i_{k+1}}$ or $v_{i_{k+1}} \to v_{i_k}$ for all $1 \leq k \leq n-1$ and $n \geq 2$.

7. A path $v_{i_1}, \ldots, v_{i_n}$ in G is a *directed path* if $v_{i_k} \to v_{i_{k+1}}$ for all $1 \leq k \leq n-1$.

8. G is a *Directed Acyclic Graph* (DAG) if it contains no directed path from v_i to itself, for all $v_i \in \mathcal{V}$.

9. A path between v_{i_1} and v_{i_n} is *blocked* by $\mathcal{Z} \subseteq \mathcal{V} \setminus \{v_{i_1}, v_{i_n}\}$ if

   - $v_{i_k} \in \mathcal{Z}$ and
     - $v_{i_{k-1}} \to v_{i_k} \to v_{i_{k+1}}$ or
     - $v_{i_{k-1}} \leftarrow v_{i_k} \leftarrow v_{i_{k+1}}$ or
     - $v_{i_{k-1}} \leftarrow v_{i_k} \to v_{i_{k+1}}$.
   - $v_{i_{k-1}} \to v_{i_k} \leftarrow v_{i_{k+1}}$ and v_{i_k} and its descendants are not in $\mathcal{Z}$.

10. Given three disjoint subsets $\mathcal{A}, \mathcal{B}, \mathcal{Z} \subseteq \mathcal{V}$, we say that $\mathcal{A}$ and $\mathcal{B}$ are *d-separated* by $\mathcal{Z}$ if all the paths between the nodes of $\mathcal{A}$ and the nodes of $\mathcal{B}$ are blocked by $\mathcal{Z}$. If so, we write $\mathcal{A} \perp\!\!\!\perp_d \mathcal{B} \,|\, \mathcal{Z}$.

Figure 6.2 shows a graph with 5 nodes and 6 edges. The graph contains a node v_4 with three parents $\mathrm{Pa}(v_4) = \{v_1, v_2, v_5\}$. The graph contains a

directed path from v_1 to v_3, which is blocked by $\mathcal{Z} = \{v_2\}$. The graph contains a blocked path from v_1 to v_5, which is unblocked under $\mathcal{Z} = \{v_4\}$. The node sets $\mathcal{A} = \{v_1\}$ and $\mathcal{B} = \{v_5\}$ are d-separated by $\mathcal{Z} = \{v_3\}$. The graph is acyclic, since there is no directed path starting and ending in the same node.

6.3.2 From graphs to graphical models

Let $G = (\mathcal{V}, \mathcal{E})$ be a DAG, and denote by $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$ a vector-valued random variable with joint probability distribution $P(\boldsymbol{x})$. For all $1 \leq i \leq d$, we associate the random variable $\boldsymbol{x}_i$ to the node $v_i \in \mathcal{V}$. Then,

1. P is *Markov* with respect to G if

$$\mathcal{A} \perp\!\!\!\perp_d \mathcal{B} \,|\, \mathcal{Z} \Rightarrow \mathcal{A} \perp\!\!\!\perp \mathcal{B} \,|\, \mathcal{Z},$$

for all disjoint sets $\mathcal{A}, \mathcal{B}, \mathcal{Z} \subseteq \mathcal{V}$. The Markov condition states that the probability distribution P embodies all the conditional independences read from the d-separations in G. The *Markov condition* enables the factorization

$$p(\boldsymbol{x}) = \prod_{i=1}^{d} p(\boldsymbol{x}_i \,|\, \mathrm{Pa}(\boldsymbol{x}_i)), \tag{6.2}$$

where p is the density function of $\boldsymbol{x}$, and $\mathrm{Pa}(\boldsymbol{x}_i)$ is the set of parents of $v_i \in \mathcal{V}$. For example, the DAG from Figure 6.2, when associated to a random variable $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_5)$, produces the Markov factorization

$$p(\boldsymbol{x} = x) = p(\boldsymbol{x}_1)\, p(\boldsymbol{x}_2 \,|\, \boldsymbol{x}_1, \boldsymbol{x}_4)\, p(\boldsymbol{x}_3 \,|\, \boldsymbol{x}_2, \boldsymbol{x}_5)\, p(\boldsymbol{x}_4 \,|\, \boldsymbol{x}_1, \boldsymbol{x}_5)\, p(\boldsymbol{x}_5).$$

Nevertheless, the probability distribution P may contain further conditional independences not depicted in the d-separations from G. This nuance is taken care by the faithfulness condition, stated next.

2. P is *faithful* to G if

$$\mathcal{A} \perp\!\!\!\perp_d \mathcal{B} \,|\, \mathcal{Z} \Leftarrow \mathcal{A} \perp\!\!\!\perp \mathcal{B} \,|\, \mathcal{Z},$$

for all disjoint sets $\mathcal{A}, \mathcal{B}, \mathcal{Z} \subseteq \mathcal{V}$. The *faithfulness condition* forces the probability distribution P to not embody any further conditional independences other than those encoded by the d-separations associated with the graphical structure of G. For example, the distribution

$$p(\boldsymbol{x}_1)\, p(\boldsymbol{x}_2 \,|\, \boldsymbol{x}_1)\, p(\boldsymbol{x}_3 \,|\, \boldsymbol{x}_2, \boldsymbol{x}_5)\, p(\boldsymbol{x}_4 \,|\, \boldsymbol{x}_1, \boldsymbol{x}_5)\, p(\boldsymbol{x}_5)$$

is unfaithful to the DAG in Figure 6.2, since the conditional independence $x_2 \perp\!\!\!\perp x_4 \,|\, x_1$ does not follow from the structure of the graph. This conditional independence, not depicted in the graph G, may be

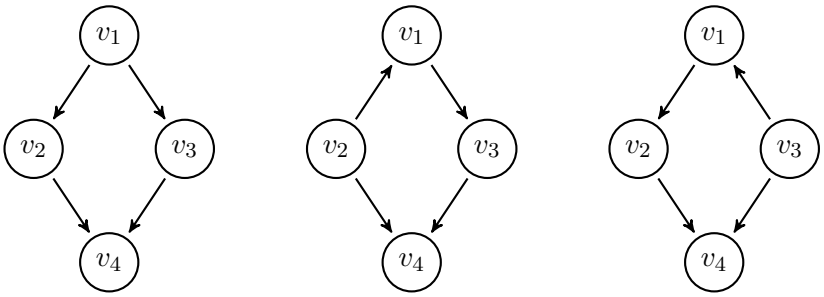


Figure 6.3: Three Markov equivalent DAGs.

due to the cancellation between the effect of x_1 on x_2 and the effect of x_4 on x_2. Faithfulness is in charge of protecting us from the existence of such spurious independences.

3. The pair (G, P) satisfies the *minimality condition* if it satisfies the Markov condition, but any pair (G', P), where G' is a graph obtained by removing edges from G, does not satisfy the Markov condition. Faithfulness implies minimality, but not vice versa.

4. We denote by

$$\text{Markov}(G) = \{P \mid P \text{ is Markov with respect to } G\}$$

the *Markov equivalence class* of G. We say that two DAGs G_1 are G_2 are *Markov equivalent* if $\text{Markov}(G_1) = \text{Markov}(G_2)$. Two graphs are Markov equivalent if they have the same skeleton and set of immoralities (Verma and Pearl, 1991).

Figure 6.3 illustrates three different but Markov equivalent DAGs. These three graphs entail the same d-separations, or equivalently, share the same skeleton and set of v-structures.

5. If P is Markov with respect to G, we call the tuple (G, P) a *graphical model*.

In short, the Markov condition says that every conditional independence described by the DAG is present in the probability distribution. Since different DAGs can entail the same set of conditional independences, the Markov condition is insufficient to distinguish between Markov equivalent DAGs. The faithfulness condition assumes more to resolve this issue, saying that no conditional independence other than the ones described by the graph is present in the probability distribution. In situations where the faithfulness condition is too restrictive, we may use the minimality condition instead.

6.3.3 From graphical models to structural equation models

A *Structural Equation Model* or SEM (Wright, 1921) is a pair $(\mathcal{S}, Q(\boldsymbol{n}))$, or simply $(\mathcal{S}, Q)$, where $\mathcal{S} = \{S_1, \ldots, S_d\}$ is a set of equations

$$S_i : \boldsymbol{x}_i = f_i(\mathrm{Pa}(\boldsymbol{x}_i), \boldsymbol{n}_i),$$

and $\boldsymbol{n} = (\boldsymbol{n}_1, \ldots, \boldsymbol{n}_d)$ is a vector of d independent *noise* or *exogenous* random variables, following the probability distribution $Q(\boldsymbol{n})$. If the functions f_i are free form, call the SEM a *nonparametric structural equation model*. On the contrary, if we place assumptions on the shape of these functions, call the SEM a *restricted structural equation model*. Wright (1921) introduced structural equation models to describe biological systems, and restricted the functions f_i to be linear.

One can map structural equation models $(\mathcal{S}, Q)$ to graphical models (G, P) as follows. First, construct the graph $G = (\mathcal{V}, \mathcal{E})$ by associating the output $\boldsymbol{x}_i$ in equation $S_i \in \mathcal{S}$ to the node $v_i \in \mathcal{V}$, and drawing an edge $(j, i) \in \mathcal{E}$ from each $v_j \in \mathrm{Pa}(\boldsymbol{x}_i)$ to v_i. Second, construct the probability distribution $P(\boldsymbol{x})$ by choosing the distributions of each of the exogenous variables $\boldsymbol{n}_1, \ldots, \boldsymbol{n}_d$. Propagating these distributions using the equations $\mathcal{S}$ produces the distributions of each of the random variables $\boldsymbol{x}_i$, jointly described by P. The mapping induces a distribution P Markov with respect to the graph G (Pearl, 2009b, theorem 1.4.1). Different structural equation models can map to the same graphical model or, the mapping from structural equation models to graphical models is surjective. Simply put, structural equation models contain strictly more information than graphical models (Peters, 2012).

6.3.4 From structural equation models to causation

Up to know, we have described the abstract concepts of directed acyclic graph and probability distribution, how to merge them together into a graphical model, and how graphical models relate to structural equation models. Yet, none of these have causal meaning, let alone model causal relationships shaping the real world.

Given a graphical model (G, P), the DAG G describes the conditional independences embodied in the probability distribution P, and allows the factorization (6.2). Although tempting, the directed edges $\boldsymbol{x}_i \to \boldsymbol{x}_j$ do not always bear the causal interpretation "$\boldsymbol{x}_i$ causes $\boldsymbol{x}_j$". Graphs are just abstract tools that, together with the Markov assumption, talk about conditional independences in distributions. Different Markov equivalent DAGs state the same conditional independences, but the orientation of some of their edges can differ. This discrepancy may lead to wrong causal claims, under a premature causal interpretation of the edges in the graph.

The causal relationships between a collection of random variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d$ are formalized as a DAG by placing two assumptions (Dawid, 2010).

1. The *representational assumption* or, the causal structure of $\boldsymbol{x}$ indeed admits an *causal DAG* G_0. The representational assumption discards the consideration of cyclic graphs.

2. The *causal Markov condition* or, the d-separations in G_0 are embodied as conditional independences in the distribution $P(\boldsymbol{x})$.

So, when the DAG G_0 turns out to be the true causal structure of P, we rename the Markov condition as the *causal Markov condition*. This new condition establishes the causal meaning of the arrows in the graph G_0, and allows to draw causal inferences from properties of conditional independence. The causal Markov condition states that the edge $\boldsymbol{x}_i \to \boldsymbol{x}_j$ means "$\boldsymbol{x}_i$ causes $\boldsymbol{x}_j$", or that "$\mathrm{Pa}(\boldsymbol{x}_i)$ are the direct causes of $\boldsymbol{x}_i$". Furthermore, the factorization (6.2) carries the semantics "variables are independent when conditioned to their direct causes".

Armed with the causal Markov condition, we can also define *causal* structural equation models $(\mathcal{S}, Q)$, with $\mathcal{S} = \{S_1, \ldots, S_d\}$, where the equations

$$S_i : \boldsymbol{x}_i = f_i(\mathrm{Pa}(\boldsymbol{x}_i), \boldsymbol{n}_i)$$

are now endowed with the causal interpretation "the causes of $\boldsymbol{x}_i$ are $\mathrm{Pa}(\boldsymbol{x}_j)$". This is the most important distinction between a regular graphical model, like a Bayesian network, and a causal graphical model. While Bayesian networks are abstract descriptions of the conditional independences embodied in a probability distribution, causal graphical models are explicit descriptions of real-world processes, and their arrows describe the causal effects of performing real-world interventions or experiments on their variables.

As it happened with conditional independence, we can further ease causal inference by placing additional, stronger assumptions (Pearl, 2009b).

1. The *causal faithfulness condition* or, the causal DAG G_0 is faithful to the distribution $P(\boldsymbol{x})$.

2. The *causal minimality condition* or, the pair (G_0, P) satisfies the minimality condition. Causal faithfulness implies causal minimality.

3. The *causal sufficiency assumption* or, the inexistence of unmeasured variables $\boldsymbol{x}_0$ causing any of the measured variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d$.

Although we have made some progress in the formalization of causation, we have not yet formalized what we mean by "$\boldsymbol{x}$ causes $\boldsymbol{y}$" or, what properties does the true causal DAG G_0 must satisfy in relation with the real world causal relations. The next section resolves this issue in terms of *interventions*.

6.3.5 From causation to the real world

We set two assumptions about how the world will react with respect to interventions (Pearl, 2009b; Dawid, 2010).

1. The *locality condition* or, under any intervention over the set of variables $\mathcal{A} \subseteq \mathcal{V}$, the distribution of the variables $\mathcal{B} = \mathcal{V} \setminus \mathcal{A}$ depends only on $\mathrm{Pa}(\mathcal{B})$, as given by the causal DAG $G_0 = (\mathcal{V}, \mathcal{E})$.

2. The *modularity condition* or, for all $1 \leq i \leq d$, the conditional distribution $p(\boldsymbol{x}_i \,|\, \mathrm{Pa}(\boldsymbol{x}_i))$ is invariant with respect to any interventions made on the variables $\boldsymbol{x} \setminus \boldsymbol{x}_i$.

Let us see what these assumptions entail. As usual, denote by $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x})$ be a random variable with probability distribution P, and density or mass function p. Let $\mathcal{K} \subseteq \{1, \ldots, d\}$ be the subset of the random variables over which we perform the interventions $\{\mathrm{do}(\boldsymbol{x}_k = q_k(\boldsymbol{x}_k))\}_{k \in \mathcal{K}}$, using some set of probability density or mass functions $\{q_k\}_{k \in \mathcal{K}}$. Then, using the locality and modularity conditions, we obtain the *truncated factorization*

$$p(\boldsymbol{x} \,|\, \{\mathrm{do}(\boldsymbol{x}_k = q_k(\boldsymbol{x}_k))\}_{k \in \mathcal{K}}) = \prod_{k \notin \mathcal{K}}^{d} p(\boldsymbol{x}_k \,|\, \mathrm{Pa}(\boldsymbol{x}_k)) \prod_{k \in \mathcal{K}} q_k(\boldsymbol{x}_k).$$

In this equation, we are forcing the random variables in $\mathcal{K}$ to follow the interventional distributions $q_k(\boldsymbol{x}_k)$. The rest of the variables and their conditional probability distributions remain unchanged, due to the locality and modularity conditions. When we intervene on a variable $\boldsymbol{x}_k$, the effects from $\mathrm{Pa}(\boldsymbol{x}_k)$ into $\boldsymbol{x}_k$ are no longer present in the truncated factorization. A corollary of this is that intervening on variables without parents is the same as conditioning on those variables, in the observational sense.

In the most common type of intervention, where we set the random variable $\boldsymbol{x}_k = x_k$, the density or mass function $q_k = \delta_k$ (Pearl, 2009b; Peters, 2012). Using the Markov and minimality conditions, together with the concept of truncated factorizations, we are now ready to define the *true causal DAG* associated with a probability distribution $P(\boldsymbol{x})$.

Definition 6.1 (True causal DAG)**.** *The DAG G_0 is the* true causal DAG *of the probability distribution $P(\boldsymbol{x})$ if G_0 satisfies the Markov and minimality conditions, and produces a truncated factorization that coincides with $p(\boldsymbol{x}|\{do(\boldsymbol{x}_k = q_k(\boldsymbol{x}_k)\}_{k \in \mathcal{K}})$ for all interventions $\{do(\boldsymbol{x}_k = q_k(\boldsymbol{x}_k)\}_{k \in \mathcal{K}}$ possible in the real-world system described by P (Peters, 2012, Def. 1.3).*

We now describe how to perform interventions in structural equation models $(\mathcal{S}, Q)$. The intervened structural equation model $(\tilde{\mathcal{S}}, Q)$ associated with the set of interventions $\{\mathrm{do}(\boldsymbol{x}_k = q(\boldsymbol{x}_k))\}_{k \in \mathcal{K}}$ is constructed by replacing the equations $S_k \in \mathcal{S}$ with the equations $\tilde{S}_k : \boldsymbol{x}_k = q(\boldsymbol{x}_k)$ in $\tilde{\mathcal{S}}$, for all $k \in \mathcal{K}$. Thus, intervening the variable $\boldsymbol{x}_k$ in a structural equation model amounts to setting such variable to be exogenous, and distributed according to the probability density or mass function q_k. The intervened SEM induces an intervened graphical model $(\tilde{G}, \tilde{P})$, Moreover, if $(\tilde{G}, \tilde{P})$ satisfies the conditions

treatment	all stones	small stones	large stones
A	78% (273/350)	93% (81/87)	73% (192/263)
B	83% (289/350)	87% (234/270)	69% (55/80)

Table 6.1: Data for the kidney stones example.

from Definition 6.1 for all possible interventions, then the graph associated with the SEM $(\mathcal{S}, Q)$ is the true causal DAG G_0.

As emphasized in the introduction of this chapter, intervening and observing a system are disparate things. While intervening modifies the mechanisms of the underlying causal graph and generates a new different probability distribution, observing amounts to passively filtering samples from the joint distribution and then computing statistics using those filtered samples. We finally have the tools to illustrate this difference formally. In the following example, we intervene in a SEM to analyze its responses, or equivalently, answer counterfactual questions. We borrow the example from Peters (2015, example 3.1.1).

Example 6.2 *(Kidney stones).* Table 6.1 summarizes the success rates of two different treatments for two different sizes of kidney stones, when tested on 350 patients each (Peters, 2015). In the following, let the binary random variables $\boldsymbol{s}$, $\boldsymbol{t}$ and $\boldsymbol{r}$ mean "kidney stone size", "treatment received", and "patient recovered". Overall, treatment B seems to be more successful, since

$$\begin{aligned} \mathbb{P}(\boldsymbol{r} = 1 \,|\, \boldsymbol{t} = A) &= 0.78, \\ \mathbb{P}(\boldsymbol{r} = 1 \,|\, \boldsymbol{t} = B) &= 0.83. \end{aligned} \tag{6.3}$$

Nevertheless, treatment A is more successful than treatment B for patients with both small and large kidney stones, when examined separately. This is a classic example of Simpson's paradox: a result appearing in different groups of data reverses when analyzing the groups combined. This is confusing, so, in the unfortunate event of having kidney stones of *unknown* size, what treatment should I prefer?

To answer this question, assume the causal graph in Figure 6.4a. We want to characterize the interventional probability mass function

$$p_t(\boldsymbol{r}) := p(\boldsymbol{r} \,|\, \mathrm{do}(\boldsymbol{t} = t)),$$

for $t \in \{0, 1\}$. To do so, we amputate the causal graph from Figure 6.4a by removing the edge $\boldsymbol{s} \to \boldsymbol{t}$, and construct two new causal graphs G_A and G_B corresponding to hold $\boldsymbol{t} = A$ and $\boldsymbol{t} = B$ constant. Figure 6.4b shows the intervened graph. The two new probability distributions induced by these

(a) Assumed causal graph. (b) Intervened causal graph.

Figure 6.4: Causal graph for the kidney stones example.

two different interventions are P_A and P_B. Then,

$$\begin{aligned} p_A(\boldsymbol{r}=1) &= \sum_s p_A(\boldsymbol{r}=1, \boldsymbol{t}=A, \boldsymbol{s}=s) \\ &= \sum_s p_A(\boldsymbol{r}=1 \,|\, \boldsymbol{t}=A, \boldsymbol{s}=s) p_A(\boldsymbol{t}=A, \boldsymbol{s}=s) \\ &= \sum_s p_A(\boldsymbol{r}=1 \,|\, \boldsymbol{t}=A, \boldsymbol{s}=s) p(\boldsymbol{s}=s) \\ &= \sum_s p(\boldsymbol{r}=1 \,|\, \boldsymbol{t}=A, \boldsymbol{s}=s) p(\boldsymbol{s}=s) \end{aligned}$$

where the last two steps follow by the relation $\boldsymbol{t} \perp\!\!\!\perp \boldsymbol{s}$ in the amputated graph, and the truncated factorization rule. Using analogous computations for $\boldsymbol{t}=B$ and the data in Table 6.1, we estimate:

$$\begin{aligned} p_A(\boldsymbol{r}=1) &= \sum_s p(\boldsymbol{r}=1 \,|\, \boldsymbol{t}=B, \boldsymbol{s}=s) p(\boldsymbol{s}=s) \approx 0.832, \qquad (6.4) \\ p_B(\boldsymbol{r}=1) &= \sum_s p(\boldsymbol{r}=1 \,|\, \boldsymbol{t}=B, \boldsymbol{s}=s) p(\boldsymbol{s}=s) \approx 0.782. \end{aligned}$$

Therefore, we should prefer to receive treatment A. The opposite decision (Equations 6.4 and 6.3) from the one taken by just observing the data! ◊

Given the true causal DAG of a system, we use truncated factorizations to express interventional distributions as observational distributions. This avoids the need of intervening on a system, which is often impractical.

Counterfactual reasoning is also tool to design interventions to maximize particular statistics of the intervened distribution, such as recovery rates with respect to patient treatments, or revenue with respect to business decisions.

Remark 6.5 *(Criticism on DAGs).* The use of DAGs to describe the causal structure of multivariate systems is not free of criticism. Dawid (2010) surveys some of the shortcomings of DAGs for causal modeling, emphasizing the amount and strength of assumptions necessary to guarantee the correctness of the counterfactual answers produced from them. As an alternative, Dawid (2010) suggests a generalization of DAGs termed *augmented DAGs*, where

interventions are additional nodes in the DAG, and the concept of conditional independence generalizes to deal with these new types of nodes.

A second criticism on the use of DAGs is their inherent incapacity to model dynamical systems with causal cycles, such as feedback loops. Those cycles exist, for instance, in protein interaction networks. We describe two solutions to the problem. First, to sample the dynamical system over time, and unroll the causal cycles into duplicate graph nodes corresponding to the same variable at different points in time. Second, to assume that data follows the equilibrium distribution of the dynamical system. For more details, consult (Mooij et al., 2011). ◊

6.4 Observational causal inference

The previous section assumed the knowledge of the true causal DAG G_0, the graph governing the causal mechanics of the system under study, the graph giving rise to the data generating distribution $P(\boldsymbol{x})$. If we know the true causal DAG G_0, we can answer counterfactual questions about the potential outcome of interventions by using truncated factorizations. But, what if we do not know G_0?

The gold standard to infer G_0 is to perform Randomized Controlled Trials (RCTs). Consider the question "Does aspirin cause relief from headache?". To answer such causal question using an RCT, we first gather a large number of patients suffering from headaches, but equal in all their other characteristics. Second, we divide the patients into two groups, the treatment group and the control group. Next, to every person in the treatment group, we supply with an aspirin pill. To every person in the control group, we supply with a placebo. Finally, we study the relief rate in each of the two groups, and determine if the difference between the recovery rate within the two groups is statistically significant. If it is, we conclude that the aspirin has an effect on relieving headaches.

Unfortunately, RCTs are often expensive, unethical, or impossible to perform: it is expensive to perform RCTs that extend over years, it is unethical to supply experimental drugs to humans, and it is impossible to reverse the rotation of the Earth. Therefore, in these situations, we face the need of inferring causal relationships from an observational position, by seeing but not doing.

Observational causal inference is the problem of recovering the true causal DAG G_0 associated with the probability distribution P, given only samples from P. In the rest of this section, we review assumptions and algorithms used for observational causal inference, as well as their limitations.

Remark 6.6 *(Causal inference as a missing data problem)***.** In our example RCT, we record each patient under one of the two possible *potential outcomes*: either they took aspirin or placebo, but never both. Instead, we could imagine

that for each patient, we have two records: the observed record associated with the assigned treatment, and the counterfactual record associated to the treatment that was not assigned to the patient. Thus, the problem of causal inference is to some extent a problem of missing data, where we must complete the counterfactual records. The Neyman-Rubin causal model builds on this idea to develop causal inference techniques, such as *propensity score matching* (Rosenbaum and Rubin, 1983), to perform causal inference in both interventional and observational data. ◊

6.4.1 Assumptions

The problem of observational causal inference is impossible without restricting the class of structural equation models under study. Even when enforcing the causal Markov condition, any distribution P is Markov with respect to a large number of different graphs. Therefore, all we can hope for is to recover a Markov equivalence class —the skeleton and the immoralities of the true causal graph, lacking the orientation of some arrows— even when using an infinite amount of data. In these situations, we say that the true underlying causal graph is *not identifiable*. But, we may be able to recover a set of causal graphs which agrees with the observed data, and contains the true causal graph. As investigated by the different algorithms reviewed below, placing further assumptions on P reduces the size of the equivalence class of graphs identifiable from data. The problem is *identifiable* if our assumptions allow us to uniquely recover the true causal graph uniquely.

In a nutshell, the precision of the recovery of the true underlying causal graph is inversely proportional to the number and strength of assumptions that we are able to encode in the causal inference problem at hand.

Independence of cause and mechanism

Section 6.3 described how to exploit conditional independences to infer causal properties about the data under study. But, conditional independence is not always applicable. For example, consider observational causal inference in a system formed by two random variables, $\boldsymbol{x}$ and $\boldsymbol{y}$. Here, our goal is to decide whether $\boldsymbol{x} \to \boldsymbol{y}$ or $\boldsymbol{x} \leftarrow \boldsymbol{y}$. Unfortunately, the absence of a third random variable prevents us from measuring conditional independences, as prescribed in Section 6.3. Because of this, the research community has developed principles for causal inference not based on conditional independence. In the following, we present a widely used principle, the Independence between Cause and Mechanism (ICM) assumption, useful to perform observational cause effect inference in the two-variable case.

To motivate the ICM assumption, recall that the joint probability distribution of two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ admits the two conditional

decompositions

$$\begin{aligned} p(\boldsymbol{x}, \boldsymbol{y}) &= p(\boldsymbol{y} \,|\, \boldsymbol{x}) p(\boldsymbol{x}) \\ &= p(\boldsymbol{x} \,|\, \boldsymbol{y}) p(\boldsymbol{y}), \end{aligned}$$

where we may interpret the conditional distribution $p(\boldsymbol{y} \,|\, \boldsymbol{x})$ as a causal mechanism mapping the cause $\boldsymbol{x}$ to its effect $\boldsymbol{y}$, and the conditional distribution $p(\boldsymbol{x} \,|\, \boldsymbol{y})$ as a causal mechanism mapping the cause $\boldsymbol{y}$ to its effect $\boldsymbol{x}$. Which of the two conditional distributions should we prefer as the true causal mechanism?

In the spirit of Occam's razor, we prefer the conditional decomposition that provides with the shortest description of the causal structure contained in the joint distribution $p(\boldsymbol{x}, \boldsymbol{y})$ (Lemeire and Dirkx, 2006). In terms of algorithmic information theory, the conditional decomposition with algorithmically independent factors has a lower Kolmogorov complexity (Janzing and Schölkopf, 2010). For instance, if the two distributions $p(\boldsymbol{y} \,|\, \boldsymbol{x})$ and $p(\boldsymbol{x})$ are "independent", then the shortest description of $p(\boldsymbol{x}, \boldsymbol{y})$ is the conditional decomposition $p(\boldsymbol{y} \,|\, \boldsymbol{x})p(\boldsymbol{x})$, and we should prefer the causal explanation $\boldsymbol{x} \to \boldsymbol{y}$ to describe the joint distribution $p(\boldsymbol{x}, \boldsymbol{y})$. In short,

> we prefer the causal direction under which the distribution of the cause is independent from the mechanism mapping the cause to the effect. (ICM)

The ICM assumption is often violated in the incorrect causal direction. In our example, this means that if the factors $p(\boldsymbol{x})$ and $p(\boldsymbol{y} \,|\, \boldsymbol{x})$ are "independent", then this will be not the case for the factors $p(\boldsymbol{y})$ and $p(\boldsymbol{x} \,|\, \boldsymbol{y})$ (Schölkopf et al., 2012). This asymmetry renders the observational causal inference possible.

In the previous paragraph, the word *independence* appears in scare quotes. This is because it is not obvious how to measure dependence between distributions and functions in full generality. Nevertheless, the next section reviews some algorithms where, thanks to parametric assumptions on the conditional decompositions, the ICM assumption becomes statistically testable.

Example 6.3 *(Limits of the ICM assumption).* The intuition behind the ICM assumption is that laws in Nature are fixed and therefore independent to what we feed into them. Although the ICM assumption enjoys this natural interpretation, it does not hold whenever $\boldsymbol{x} \to \boldsymbol{y}$ but $\boldsymbol{y} = f(p(\boldsymbol{x}))$ is some statistic of the distribution $p(\boldsymbol{x})$. For example, the spatial probability distribution of precious stones causes their price, the probability of a poker hand causes its expected reward, and the probability of genetic mutations cause the average phenotype expression of a population. $\lozenge$

6.4.2 Algorithms

In the following, we review a collection of algorithms for observational causal inference. The algorithms differ on how they operate, and the assumptions that they place to guarantee their correctness.

Conditional independence methods

The Spirtes-Glymour-Scheines (SGS) algorithm (Spirtes et al., 2000) assumes the representational, causal Markov, sufficiency, and faithfulness conditions, but does not place any assumption on the relationships between variables. Furthermore, SGS assumes the faithfulness condition between the data generating distribution $P(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$ and the true causal graph G. The SGS algorithm works as follows:

1. Build $K = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d\}$ and $(i, j), (j, i) \in \mathcal{E}$, for all $1 \leq i, j \leq d$.
2. For each pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$, if $\exists \mathcal{Z} \subseteq \mathcal{V} \setminus \{\boldsymbol{x}_i, \boldsymbol{x}_j\}$ such that $\boldsymbol{x}_i \perp\!\!\!\perp_d \boldsymbol{x}_j \,|\, \mathcal{Z}$, remove (i, j) from $\mathcal{E}$.
3. For each structure $\boldsymbol{x}_i - \boldsymbol{x}_j - \boldsymbol{x}_k$ with $(i, k) \notin \mathcal{E}$ and no $\mathcal{Z} \subseteq \boldsymbol{x}_j \cup \mathcal{V} \setminus \{\boldsymbol{x}_i, \boldsymbol{x}_k\}$ such that $\boldsymbol{x}_i \perp\!\!\!\perp_d \boldsymbol{x}_k \,|\, \mathcal{Z}$, remove (j, i) and (j, k) from $\mathcal{E}$.
4. Until no more edges get removed from $\mathcal{E}$, repeat
   (a) if $\boldsymbol{x}_i \to \boldsymbol{x}_j - \boldsymbol{x}_k$, $\boldsymbol{x}_i \not\to \boldsymbol{x}_k$, and $\boldsymbol{x}_i \not\leftarrow \boldsymbol{x}_k$, then remove (k, j) from $\mathcal{E}$.
   (b) if there is a directed path from $\boldsymbol{x}_i$ to $\boldsymbol{x}_k$, and $\boldsymbol{x}_i \to \boldsymbol{x}_k$, remove (k, j) from $\mathcal{E}$.

The second step of the SGS algorithm performs a conditional independence test for all possible conditioning sets $\mathcal{Z} \subseteq \mathcal{V} \setminus \{\boldsymbol{x}_i, \boldsymbol{x}_j\}$ and pair of distinct nodes $(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Thus, for a node set $\mathcal{V}$ of d nodes, SGS performs 2^{d-2} conditional independence tests for each pair of distinct nodes. For large d, this exponential amount of conditional independence tests is prohibitive, both computationally and statistically. Computationally, because each conditional independence test takes a nontrivial amount of computation. Statistically, because conditional independence tests with limited data and high-dimensional conditioning sets suffer from the curse of dimensionality.

Because of these reasons, the SGS algorithm evolved into the PC algorithm, which exploits a clever sorting of the variables to reduce the amount of necessary conditional independence tests. For some problems, the PC algorithm can not improve the computational complexity of the SGS algorithm. The FCI algorithm is an extension of the SGS/PC algorithm to deal with *insufficiency*: causal inference on the presence unobserved confounders (Spirtes et al., 2000).

The SGS is *universally* consistent —able to recover the Markov equivalence class containing the true causal DAG for all P— but not *uniformly* consistent —there exists no upper bound on how fast SGS recovers such result as the amount of available data increases. In fact, no causal inference algorithm can be both universally and uniformly consistent. To achieve uniform (but not universal) consistency, it is necessary to strengthen the faithfulness assumption (for further discussion and references, see Peters (2012)).

Score methods

Score methods (Heckerman et al., 1997) construct a mapping from parameter vectors $\theta \in \mathbb{R}^m$ to the set of DAGs on d nodes, and evaluate the score of each candidate by using the posterior distribution

$$p(\boldsymbol{\theta} = \theta | x_1, \ldots, x_n) = \frac{p(x_1, \ldots, x_n | \boldsymbol{\theta} = \theta) p(\boldsymbol{\theta} = \theta)}{p(x_1, \ldots, x_n)}, \tag{6.5}$$

where the prior distribution $p(\boldsymbol{\theta})$ incorporates the available prior knowledge to favour some DAG structures over others, and the likelihood distribution $p(x_1, \ldots, x_n | \theta)$ measures how well does a given DAG, parametrized by the parameter vector θ, explain the data $x_1, \ldots, x_n$, where $x_i \in \mathbb{R}^d$ for all $1 \leq i \leq n$. For instance, the prior distribution can favor sparse DAGs, simple conditional distributions, and known independences in the factorization of the data distribution. Score methods return the DAG corresponding to the parameter vector maximizing the posterior distribution (6.5) as the true causal DAG generating the data. One must choose prior and likelihood distributions that allow for efficient posterior inference; this restriction, in turn, translates into additional assumptions about the true causal graph under search.

Additive noise models

The family of Additive Noise Models (ANM) assumes structural equation models $(\mathcal{S}, Q)$ with a set of equations $\mathcal{S} = (S_1, \ldots, S_d)$ of form

$$S_i : \boldsymbol{x}_i = f_i(\mathrm{Pa}(\boldsymbol{x}_i)) + \boldsymbol{n}_i,$$

where the exogenous or noise variables $\boldsymbol{n}_i$ and functions f_i are absolutely continuous with respect to the Lebesgue measure for all $1 \leq i \leq d$.

The identifiability of additive noise models calls for additional assumptions, either on the shape of the functions f_i, or the distribution of the independent noise variables $\boldsymbol{n}_i$. Additive noise models are identifiable when the representational, sufficiency, and causal Markov assumptions hold, and

1. the functions f_i are linear with nonzero coefficients, and the noise variables $\boldsymbol{n}_i$ are non-Gaussian (Shimizu et al., 2006), or

2. the functions f_j are smooth and nonlinear, and the densities of both the equation outputs $\boldsymbol{x}_i$ and noise variables $\boldsymbol{n}_i$ are strictly positive and smooth (Peters et al., 2014, condition 19).

On the one hand, the identifiability of the first point above is due to Independent Component Analysis (ICA), proved using the Darmois-Skitovič theorem, and does not require the faithfulness condition (Shimizu et al., 2006). On the other hand, the identifiability of the second point above requires a mild technical assumption (Hoyer et al., 2009, Theorem 1), and the causal minimality condition. These results do not rely on conditional dependencies, so they apply to the case where the causal DAG has only two variables. The identifiability result in both cases full: we can not only recover the Markov equivalence class containing the true causal DAG, but the true causal DAG itself.

The statistical footprint revealing the direction of causation in additive noise models is the dependence structure between the cause and noise variables. More specifically, given two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ with causal relation $\boldsymbol{x} \to \boldsymbol{y}$, if we assume the previous conditions there exists an additive noise model

$$\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{n},$$

in the correct causal direction, but there exists no additive noise model

$$\boldsymbol{x} = g(\boldsymbol{y}) + \boldsymbol{n}',$$

in the anticausal direction. Due to the definition of the additive noise model, this means that $\boldsymbol{x} \perp\!\!\!\perp \boldsymbol{n}$, but it cannot be the case that $\boldsymbol{y} \perp\!\!\!\perp \boldsymbol{n}'$.

Given a consistent nonparametric regression method and a consistent nonparametric independence test (such as the ones reviewed in Section 5.2), it is possible to decide whether $\boldsymbol{x} \to \boldsymbol{y}$ or $\boldsymbol{x} \leftarrow \boldsymbol{y}$ on the basis of empirical data $\{(x_i, y_i)\}_{i=1}^n \sim P^n(\boldsymbol{x}, \boldsymbol{y})$, as n tends to infinity. Under each of the two possible causal directions, proceed by computing a regression function from one variable to the other, and then testing for independence between the input variable and the obtained regression residuals. The independence tests can be replaced with Gaussianity tests, to discover both linear and nonlinear causal relationships (Hernández-Lobato et al., 2016).

The additive noise model is not identifiable for structural equations with linear functions and Gaussian exogenous variables. We now exemplify this phenomena in the case of two random variables:

- In the four plots from the left half of Figure 6.5, we have a cause variable $\boldsymbol{x} \equiv \mathcal{N}$, a noise variable $\boldsymbol{n} \equiv \mathcal{N}(0, 1)$, and a effect variable $\boldsymbol{y} \leftarrow 2\boldsymbol{x} + \boldsymbol{n}$. In this setup, the joint distribution $P(\boldsymbol{x}, \boldsymbol{y})$ is also Gaussian. This means that the joint distribution is elliptical, and that there exists no asymmetry that we could exploit to infer the direction of causation between $\boldsymbol{x}$ and $\boldsymbol{y}$. Thus, the data admits an additive noise model in

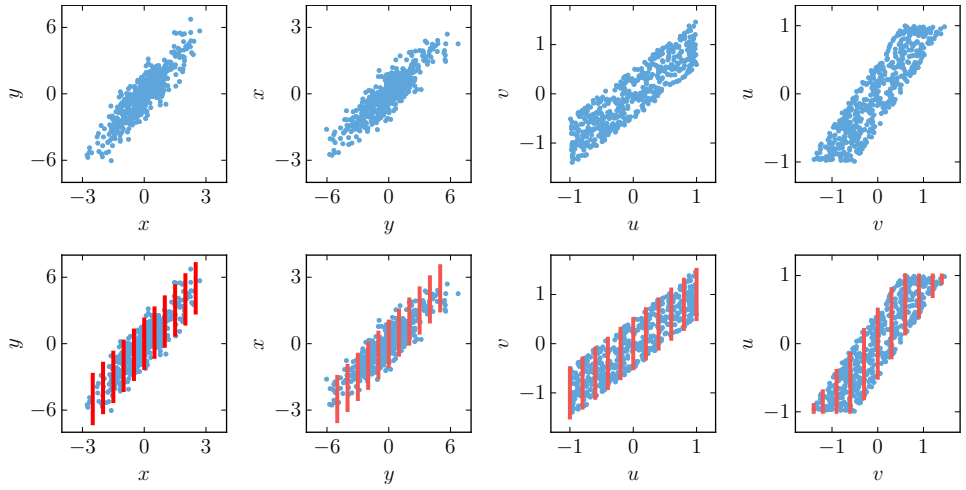


Figure 6.5: Examples of linear additive noise models.

both directions, since the regression noise (depicted as red bars) is always independent from the alleged cause.

- In the four plots from the right half of Figure 6.5, we have a cause variable $\boldsymbol{u} \equiv \mathcal{U}[-1, +1]$, a noise variable $\boldsymbol{e} \equiv \mathcal{U}[-1, +1]$, and an effect variable $\boldsymbol{v} \leftarrow \boldsymbol{u} + 0.5\boldsymbol{e}$. Therefore, this setup falls under the identifiability conditions of Shimizu et al. (2006), since the data does not admit an additive noise model in the incorrect causal direction $\boldsymbol{u} \leftarrow \boldsymbol{v}$. We see this because the regression noise (depicted as red bars) is dependent from the alleged cause $\boldsymbol{v}$: its variance peaks at $\boldsymbol{v} = 0$, and shrinks as the absolute value of $\boldsymbol{v}$ increases. This asymmetry renders causal inference possible from observing the statistics of the data.

Additive noise models are consistent (Kpotufe et al., 2014), and there exists extensions to discrete variables (Peters et al., 2011), latent variables (Stegle et al., 2010), cyclic graphs (Mooij et al., 2011; Lacerda et al., 2012), and postnonlinear equations $\boldsymbol{x}_i = g_i(f_i(\mathrm{Pa}(\boldsymbol{x}_i)) + \boldsymbol{n}_i)$, where $g_i : \mathbb{R} \to \mathbb{R}$ is an additional monotone function (Zhang and Hyvärinen, 2009). Some of these extensions, however, sacrifice the identifiability of the problem up to the true causal DAG, and return a equivalence class of graphs instead.

Information geometric casual inference

Additive noise models rely on the independences between the cause variable and the exogenous noise variable. Therefore, they are not applicable to discover cause-effect relationships

$$\boldsymbol{y} = f(\boldsymbol{x}),$$

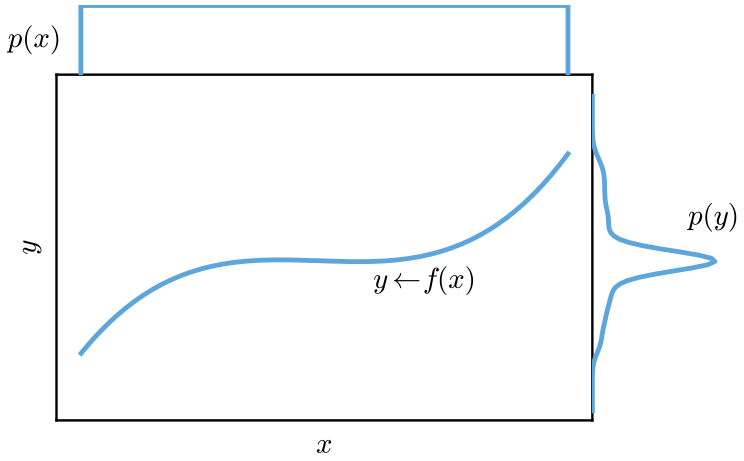


Figure 6.6: Example of information geometric causal inference.

where f is an invertible function, and no noise is present.

Let us exploit the Independence between Cause and Mechanism (ICM) assumption to achieve the identifiability of deterministic causal relations. This is the strategy followed by the Information Geometric Causal Inference (IGCI) method (Daniusis et al., 2010; Janzing et al., 2012), which prefers the causal direction under which the distribution of the cause is independent from the derivative of the mechanism mapping the cause to the effect.

Figure 6.6 illustrates the IGCI method. Here, $\boldsymbol{x} \to \boldsymbol{y}$, $\boldsymbol{x} \equiv \mathcal{U}[a, b]$, and $\boldsymbol{y} \leftarrow f(\boldsymbol{x})$, where f is a smooth, invertible function. The probability density function of the effect variable carries a footprint of the derivative of the function f: regions of large density in $p(\boldsymbol{y})$ correlate with regions of small derivative $f'(\boldsymbol{x})$. Therefore, if we believe in the ICM assumption, these correlations should look suspicious to us, and we should prefer the model $\boldsymbol{x} \to \boldsymbol{y}$, since the density $p(\boldsymbol{x})$ carries no footprint (is independent) from the inverse function $f'^{-1}(\boldsymbol{y})$. On the contrary, if we insist to believe in the incorrect causal relation $\boldsymbol{y} \to \boldsymbol{x}$, we have to also believe that the correlations between $p(\boldsymbol{y})$ and $f'(\boldsymbol{x})$ are spurious.

Time series algorithms

Time series data are collections of samples measured from a system over time, presented as

$$x_{\mathcal{T}} = (x_1, \ldots, x_T)$$

where $\mathcal{T} = \{1, \ldots, T\}$, and $x_t \in \mathbb{R}^d$ is the value of the system at time t, for all $t \in \mathcal{T}$. The major challenge in time series analysis is that samples x_t and $x_{t'}$ measured at nearby times depend on each other. Therefore, we can not assume that time series data is identically and independently distributed

according to some fixed probability distribution, a condition required by all the algorithms reviewed so far in this thesis.

One classic way to measure causal relationships between time series is Granger causation (Granger, 1969). The key idea behind Granger causation is simple. Let

$$x_{\mathcal{T}} = (x_1, \ldots, x_T),$$
$$y_{\mathcal{T}} = (y_1, \ldots, y_T),$$

be two time series forming one isolated system. Then, $x_{\mathcal{T}}$ causes $y_{\mathcal{T}}$ if the prediction of y_{t+1} given $(x_{\mathcal{T}'}, y_{\mathcal{T}'})$ is significantly better than the prediction of y_{t+1} given $(y_{\mathcal{T}'})$ for all $\mathcal{T}' = \{1, \ldots, T-1\}$.

Granger causation was first developed in the context of linear time series, and then extended to model causal effects between nonlinear time series (see the references in Peters (2012)). Granger causation does not account for instantaneous effects between time series, that is, when the value x_t has an effect on the value y_t, and it is prone to failure in the presence of unmeasured, confounding time series. To address some of these issues, Peters (2012, Chapter 8) extends the framework of structural equation models, reviewed in Section 6.3, to the analysis of time series data.

Remark 6.7 *(Causality and time).* In most natural situations, causes precede their effects in time. What is the exact relation between causation, space, and time? Is causal order defined in terms of time order, or vice versa?

These are challenging questions. One can define causal order to follow time order. In turn, time order can be described in terms of the Second Law of Thermodynamics, which states that the entropy of an isolated system increases over time with high probability. The direction of time is then established in two steps. First, we assume a "boundary condition": the universe started in an configuration of extremely low entropy (See Remark 3.4). Second, we define the direction of time as the most common direction of increasing entropy among most isolated systems in the universe. For example, coffee mixing with milk or eggs turning into omelettes are examples of processes of increasing entropy. If we were to play a reversed video of these processes, it would look highly unnatural or "anticausal" to us.

Alternatively, we can adopt a causal theory of time, as put forward by Leibniz, and define time order in terms of causal order. In modern terms, follow Reichenbach's principle of common cause: if a random variable $\boldsymbol{z}$ is a common cause of two other random variables $\boldsymbol{x}$ and $\boldsymbol{y}$, we conclude that $\boldsymbol{z}$ happened before $\boldsymbol{x}$ and $\boldsymbol{y}$. ◊

6.4.3 Limitations of existing algorithms

This section reviewed a variety of observational causal inference algorithms. Each of these algorithms works in a different way, under a different set of

assumptions such as the causal Markov, faithfulness, sufficiency, minimality, acyclicity, linearity, or non-Gaussianity conditions. Unfortunately, these conditions are difficult or impossible to test in practice, and when assumed but violated, causal inferences will be erroneous.

The next chapter presents a different point of view on observational causal inference. There, we pose the problem of deciding the direction of a cause-effect relationship as the problem of classifying probability distributions (Lopez-Paz et al., 2015, 2016b). This interpretation allow us to transfer all the theoretical guarantees and practical advances of machine learning to the problem of observational causal inference, as well as implementing arbitrarily complex prior knowledge about causation as training data.

6.5 Causality and learning

The ICM assumption has remarkable implications in learning (Schölkopf et al., 2012). Consider the common scenario where using data $\{(x_i, y_i)\}_{i=1}^n \sim P(\boldsymbol{x}, \boldsymbol{y})$, we want to learn the function $\mathbb{E}[\boldsymbol{y} \,|\, \boldsymbol{x} = x]$. From a causal point of view, here we face one of two scenarios: either $\boldsymbol{x}$ causes $\boldsymbol{y}$, or $\boldsymbol{y}$ causes $\boldsymbol{x}$. We call the former a *causal learning problem*, since we want to learn a function $\mathbb{E}[\boldsymbol{y} \,|\, \boldsymbol{x} = x]$ mapping one cause to its effect. We call the latter an *anticausal learning problem*, since we want to learn a function $\mathbb{E}[\boldsymbol{x} \,|\, \boldsymbol{y} = y]$ mapping one effect to its cause.

This asymmetry, together with the ICM, entails some distinctions between learning a causal or an anticausal problem. When learning a causal learning problem, further amounts of unlabeled input data $\{(x_i)\}_{i=n+1}^{n+m} \sim P^m(\boldsymbol{x})$ are unhelpful. This is because the ICM assumption tells us that the *cause* distribution $P(\boldsymbol{x})$ contains no information about the function of interest $\mathbb{E}[\boldsymbol{y} \,|\, \boldsymbol{x} = x]$. This negative result holds for regular semisupervised learning, or more complicated variants such as unsupervised, semisupervised, transfer, and domain adaptation learning problems. On the contrary, if we are dealing with an anticausal learning problem, additional unlabeled input data can be of help, since now $P(\boldsymbol{y})$ is the *effect* distribution, which possibly contains information about the function $\mathbb{E}[\boldsymbol{x} \,|\, \boldsymbol{y} = y]$ that we are trying to learn. This distinction is not unique to semisupervised learning, but extend to unsupervised learning, domain adaptation, and multitask learning problems (Schölkopf et al., 2012).

Chapter 7

Learning causal relations

This chapter contains novel material. In particular, we pose the problem of observational cause-effect inference as a binary classification task (Lopez-Paz et al., 2015). To this end, Section 7.2 extends the theory of surrogate risk minimization for binary classification to the problem of learning from samples of probability distributions. Section 7.4 instantiates an algorithm built on top of this theory, termed the Randomized Causation Coefficient (RCC), and shows state-of-the-art causal inference on a variety of simulations on real-world data. Finally, Section 7.6 proposes a variant of RCC based on neural networks, the Neural Causation Coefficient (NCC), and illustrates its use to reveal causal signals in collections of static images, when described by convolutional neural network features (Lopez-Paz et al., 2016c).

A quick look to Figure 7.1 summarizes the central question of this chapter:

given samples from two random variables $\boldsymbol{x}$ *and* $\boldsymbol{y}$*, does* $\boldsymbol{x} \to \boldsymbol{y}$ *or* $\boldsymbol{y} \to \boldsymbol{x}$*?*

The same figure highlights the challenge of answering this question: even for our human eyes, telling between cause and effect from data is a complex task. As opposed to statistical dependence, sharply defined in terms of the difference between joint and marginal distributions, causation lacks a closed mathematical expression, and reveals itself in many forms. This inspires the use of different algorithms in different situations.

In principle, we could tackle the problem of observational causal inference using any of the algorithms reviewed in Section 6.4.2: conditional dependence based algorithms, information geometric methods, additive noise models, and so forth. But which one should we use? In the end, each of these algorithms work under a different and specialized set of assumptions, which are difficult to verify in practice. Each of them exploit a particular *observable causal footprint*, and construct a suitable statistical test to verify its presence in data. But is that particular footprint in our data, or is it another one? What if we want to consider a new footprint? Developing a new causal inference algorithm is adding one new item to the catalog of causal footprints, together with its corresponding statistical test.

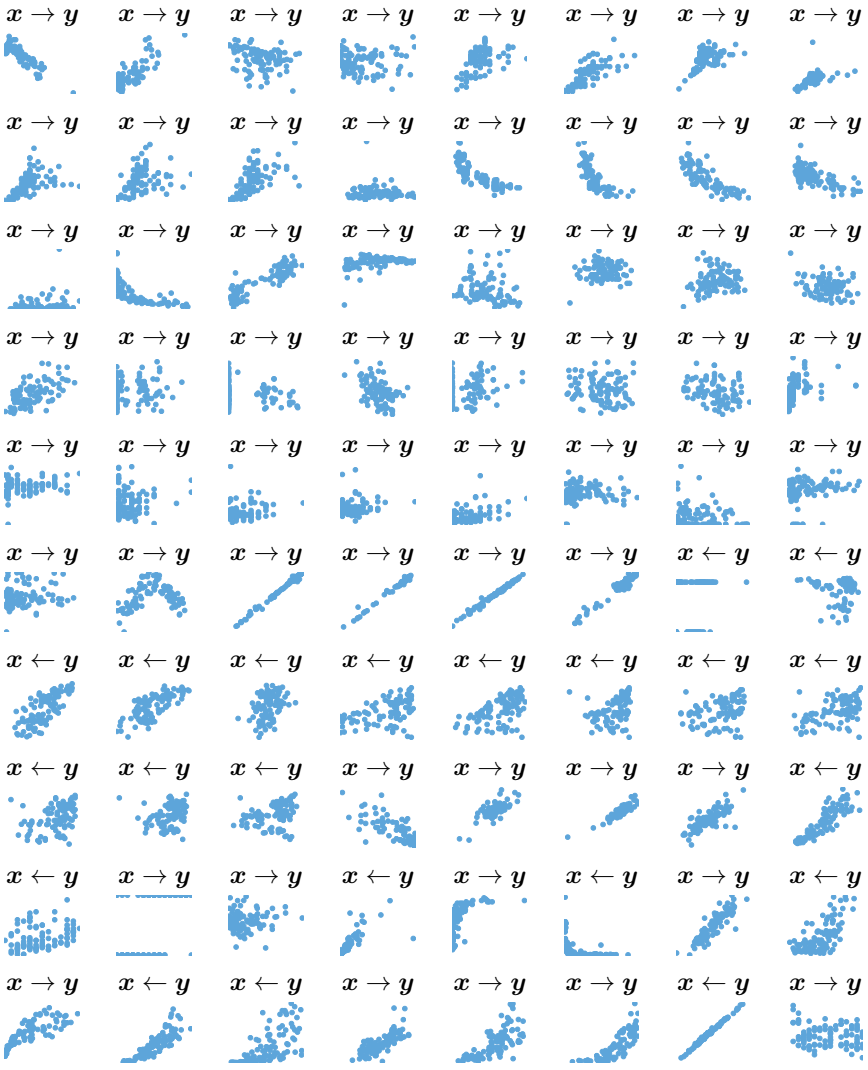


Figure 7.1: Eighty Tübingen pairs of real-world samples with known causal structure. In each plot, the variable $\boldsymbol{x}$ lays on the horizontal axis, and the variable $\boldsymbol{y}$ lays on the vertical axis.

Engineering and maintaining a catalog of causal footprints is a tedious task. Moreover, any such catalog will most likely be incomplete. To amend this issue, this chapter proposes to *learn* such catalog and how to perform causal inference from a corpus of data with labeled causal structure. Such a "data driven" approach moves forward by allowing complex causal assumptions and data generating processes, and removes the need of characterizing new causal footprints and their identifiability conditions.

More specifically, this chapter poses causal inference as the problem of learning to classify probability distributions. To this end, we setup a learning task on the collection of input-output pairs

$$\{(S_i, l_i)\}_{i=1}^n,$$

where each input sample

$$S_i = \{(x_{i,j}, y_{i,j})\}_{j=1}^{n_i} \sim P^{n_i}(\boldsymbol{x}_i, \boldsymbol{y}_i)$$

and each output binary label l_i indicates whether "$\boldsymbol{x}_i \to \boldsymbol{y}_i$" or "$\boldsymbol{x}_i \leftarrow \boldsymbol{y}_i$". Given these data, we build a causal inference rule in two steps. First, we featurize each variable-length input sample S_i into a fixed-dimensional vector representation $\mu_k(S_i)$. Second, we train a binary classifier on the data $\{(\mu_k(S_i), l_i)\}_{i=1}^n$ to distinguish between causal directions.

We organize the exposition as follows. We start by introducing the concept of *kernel mean embeddings* in Section 7.1. These will be the tool of choice to featurize variable-length input samples into fixed-dimensional vector representations. Using kernel mean embeddings, Section 7.2 poses the problem of bivariate causal inference as the task of classifying probability distributions. In that same section, we provide a theoretical analysis on the consistency, learning rates, and large-scale approximations of our setup. In Section 7.2, we extend our ideas from bivariate to multivariate causal inference. Section 7.4 provides a collection of numerical simulations, illustrating that a simple implementation of our framework achieves state-of-the-art causal inference performance in a variety of real world datasets. Finally, Section 7.6 closes this chapter by proposing a variant of RCC based on neural networks, and applying it to the discovery of causal signals in collections of static images.

Example 7.1 *(Prior work on learning from distributions).* The competitions organized by Guyon (2013, 2014) pioneered the view of causal inference as a learning problem. These competitions provided the participants with a large collection of *cause-effect samples* $\{(S_i, l_i)\}_{i=1}^n$, where we sample $S_i = \{(x_{i,j}, y_{i,j})\}_{j=1}^{n_i}$ from the probability distribution $P^{n_i}(\boldsymbol{x}_i, \boldsymbol{y}_i)$, and l_i is a binary label indicating whether "$\boldsymbol{x}_i \to \boldsymbol{y}_i$" or "$\boldsymbol{y}_i \to \boldsymbol{x}_i$". Given these data, most participants adopted the strategy of i) crafting a vector of features from each S_i, and ii) training a binary classifier on top of the constructed features

and paired labels. Although these "data-driven" methods achieved state-of-the-art performance (Guyon, 2013), their hand-crafted features render the theoretical analysis of the algorithms impossible.

In a separate strand of research, there has been multiple proposals to learn from probability distributions (Jebara et al., 2004; Hein and Bousquet, 2005; Cuturi et al., 2005; Martins et al., 2009; Muandet et al., 2012; Póczos et al., 2013). Szabó et al. (2014) presented the first theoretical analysis of distributional learning based on kernel mean embeddings, with a focus on kernel ridge regression. Similarly, Muandet et al. (2012) studied the problem of classifying kernel mean embeddings of distributions, but provided no guarantees regarding consistency or learning rates. ◊

7.1 Kernel mean embeddings

The recurring idea in this chapter is the classification of probability distributions according to their causal structure. Therefore, we first need a way to featurize probability distributions into a vector of features. To this end, we will use *kernel mean embeddings* (Smola et al., 2007; Muandet, 2015). Kernel mean embeddings are tools based on kernel methods: this may be a good time to revisit the introduction about kernels provided in Section 3.1.

In particular, let $P \in \mathcal{P}$ be the probability distribution of some random variable $\boldsymbol{z}$ taking values in the separable topological space $(\mathcal{Z}, \tau_z)$. Then, the *kernel mean embedding* of P associated with the continuous, bounded, and positive-definite kernel function $k : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ is

$$\mu_k(P) := \int_{\mathcal{Z}} k(z, \cdot) \, \mathrm{d}P(z), \tag{7.1}$$

which is an element in $\mathcal{H}_k$, the Reproducing Kernel Hilbert Space (RKHS) associated with k (Schölkopf and Smola, 2001). A key fact is that the mapping $\mu_k : \mathcal{P} \to \mathcal{H}_k$ is injective if k is a *characteristic* kernel (Sriperumbudur et al., 2010). Thus, characteristic kernel mean embeddings satisfy

$$\|\mu_k(P) - \mu_k(Q)\|_{\mathcal{H}_k} = 0 \Leftrightarrow P = Q.$$

The previous implication means that, when using a characteristic kernel, we do not lose any information by embedding distributions. An example of characteristic kernel is the Gaussian kernel, reviewed in Section 3.1.2, and with form

$$k(z, z') = \exp\left(-\gamma \|z - z'\|_2^2\right), \ \gamma > 0. \tag{7.2}$$

We will work with the Gaussian kernel during the remainder of this chapter.

In practice, it is unrealistic to assume access to the distributions P that we wish to embed, and consequently to their exact embeddings $\mu_k(P)$.

Instead, we often have access to a sample $S = \{z_i\}_{i=1}^n \sim P^n$, which we can use to construct the empirical distribution

$$P_S := \frac{1}{n} \sum_{z_i \in S} \delta_{(z_i)},$$

where $\delta_{(z)}$ is the Dirac distribution centered at z. Using the empirical distribution P_S, we can approximate (7.1) by the *empirical kernel mean embedding*

$$\mu_k(P_S) := \frac{1}{n} \sum_{i=1}^{n} k(z_i, \cdot) \in \mathcal{H}_k. \tag{7.3}$$

Figure 7.2 illustrates the transformation of a sample $S = \{z_1, \ldots, z_n\} \sim P^n$ into the empirical kernel mean embedding $\mu_k(P_S) = n^{-1} \sum_{i=1}^n k(\cdot, z_i)$, depicted as a red dot in the Hilbert space $\mathcal{H}_k$.

The following result, slightly improved from (Song, 2008, Theorem 27), characterizes the convergence of the empirical embedding $\mu_k(P_S)$ to the true embedding $\mu_k(P)$ as the sample size n grows.

Theorem 7.1 (Convergence of empirical kernel mean embedding)**.** *Assume that $\|f\|_\infty \leq 1$ for all $f \in \mathcal{H}_k$ with $\|f\|_{\mathcal{H}_k} \leq 1$. Then with probability at least $1 - \delta$ we have*

$$\|\mu_k(P) - \mu_k(P_S)\|_{\mathcal{H}_k} \leq 2\sqrt{\frac{\mathbb{E}_{z \sim P}[k(z,z)]}{n}} + \sqrt{\frac{2 \log \frac{1}{\delta}}{n}}.$$

Proof. See Section 7.7.2. □

At this point, we have the necessary machinery to summarize sets of samples S drawn from distributions P as vectors $\mu_k(P_S)$, which live in the RKHS $\mathcal{H}_k$ associated with some kernel function k. Let's apply these tools to the problem of causal inference.

7.2 Causal inference as distribution classification

This section poses causal inference as the classification of kernel mean embeddings associated to probability distributions with known causal structure, and analyzes the learning rates, consistency, and approximations of such approach. To make things concrete, we encapsulate the setup of our learning problem in the following definition.

Definition 7.1 (Distributional learning setup)**.** *Throughout this chapter, our learning setup is as follows:*

1. *Assume the existence of some* Mother distribution $\mathcal{M}$*, defined on $\mathcal{P} \times \mathcal{L}$, where $\mathcal{P}$ is the set of all Borel probability measures on the space $\mathcal{Z}$ of two causally related random variables, and $\mathcal{L} = \{-1, +1\}$.*

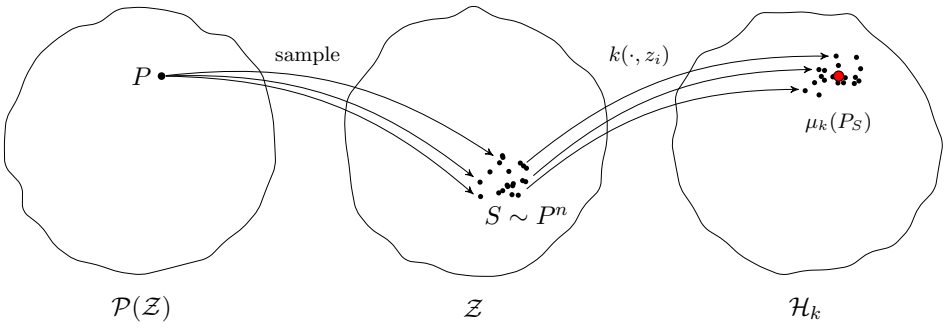


Figure 7.2: Transforming a sample S drawn from a distribution P into the empirical mean embedding $\mu_k(P_S)$.

2. *A set* $\{(P_i, l_i)\}_{i=1}^n$ *is sampled from* $\mathcal{M}^n$*. Each measure* $P_i \in \mathcal{P}$ *is the joint distribution of the causally related random variables* $\boldsymbol{z}_i = (\boldsymbol{x}_i, \boldsymbol{y}_i)$*, and the label* $l_i \in \mathcal{L}$ *indicates whether "*$\boldsymbol{x}_i \to \boldsymbol{y}_i$*" or "*$\boldsymbol{x}_i \leftarrow \boldsymbol{y}_i$*".*

3. *In practice, we do not have access to the measures* $\{P_i\}_{i=1}^n$*. Instead, we observe samples* $S_i = \{(x_{i,j}, y_{i,j})\}_{j=1}^{n_i} \sim P_i^{n_i}$*, for all* $1 \leq i \leq n$*.*

4. *We featurize every sample* S_i *into the empirical kernel mean embedding* $\mu_k(P_{S_i})$ *associated with some kernel function* k *(Equation 7.3). If* k *is a characteristic kernel, we incur no loss of information in this step.*

5. *For computational considerations, we approximate each high-dimensional embedding* $\mu_k(P_{S_i})$ *into the* m*-dimensional embedding* $\mu_{k,m}(P_{S_i})$*. The data* $\{(\mu_{k,m}(P_{S_i}), l_i)\}_{i=1}^n \subseteq \mathbb{R}^m \times \mathcal{L}$ *is provided to the classifier.*

Figure 7.3 summarizes this learning setup.

Using Definition 7.1, we will use the data set $\{(\mu_k(P_{S_i}), l_i)\}_{i=1}^n$ to train a binary classifier from $\mathcal{H}_k$ to $\mathcal{L}$, which we will use to unveil the causal directions of new, unseen probability measures drawn from $\mathcal{M}$. This framework can be straightforwardly extended to also infer the "confounding ($\boldsymbol{x} \leftarrow \boldsymbol{z} \to \boldsymbol{y}$)" and "independent ($\boldsymbol{x} \perp\!\!\!\perp \boldsymbol{y}$)" cases by adding two extra labels to $\mathcal{L}$, as we will exemplify in our numerical simulations.

Given the two nested levels of sampling (being the first one from the Mother distribution $\mathcal{M}$, and the second one from each of the drawn cause-effect measures P_i), it is not trivial to conclude whether this learning procedure is consistent, or how its learning rates depend on the sample sizes n and $\{n_i\}_{i=1}^n$. In the following, we will answer these questions by studying the generalization performance of empirical risk minimization over this learning setup. Specifically, our goal is to upper bound the excess risk between the empirical risk minimizer and the best classifier from our hypothesis class, with respect to the Mother distribution $\mathcal{M}$.

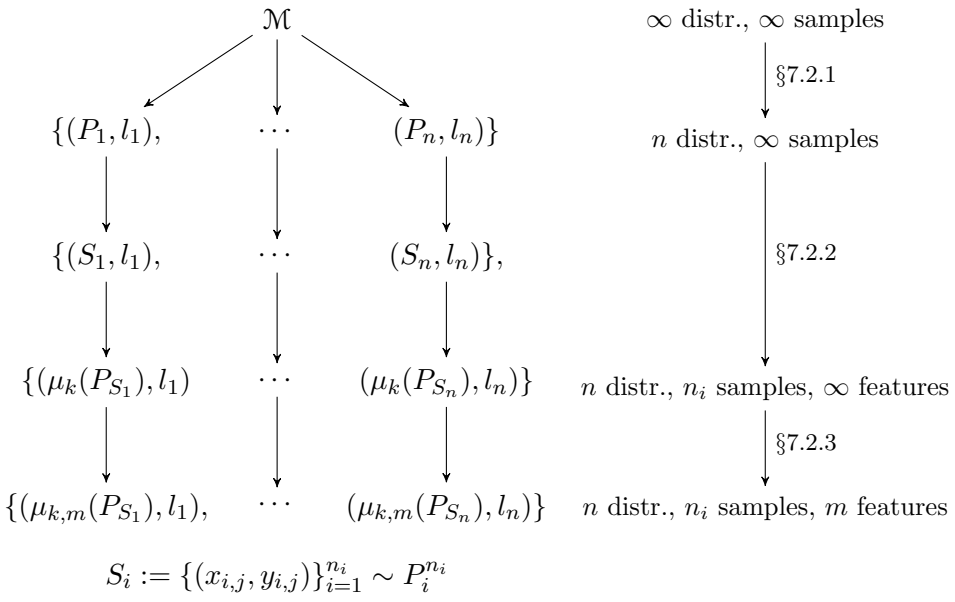


Figure 7.3: Generative process of our learning setup.

We divide our analysis in three parts. Each part will analyze the impact of each of the finite samplings described in Definition 7.1 and depicted in Figure 7.3. First, Section 7.2.1 reviews standard learning theory for surrogate risk minimization. Second, Section 7.2.2 adapts these standard results to the case of empirical kernel mean embedding classification. Third, Section 7.2.3 considers embedding approximations suited to deal with big data, and analyses their impact on learning rates.

Remark 7.1 *(Philosophical considerations).* Reducing causal inference to a learning problem is reducing identifiability assumptions to learnability assumptions. For example, we know from Section 6.4.2 that additive noise models with linear functions and additive Gaussian noise are not identifiable. In the language of learning, this means that the kernel mean embeddings of causal distributions and anticausal distributions fully overlap. Under this framework, a family of distributions $\mathcal{P}$ is causally identifiable if and only if the conditional Mother distributions $\mathcal{M}(\mu_k(\mathcal{P}) \,|\, l = +1)$ and $\mathcal{M}(\mu_k(\mathcal{P}) \,|\, l = -1)$ are separable.

Learning to tell cause from effect on the basis of empirical data relates to other philosophical questions. Paraphrasing Goodman et al. (2011), is the human sense of causation innate or learned? Or invoking David Hume, is the human sense of causation a generalization from the observation of constant association of events? Perhaps the most troubling fact of our framework from a philosophical perspective is that the training data from our learning

setup is *labeled* so the machine, as opposed to learning humans, gets an explicit peek at the true causal structure governing the example distributions. One can mitigate this discrepancy by recalling another: unlike observational causal inference machines, humans obtain causal labels by interacting with the world. ◊

7.2.1 Theory of surrogate risk minimization

Let P be some unknown probability measure defined on $\mathcal{Z} \times \mathcal{L}$, where we call $\mathcal{Z}$ the *input space*, and $\mathcal{L} = \{-1, +1\}$ the *output space*. As introduced in Section 2.3.1, one of the main goals of statistical learning theory is to find a classifier $h\colon \mathcal{Z} \to \mathcal{L}$ that minimizes the *expected risk*

$$R(h) = \mathbb{E}\big[\ell\big(h(\boldsymbol{z}), \boldsymbol{l}\big)\big]$$

for a suitable *loss function* $\ell\colon \mathcal{L} \times \mathcal{L} \to \mathbb{R}^+$, which penalizes departures between predictions $h(z)$ and true labels l. For classification, one common choice of loss function is the *0-1 loss* $\ell_{01}(l, l') = |l - l'|$, for which the expected risk measures the probability of misclassification. Since P is unknown in natural situations, one usually resorts to the minimization of the *empirical risk* $\frac{1}{n}\sum_{i=1}^{n} \ell\big(h(z_i), l_i\big)$ over some fixed hypothesis class $\mathcal{H}$, for *the training set* $\{(z_i, l_i)\}_{i=1}^{n} \sim P^n$. It is well known that this procedure is consistent under mild assumptions (Boucheron et al., 2005).

The exposition is so far parallel to the introduction of learning theory in Section 2.3.1. Unfortunately, the 0-1 loss function is nonconvex, which turns empirical risk minimization intractable. Instead, we will focus on the minimization of surrogate risk functions (Bartlett et al., 2006). We proceed by considering the set of classifiers with form $\mathcal{H} = \{\operatorname{sign} \circ f\colon f \in \mathcal{F}\}$, where $\mathcal{F}$ is some fixed set of real-valued functions $f\colon \mathcal{Z} \to \mathbb{R}$. Introduce a nonnegative *cost function* $\varphi\colon \mathbb{R} \to \mathbb{R}^+$ which is surrogate to the 0-1 loss, that is, $\varphi(\epsilon) \geq \mathbb{I}_{\epsilon > 0}$. For any $f \in \mathcal{F}$, we define its expected and empirical φ-risks as

$$R_\varphi(f) = \mathbb{E}_{(z,l)\sim P}\big[\varphi\big(-f(\boldsymbol{z})\boldsymbol{l}\big)\big] \tag{7.4}$$

and

$$\hat{R}_\varphi(f) = \frac{1}{n}\sum_{i=1}^{n} \varphi\big(-f(z_i)l_i\big). \tag{7.5}$$

Some natural choices of φ lead to tractable empirical risk minimization. Common examples of cost functions include the *hinge loss* $\varphi(\epsilon) = \max(0, 1+\epsilon)$ used in SVM, the *exponential loss* $\varphi(\epsilon) = \exp(\epsilon)$ used in Adaboost, the *logistic loss* $\varphi(\epsilon) = \log_2\big(1 + e^{\epsilon}\big)$ used in logistic regression, and the squared loss $\varphi(\epsilon) = (1 + \epsilon)^2$ used in least-squares regression. Figure 7.4 depicts these losses together with the intractable 0-1 loss.

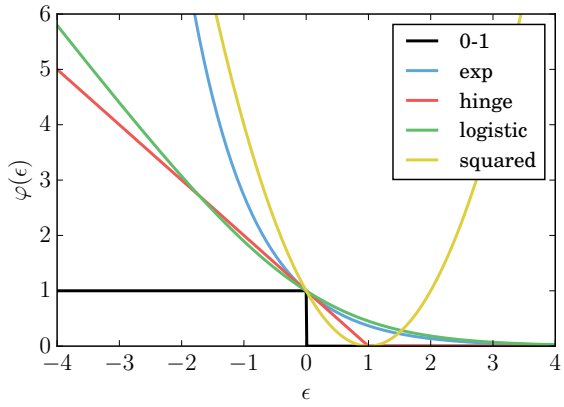


Figure 7.4: Surrogate loss functions for margin-based learning. All tractable surrogate losses upper-bound the intractable 0-1 loss.

The misclassification error of $\text{sign} \circ f$ is always upper bounded by $R_\varphi(f)$. The relationship between functions minimizing $R_\varphi(f)$ and functions minimizing $R(\text{sign} \circ f)$ has been intensively studied in the literature (Steinwart and Christmann, 2008, Chapter 3). Given the high uncertainty associated with causal inferences, we argue that it is more natural to predict class-probabilities instead of hard labels (see Section 2.3.4), a fact that makes the study of margin-based classifiers well suited for our problem.

We now focus on the estimation of $f^* \in \mathcal{F}$, the function minimizing (7.4). But, since the distribution P is unknown, we can only hope to estimate $\hat{f}_n \in \mathcal{F}$, the function minimizing (7.5). Therefore, our goal is to develop high-probability upper bounds on the *excess φ-risk*

$$\mathcal{E}_\mathcal{F}(\hat{f}_n) = R_\varphi(\hat{f}_n) - R_\varphi(f^*), \tag{7.6}$$

with respect to the random training sample $\{(z_i, l_i)\}_{i=1}^n \sim P^n$. As we did back in Equation 2.6, we can upper bound the excess risk (7.6) as:

$$\begin{aligned} \mathcal{E}_\mathcal{F}(\hat{f}_n) &\leq R_\varphi(\hat{f}_n) - \hat{R}_\varphi(\hat{f}_n) + \hat{R}_\varphi(f^*) - R_\varphi(f^*) \\ &\leq 2 \sup_{f \in \mathcal{F}} |R_\varphi(f) - \hat{R}_\varphi(f)|. \end{aligned} \tag{7.7}$$

The following result — in spirit of Koltchinskii and Panchenko (2000); Bartlett and Mendelson (2003) and found in Boucheron et al. (2005, Theorem 4.1) — extends Theorem 2.12 to surrogate risk minimization.

Theorem 7.2 (Excess risk of empirical risk minimization)**.** *Consider a class $\mathcal{F}$ of functions mapping $\mathcal{Z}$ to $\mathbb{R}$. Let $\varphi\colon \mathbb{R} \to \mathbb{R}^+$ be a L_φ-Lipschitz function such that $\varphi(\epsilon) \geq \mathbb{I}_{\epsilon>0}$. Let B be a uniform upper bound on $\varphi\big(-f(\epsilon)l\big)$. Let*

$D = \{(z_i, l_i)\}_{i=1}^n \sim P$ and $\{\sigma_i\}_{i=1}^n$ be iid Rademacher random variables. Then, with probability at least $1-\delta$,

$$\sup_{f\in\mathcal{F}} |R_\varphi(f) - \hat{R}_\varphi(f)| \leq 2L_\varphi \mathrm{Rad}_n(\mathcal{F}) + B\sqrt{\frac{\log(1/\delta)}{2n}},$$

where $\mathrm{Rad}_n(\mathcal{F})$ is the Rademacher complexity of $\mathcal{F}$, see Definition 2.6.

7.2.2 Distributional learning theory

The empirical risk minimization bounds from Theorem 7.2 do not directly apply to our causal learning setup from Definition 7.1. This is because instead of learning a classifier on some sample $\{\mu_k(P_i), l_i\}_{i=1}^n$, we are learning over the set $\{\mu_k(P_{S_i}), l_i\}_{i=1}^n$, where $S_i \sim P_i^{n_i}$. Thus, our input vectors $\mu_k(P_{S_i})$ are "noisy": they exhibit an additional source of variation, just like two any different random samples $S_i, S_i' \sim P_i^{n_i}$ do. This is because the *empirical* mean embedding of two different samples $S_1, S_2 \sim P^n$, depicted as a red dot in Figure 7.2, will differ due to the randomness of the embedded samples S_1 and S_2. In the following, we study how to incorporate these nested sampling effects into an argument similar to Theorem 7.2.

To this end, let us frame the actors playing in Definition 7.1 within the language of standard learning theory laid out in the previous section. Recall that our learning setup initially considers some *Mother distribution* $\mathcal{M}$ over $\mathcal{P} \times \mathcal{L}$. Let $\mu_k(\mathcal{P}) = \{\mu_k(P) : P \in \mathcal{P}\} \subseteq \mathcal{H}_k$, $\mathcal{L} = \{-1, +1\}$, and $\mathcal{M}_k$ be a measure on $\mu_k(\mathcal{P}) \times \mathcal{L}$ induced by $\mathcal{M}$. Although this is an intricate technical condition, we prove the existence of the measure $\mathcal{M}_k$ in Lemma 7.2. Under this measure, we will consider $\mu_k(\mathcal{P}) \subseteq \mathcal{H}_k$ and $\mathcal{L}$ to be the input and output spaces of our learning problem. Let $\big\{\big(\mu_k(P_i), l_i\big)\big\}_{i=1}^n \sim \mathcal{M}_k^n$ be our training set. We will now work with the set of classifiers $\{\mathrm{sign} \circ f \colon f \in \mathcal{F}_k\}$ for some fixed class $\mathcal{F}_k$ of functionals mapping from the RKHS $\mathcal{H}_k$ to $\mathbb{R}$.

As pointed out in the description of our learning setup, we do not have access to the distributions $\{P_i\}_{i=1}^n$, but to samples $S_i \sim P_i^{n_i}$, for all $1 \leq i \leq n$. Because of this reason, we define the *sample-based empirical φ-risk*

$$\tilde{R}_\varphi(f) = \frac{1}{n}\sum_{i=1}^n \varphi\big(-l_i f\big(\mu_k(P_{S_i})\big)\big),$$

which is the approximation to the empirical φ-risk $\hat{R}_\varphi(f)$ that results from substituting the embeddings $\mu_k(P_i)$ with their empirical counterparts $\mu_k(P_{S_i})$.

Our goal is again to find the function $f^* \in \mathcal{F}_k$ minimizing expected φ-risk $R_\varphi(f)$. Since $\mathcal{M}_k$ is unknown to us, and we have no access to the embeddings $\{\mu_k(P_i)\}_{i=1}^n$, we will instead use the minimizer of $\tilde{R}_\varphi(f)$ in $\mathcal{F}_k$:

$$\tilde{f}_n \in \arg\min_{f\in\mathcal{F}_k} \tilde{R}_\varphi(f). \tag{7.8}$$

To sum up, the excess risk (7.6) is equal to

$$R_\varphi(\tilde{f}_n) - R_\varphi(f^*). \tag{7.9}$$

Note that the estimation of f^* drinks from two nested sources of error, which are i) having only n training samples from the distribution $\mathcal{M}_k$, and ii) having only n_i samples from each measure P_i. Using a similar technique to (7.7), we can upper bound (7.9) as

$$R_\varphi(\tilde{f}_n) - R_\varphi(f^*) \leq \sup_{f \in \mathcal{F}_k} |R_\varphi(f) - \hat{R}_\varphi(f)| \tag{7.10}$$

$$+ \sup_{f \in \mathcal{F}_k} |\hat{R}_\varphi(f) - \tilde{R}_\varphi(f)|. \tag{7.11}$$

The term (7.10) is upper bounded by Theorem 7.2. On the other hand, to deal with (7.11), we will need to upper bound the deviations $\left|f\big(\mu_k(P_i)\big) - f\big(\mu_k(P_{S_i})\big)\right|$ in terms of the distances $\|\mu_k(P_i) - \mu_k(P_{S_i})\|_{\mathcal{H}_k}$, which are in turn upper bounded using Theorem 7.1. To this end, we will have to assume that the class $\mathcal{F}_k$ consists of functionals with uniformly bounded Lipschitz constants. One natural example of such a class is the set of linear functionals with uniformly bounded operator norm (Maurer, 2006).

We now present the main result of this section, which provides a high-probability bound on the excess risk (7.9).

Theorem 7.3 (Excess risk of ERM on empirical kernel mean embeddings)**.** *Consider the RKHS $\mathcal{H}_k$ associated with some bounded, continuous, characteristic kernel function k, such that $\sup_{z \in \mathcal{Z}} k(z,z) \leq 1$. Consider a class $\mathcal{F}_k$ of functionals mapping $\mathcal{H}_k$ to $\mathbb{R}$ with Lipschitz constants uniformly bounded by $L_\mathcal{F}$. Let $\varphi\colon \mathbb{R} \to \mathbb{R}^+$ be a L_φ-Lipschitz function such that $\varphi(z) \geq \mathbb{I}_{z>0}$. Let $\varphi\big(-f(h)l\big) \leq B$ for every $f \in \mathcal{F}_k$, $h \in \mathcal{H}_k$, and $l \in \mathcal{L}$. Then, with probability not less than $1-\delta$ (over all sources of randomness)*

$$R_\varphi(\tilde{f}_n) - R_\varphi(f^*) \leq 4L_\varphi R_n(\mathcal{F}_k) + 2B\sqrt{\frac{\log(2/\delta)}{2n}}$$
$$+ \frac{4L_\varphi L_\mathcal{F}}{n} \sum_{i=1}^{n} \left(\sqrt{\frac{\mathbb{E}_{z \sim P_i}[k(z,z)]}{n_i}} + \sqrt{\frac{\log \frac{2n}{\delta}}{2n_i}} \right).$$

Proof. See Section 7.7.3. □

As mentioned in Section 7.2.1, the typical order of $R_n(\mathcal{F}_k)$ is $O(n^{-1/2})$. In such cases, the upper bound in Theorem 7.3 converges to zero (meaning that our procedure is consistent) as both n and n_i tend to infinity, as long as $\log n/n_i = o(1)$. The rate of convergence with respect to n can improve to $O(n^{-1})$ if we place additional assumptions on $\mathcal{M}$ (Bartlett et al., 2005). On the contrary, the rate with respect to n_i is not improvable in general. Namely, the convergence rate $O(n^{-1/2})$ presented in the upper bound of Theorem 7.1 is tight, as shown in the following novel result.

Theorem 7.4 (Lower bound on empirical kernel mean embedding)**.** *Under the assumptions of Theorem 7.1 denote*

$$\sigma^2_{\mathcal{H}_k} = \sup_{\|f\|_{\mathcal{H}_k} \leq 1} \mathbb{V}[f(\boldsymbol{z})] \,.$$

Then there exist universal constants c, C such that for every integer $n \geq 1/\sigma^2_{\mathcal{H}_k}$, and with probability at least c

$$\|\mu_k(P) - \mu_k(P_S)\|_{\mathcal{H}_k} \geq C\frac{\sigma_{\mathcal{H}_k}}{\sqrt{n}}.$$

Proof. See Section 7.7.4. □

For a minimax lower bound on the set of all kernel mean embedding estimators, see (Tolstikhin et al., 2016).

It is instructive to relate the notion of "identifiability" often considered in the causal inference community (Pearl, 2009b) to the properties of the Mother distribution. Saying that the model is *identifiable* means that $\mathcal{M}$ labels each $P \in \mathcal{P}$ deterministically. In this case, learning rates can become as fast as $O(n^{-1})$. On the other hand, as $\mathcal{M}(l|P)$ becomes nondeterministic, the problem degrades to unidentifiable, and learning rates slow down (for example, in the extreme case of cause-effect pairs related by linear functions polluted with additive Gaussian noise, $\mathcal{M}(l = +1|P) = \mathcal{M}(l = -1|P)$ almost surely). The Mother distribution is an useful tool to characterize the difficulty of causal inference problems, as well as a convenient language to place assumptions over the distributions that we want to classify.

7.2.3 Low dimensional embeddings

The embeddings $\mu_k(P_S) \in \mathcal{H}_k$ are nonparametric. As we saw in Remark 3.1, nonparametric representations require solving dual optimization problems, which often involve the construction and inversion of big kernel matrices. Since these are prohibitive operations for large n, in this section we provide m-dimensional approximations to these embeddings based on the random Mercer features, introduced in Section 3.2.2.

We now show that, for any probability measure Q on $\mathcal{Z}$ and $z \in \mathcal{Z}$, we can approximate $k(z, \cdot) \in \mathcal{H}_k \subseteq L^2(Q)$ by a linear combination of randomly chosen elements from the Hilbert space $L^2(Q)$, where $L^2(Q)$ is the set of functions $f : \mathcal{X} \to \mathbb{R}$ satisfying

$$\sqrt{\int_{\mathcal{X}} f^2(x) \mathrm{d}Q(x)} < \infty.$$

Namely, consider the functions parametrised by $w, z \in \mathcal{Z}$ and $b \in [0, 2\pi]$:

$$g^z_{w,b}(\cdot) = 2c_k \cos(\langle w, z\rangle + b) \cos(\langle w, \cdot\rangle + b),$$

which belong to $L^2(Q)$, since they are bounded. If we sample $\{(w_j, b_j)\}_{j=1}^m$ iid, as discussed above, the average

$$\hat{g}_m^z(\cdot) = \frac{1}{m}\sum_{i=1}^m g_{w_i,b_i}^z(\cdot)$$

is an $L^2(Q)$-valued random variable. Moreover, Section 3.2.2 showed that $\mathbb{E}_{\boldsymbol{w},\boldsymbol{b}}[\hat{g}_m^z(\cdot)] = k(z,\cdot)$. This enables us to invoke concentration inequalities for Hilbert spaces (Ledoux and Talagrand, 2013), to show the following result. For simplicity, the following lemma uses Rahimi and Recht (2008, Lemma 1), although a tighter bound could be achieved using recent results from (Sriperumbudur and Szabó, 2015).

Lemma 7.1 (Convergence of random features to $L^2(Q)$ functions)**.** *Let $\mathcal{Z} = \mathbb{R}^d$. For any shift-invariant kernel k, such that $\sup_{z\in\mathcal{Z}} k(z,z) \leq 1$, any fixed $S = \{z_i\}_{i=1}^n \subset \mathcal{Z}$, any probability distribution Q on $\mathcal{Z}$, and any $\delta > 0$, we have*

$$\left\|\mu_k(P_S) - \frac{1}{n}\sum_{i=1}^n \hat{g}_m^{z_i}(\cdot)\right\|_{L^2(Q)} \leq \frac{2c_k}{\sqrt{m}}\left(1 + \sqrt{2\log(n/\delta)}\right)$$

with probability larger than $1-\delta$ over $\{(w_i, b_i)\}_{i=1}^m$.

Proof. See Section 7.7.5. □

Once sampled, the parameters $\{(w_i, b_i)\}_{i=1}^m$ allow us to approximate the empirical kernel mean embeddings $\{\mu_k(P_{S_i})\}_{i=1}^n$ using elements from $\mathrm{span}(\{\cos(\langle w_i, \cdot\rangle + b_i)\}_{i=1}^m)$, which is a finite-dimensional subspace of $L^2(Q)$. Therefore, we propose to use $\{(\mu_{k,m}(P_{S_i}), l_i)\}_{i=1}^n$ as the training sample for our final empirical risk minimization problem, where

$$\mu_{k,m}(P_S) = \frac{2c_k}{|S|}\sum_{z\in S}\big(\cos(\langle w_j, z\rangle + b_j)\big)_{j=1}^m \in \mathbb{R}^m. \tag{7.12}$$

These m-dimensional embeddings require $O(m)$ computation time and $O(1)$ memory storage; Moreover, these finite dimensional embeddings are compatible with most off-the-shelf learning algorithms. For the precise excess risk bounds that take into account the use of these low-dimensional approximations, see Theorem 7.6 in Section 7.7.6.

7.3 Extensions to multivariate causal inference

Although we have focused so far on causal inference between two variables, it is possible to extend our framework to infer causal relationships between $d \geq 2$ variables $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$. To this end, as introduced in Section 6.3.4,

assume the existence of a causal directed acyclic graph G which underlies the dependencies in the probability distribution $P(\boldsymbol{x})$. Therefore, our task is to recover G from $S \sim P^n$.

Naïvely, one could extend the framework presented in Section 7.2 from the binary classification of 2-dimensional distributions to the multiclass classification of d-dimensional distributions. Unfortunately, the number of possible DAGs, which equals the number of labels in the planned multiclass classification problem, grows super-exponentially in d. As an example, attacking causal inference over ten variables using this strategy requires solving a classification problem with 4175098976430598143 different labels.

An alternative approach is to consider the probabilities of the three labels "$\boldsymbol{x}_i \to \boldsymbol{x}_j$", "$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_j$", and "$\boldsymbol{x}_i \perp\!\!\!\perp \boldsymbol{x}_j$" for each pair of variables $\{\boldsymbol{x}_i, \boldsymbol{x}_j\} \subseteq X$, when embedded along with every possible *context* $\boldsymbol{x}_k \subseteq \boldsymbol{x} \setminus \{\boldsymbol{x}_i, \boldsymbol{x}_j\}$. The intuition here is the same as in the PC algorithm described in Section 6.4.2: in order to decide the (absence of a) causal relationship between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, one must analyze the confounding effects of every $\boldsymbol{x}_k \subseteq \boldsymbol{x} \setminus \{\boldsymbol{x}_i, \boldsymbol{x}_j\}$.

7.4 Numerical simulations

We conduct an array of experiments to test the effectiveness of a simple implementation of the causal learning framework described in Definition 7.1, illustrated in Figure 7.3, and analyzed in Section 7.2. Since we will use a set of random features to represent cause-effect samples, we term our method the *Randomized Causation Coefficient* (RCC).

7.4.1 Setting up RCC

In the following three sections we define the protocol to construct our causal direction finder, RCC. To this end, we need to i) construct synthetic distributions on two random variables with labeled causal structures, ii) featurize those distributions into m-dimensional empirical mean embeddings, and iii) train a binary classifier on embeddings and labels.

Synthesis of observational samples

We setup the following generative model to synthesize training data for RCC. Importantly, this generative model is independent from all the experiments that follow. We build $N = 10000$ observational samples $S_i = \{(x_{i,j}, y_{i,j}\}_{j=1}^{n_i}$, with $n_i = n = 1000$ for all $1 \leq i \leq n$. The observational sample S_i has a set

of random hyper-parameters drawn from

$$\begin{aligned} k_i &\sim \text{RandomInteger}[1, 10), \\ m_i &\sim \text{Uniform}[0, 10), \\ s_i &\sim \text{Uniform}[1, 10), \\ v_i &\sim \text{Uniform}[0, 10), \\ d_i &\sim \text{RandomInteger}[4, 10), \\ f_i &\sim \text{RandomSpline}(d_i), \end{aligned}$$

where RandomSpline is a smoothing spline with d_i knots sampled from Gaussian$(0, 1)$. After sampling one set of random hyper-parameters, the pairs $(x_{i,j}, y_{i,j})$ forming the observational sample S_i follow the generative model

$$\begin{aligned} x_{i,j} &\sim \text{GMM}(k_i, m_i, s_i), \\ \epsilon_{i,j} &\sim \text{Gaussian}(0, v_i), \\ y_{i,j} &\leftarrow f_i(x_{i,j}, \epsilon_{i,j}), \end{aligned}$$

where $\text{GMM}(k, p_1, p_2, v)$ is a Gaussian Mixture Model of k_i components with mixing weights sampled from Uniform$[0, 1)$ and normalized to sum to one, component means sampled from Gaussian$(0, m_i^2)$, and variance magnitudes sampled from Gaussian$(0, s_i^2)$. We now have a collection $\{S_i\}_{i=1}^N$ of observational samples $S_i = \{(x_{i,j}, y_{i,j})\}_{j=1}^n$ with known causal relationship $\boldsymbol{x}_i \to \boldsymbol{y}_i$, for all $1 \le i \le N$. To learn from these data, we first have to featurize it into a vector representation compatible with off-the-shelf binary classifiers.

Featurization of observational samples

After constructing each observational sample $S_i = \{(x_{i,j}, y_{i,j})\}_{j=1}^n$, the *featurized* training data for RCC is

$$\begin{aligned} D = \{&(M(\{(x_{i,j}, y_{i,j})\}_{j=1}^n), +1), \\ &(M(\{(y_{i,j}, x_{i,j})\}_{j=1}^n), -1)\}_{i=1}^N, \end{aligned}$$

where we assume that all observational samples have zero mean and unit variance. Here the featurization map M accepts an observational sample $S_i = \{(x_{i,j}, y_{i,j})\}_{i=1}^n$ and takes the form

$$\begin{aligned} M(S_i) = \frac{1}{n} \sum\nolimits_{j=1}^n \big(&(\cos(\langle w_k^x, x_{i,j} \rangle + b_k))_{k=1}^m, \\ &(\cos(\langle w_k^y, y_{i,j} \rangle + b_k))_{k=1}^m, \\ &(\cos(\langle (w_k^x, w_k^y), (x_{i,j}, y_{i,j}) \rangle + b_k))_{k=1}^m \big) \in \mathbb{R}^{3m}, \end{aligned} \tag{7.13}$$

where $w_k^x, w_k^y \sim \mathcal{N}(0, 2\gamma)$, $b_k \sim \mathcal{U}[0, 2\pi]$, and $m = 500$, for all $1 \le k \le m$.

This featurization is a randomized approximation of the empirical kernel mean embedding associated to the Gaussian kernel (7.2). To improve statistical efficiency, M embeds the marginal distribution of $\boldsymbol{x}_i$, the marginal distribution of $\boldsymbol{y}_i$, and the joint distribution of $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ separately. This separate embedding is also to facilitate the inference of causal asymmetries between marginal and conditional distributions. In practice we concatenate the M associated with all bandwidths $\gamma \in \{10^{-2}, \ldots, 10^2\}$, following the multiple kernel learning strategy described in Remark 3.3.

We are almost ready: the synthetic featurized observational data D contains pairs of $3m-$dimensional real vectors $M(S_i)$ and binary labels l_i; therefore, we can now use any standard binary classifier to predict the cause-effect relation for a new observational sample S.

What classifier to use?

To classify the embeddings (7.13) into causal or anticausal, we use the random forest implementation from Python's `sklearn-0.16-git`, with 1000 trees. Random forests are the most competitive alternatives from all the classifiers that we tested, including support vector machines, gradient boosting machines, and neural networks. One possible reason for this is that random forests, as a bagging ensemble, aim at reducing the predictive variance. This is beneficial in our setup, since we know a priori that the test data will come from a different distribution than the Mother distribution. In terms of our theory, the random forest feature map (3.16) induces a valid kernel, over which we perform linear classification.

7.4.2 Classification of Tübingen cause-effect pairs

The *Tübingen cause-effect pairs v0.8* is a collection of heterogeneous, hand-collected, real-world cause-effect samples (Mooij et al., 2014). Figure 7.5 plots the classification accuracy of RCC, IGCI (see Section 6.4.2), and ANM (see Section 6.4.2) versus the fraction of decisions that the algorithms are forced to take of the 82 scalar Tübingen cause-effect pairs. Each algorithm sorts its decisions in decreasing order by confidence. To compare these results to other lower-performance methods, refer to Janzing et al. (2012). Overall, RCC surpasses the state-of-the-art in these data, with a classification accuracy of 82.47% when inferring the causal directions on all pairs. The confidence of RCC are the random forest class probabilities. Computing the RCC statistic for the whole Tübingen dataset takes under three seconds in a single 1.8GhZ processor.

7.4.3 Inferring the arrow of time

We apply RCC to infer the arrow of time from causal time series. More specifically, we assume access to a time series $(x_i)_{i=1}^n$, and our task is to infer

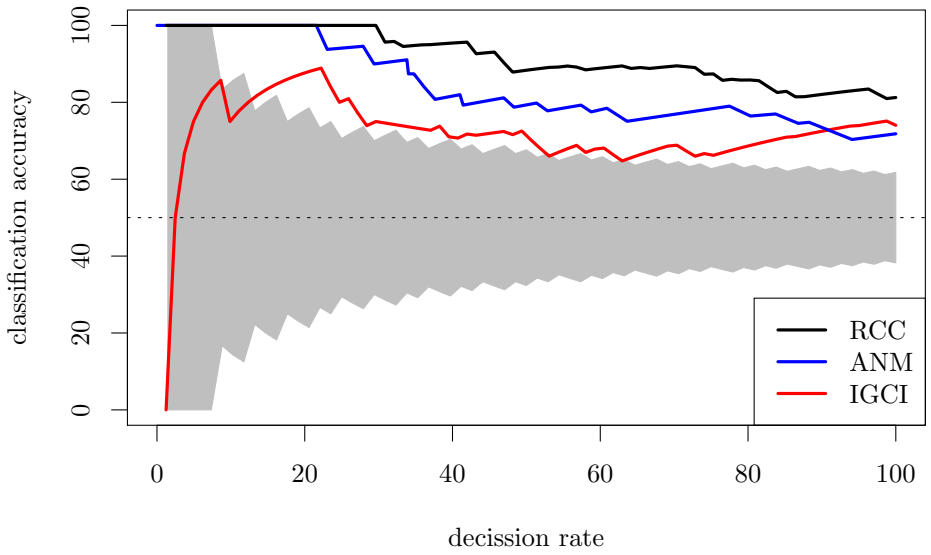


Figure 7.5: Accuracy of RCC, IGCI and ANM on the Tübingen cause-effect pairs, as a function of decision rate. The gray area depicts accuracies not statistically significant.

whether $\boldsymbol{x}_i \to \boldsymbol{x}_{i+1}$ or $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_{i+1}$.

We compare RCC to the state-of-the-art of Peters et al. (2009), using the same electroencephalography signals (Blankertz, 2005) as in their original experiment. On the one hand, Peters et al. (2009) construct two Auto-Regressive Moving-Average (ARMA) models for each causal time series and time direction, and prefers the solution under which the model residuals are independent from the inferred cause. To this end, the method uses two parameters, chosen with heuristics. On the other hand, our approach makes no assumptions whatsoever about the parametric model underlying the series, at the expense of requiring a disjoint set of $N = 10000$ causal time series for training. Our method matches the best performance of Peters et al. (2009), with an accuracy of 82.66%.

7.4.4 ChaLearn's challenge data

The cause-effect challenges organized by Guyon (2014) provided $N = 16199$ training causal samples S_i, each drawn from the distribution of $\boldsymbol{x}_i \times \boldsymbol{y}_i$, and labeled either "$\boldsymbol{x}_i \to \boldsymbol{y}_i$", "$\boldsymbol{x}_i \leftarrow \boldsymbol{y}_i$", "$\boldsymbol{x}_i \leftarrow \boldsymbol{z}_i \to \boldsymbol{y}_i$", or "$\boldsymbol{x}_i \perp\!\!\!\perp \boldsymbol{y}_i$". The goal of the competition was to develop a *causation coefficient* which would predict large positive values to causal samples following "$\boldsymbol{x}_i \to \boldsymbol{y}_i$", large negative values to samples following "$\boldsymbol{x}_i \leftarrow \boldsymbol{y}_i$", and zero otherwise. Using these data, RCC obtained a test *bidirectional area under the curve*

score (Guyon, 2014) of 0.74 in one minute and a half. The winner of the competition obtained a score of 0.82 in thirty minutes, and resorted to dozens of hand-crafted features. Overall, RCC ranked third in the competition.

Partitioning these same data in different ways, we learned two related but different binary classifiers. First, we trained one classifier to *detect latent confounding*, and obtained a test classification accuracy of 80% on the task of distinguishing "$\boldsymbol{x} \to \boldsymbol{y}$ or $\boldsymbol{x} \leftarrow \boldsymbol{y}$" from "$\boldsymbol{x} \leftarrow \boldsymbol{z} \to \boldsymbol{y}$". Second, we trained a second classifier to *measure dependence*, and obtained a test classification accuracy of 88% on the task of distinguishing between "$\boldsymbol{x} \perp\!\!\!\perp \boldsymbol{y}$" and "else". We consider this result to be a promising direction to learn nontrivial statistical tests *from* data.

7.4.5 Reconstruction of causal DAGs

We apply the strategy described in Section 7.3 to reconstruct the causal DAGs of two multivariate datasets: *autoMPG* and *abalone* (Asuncion and Newman, 2007). Once again, we resort to synthetic training data, generated in a similar procedure to the one used in Section 7.4.2. Refer to Section 7.8 for details.

Regarding *autoMPG*, in Figure 7.6, we can see that 1) the release date of the vehicle (AGE) causes the miles per gallon consumption (MPG), acceleration capabilities (ACC) and horse-power (HP), 2) the weight of the vehicle (WEI) causes the horse-power and MPG, and that 3) other characteristics such as the engine displacement (DIS) and number of cylinders (CYL) cause the MPG. For *abalone*, in Figure 7.7, we can see that 1) the age of the snail causes all the other variables, 2) the partial weights of its meat (WEA), viscera (WEB), and shell (WEC) cause the overall weight of the snail (WEI), and 3) the height of the snail (HEI) is responsible for other physically attributes such as its diameter (DIA) and length (LEN).

In Figures 7.6 and 7.7, the target variable for each dataset is shaded in gray. Our inference reveals that the *autoMPG* dataset is a *causal* prediction task (the features *cause* the target), and that the *abalone* dataset is an *anticausal* prediction task (the target *causes* the features). This distinction has implications when learning from these data (Section 6.5).

7.5 Future research directions

Causation and optimal transport The probabilistic account of causation allows for a mathematical characterization of change: given a pair of cause and effect distributions, causation is the operator mapping the cause distribution to the effect distribution. This is a common operation in the research field of *optimal transportation* (Villani, 2003), concerned with studying how to optimally map one distribution into another. However, the

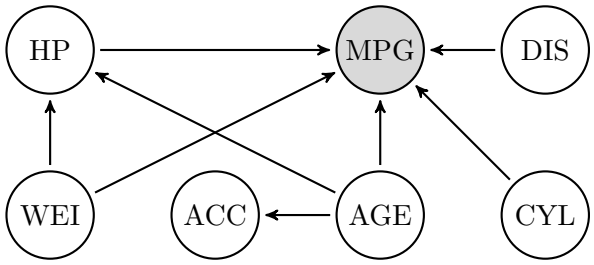


Figure 7.6: Causal DAG recovered from data *autoMPG*.

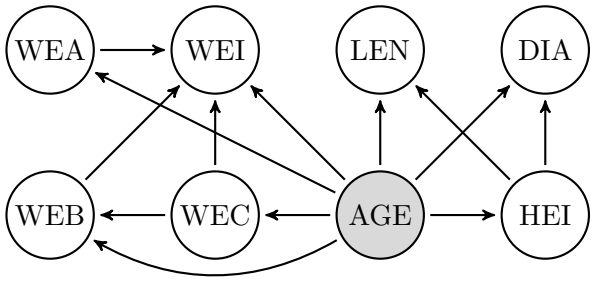


Figure 7.7: Causal DAG recovered from data *abalone*.

author is not aware of any cross-fertilization between the research fields of causal inference and optimal transportation.

Causal regularization Differentiable causal inference methods can act as *causal regularizers*. In particular, one could use RCC or any other causal direction finder to promote learning (anti)causal features in unsupervised learning algorithms, or (anti)causal interventions (with respect to an effect of interest) in reinforcement learning environments.

7.6 Discovering causal signals in images

Imagine an image of a bridge over a river. On top of the bridge, a car speeds through the right lane. Consider the question

"Is there a car in this image?"

This is a question about the observable properties of the scene under consideration, and modern computer vision algorithms excel at answering these kinds of questions. Excelling at this task is fundamentally about leveraging correlations between pixels and image features across large datasets of im-

ages.[1] However, a more nuanced understanding of images arguably requires the ability to *reason about* how the scene depicted in the image would change in response to interventions. The list of possible interventions is long and complex but, as a first step, we can reason about the intervention of removing an object.

To this end, consider the two counterfactual questions *"What would the scene look like if we were to remove the car?"* and *"What would the scene look like if we were to remove the bridge?"* On the one hand, the first intervention seems rather benign. We could argue that the rest of the scene depicted in the image (the river, the bridge) would remain the same if the car were removed. On the other hand, the second intervention seems more severe. If the bridge were removed from the scene, it would make little sense for us to observe the car floating weightless over the river. Thus, we understand that removing the bridge would have an effect on the cars located on top of it. Reasoning about these and similar counterfactuals allows to begin asking questions of the form

"Why is there a car in this image?"

This question is of course poorly defined, but the answer is linked to the causal relationship between the bridge and the car. In our example, the presence of the bridge *causes* the presence of the car, in the sense that if the bridge were not there, then the car would not be either. Such *interventional* semantics of what is meant by *causation* aligns with current approaches in the literature (Pearl, 2009b).

In light of this exposition, it seems plausible that the objects in a scene share asymmetric causal relationships. These causal relationships, in turn, may differ significantly from the correlation structures that modern computer vision algorithms exploit. For instance, most of the images of cars in a given dataset may also contain roads. Therefore, features of cars and features of roads will be highly correlated, and therefore features of roads may be good car predictors in an iid setting irrespective of the underlying causal structure (Schölkopf et al., 2012). However, should a car sinking in the ocean be given a low "car score" by our object recognition algorithm because of its unusual context? The answer depends on the application. If the goal is to maximize the average object recognition score over a test set that has the same distribution as the training set, then we should use the context to make our decision. However, if the goal is to reason about non-iid situations, or cases that may require intervention, such as saving the driver from drowning in the ocean, we should be robust and not refuse to believe that a car is a car just because of its context.

[1] Here and below, the term *correlation* is meant to include the more general concept of *statistical dependence*. The term *feature* denotes, for instance, a numerical value from the image representation of a convolutional neural network.

While the correlation structure of image features may shift dramatically between different data sets or between training data and test data, we expect the causal structure of image features to be more stable. Therefore, object recognition algorithms capable of leveraging knowledge of the cause-effect relations between image features may exhibit better generalization to novel test distributions. For these reasons, the detection of causal signals in images is of great interest. However, this is a very challenging task: in static image datasets we lack the arrow of time, face strong selection biases (pictures are often taken to show particular objects), and randomized experiments (the gold standard to infer causation) are unfeasible. Because of these reasons, our present interest is in detecting causal signals in *observational* data.

In the absence of any assumptions, the determination of causal relations between random variables given samples from their joint distribution is impossible in principle (Pearl, 2009b; Peters et al., 2014). In particular, any joint distribution over two random variables A and B is consistent with any of the following three underlying causal structures: (i) A causes B, (ii) B causes A, and (iii) A and B are both caused by an unobserved confounder C (Reichenbach, 1956). However, while the causal structure may not be identifiable in principle, it may be possible to determine the structure in practice. For joint distributions that occur in the real world, the different causal interpretations may not be equally likely. That is, the causal direction between typical variables of interest may leave a detectable signature in their joint distribution. In this work, we will exploit this insight to build a classifier for determining the cause-effect relation between two random variables from samples of their joint distribution.

Our experiments will show that the higher-order statistics of image datasets can inform us about causal relations. To our knowledge, *no prior work has established, or even considered, the existence of such a signal.*

In particular, we make a first step towards the discovery of causation in visual features by examining large collections of images of different *objects of interest* such as cats, dogs, trains, buses, cars, and people. The locations of these objects in the images are given to us in the form of bounding boxes. For each object of interest, we can distinguish between *object features* and *context features*. By definition, object features are those mostly activated inside the bounding box of the object of interest. On the other hand, context features are those mostly found outside the bounding box of the object of interest. Independently and in parallel, we will distinguish between *causal features* and *anticausal features*, cf. (Schölkopf et al., 2012). Causal features are those that *cause the presence of the object of interest in the image* (that is, those features that cause the object's class label), while anticausal features are those *caused by the presence of the object in the image* (that is, those features caused by the class label). Our hypothesis, to be validated empirically, is

Hypothesis 7.1. *Object features and anticausal features are closely related.*

Context features and causal features are not necessarily related.

We expect Hypothesis 7.1 to be true because many of the features caused by the presence of an object should be features of subparts of the object and hence likely to be contained inside its bounding box (the presence of a car causes the presence of the car's wheels). However, the context of an object may cause or be caused by its presence (road-like features cause the presence of a car, but the presence of a car causes its shadow on a sunny day). Providing empirical evidence supporting Hypothesis 7.1 would imply that (1) there exists a relation between causation and the difference between objects and their contexts, and (2) there exist observable causal signals within sets of static images.

Our exposition is organized as follows. Section 7.6.1 proposes a new algorithm, the Neural Causation Coefficient (NCC), for learning to infer causation from a corpus of labeled data end-to-end using neural networks. Section 7.6.2 makes use of NCC to distinguish between causal and anticausal features. As hypothesized, we show a consistent relationship between anticausal features and object features.

Example 7.2 *(Tanks in bad weather).* The US Army was once interested in detecting the presence of camouflaged tanks in images. To this end, the Army trained a neural network on a dataset of 50 images containing camouflaged tanks, and 50 images not containing camouflaged tanks. Unluckily, all the images containing tanks were taken in cloudy days, and all the images not containing tanks were taken in sunny days. Therefore, the resulting neural network turned out to be a "weather classifier", and its performance to detect tanks in new images was barely above chance (Yudkowsky, 2008). ◊

7.6.1 The neural causation coefficient

To learn causal footprints from data, we follow Section 7.4.1 and pose cause-effect inference as a binary classification task. Our input patterns S_i are effectively scatterplots similar to those shown in Figure 6.5. That is, *each data point is a bag of samples* $(x_{ij}, y_{ij}) \in \mathbb{R}^2$ *drawn iid from a distribution* $P(X_i, Y_i)$. The class label l_i indicates the causal direction between X_i and Y_i.

$$\begin{aligned} D &= \{(S_i, l_i)\}_{i=1}^{n}, \\ S_i &= \{(x_{ij}, y_{ij})\}_{j=1}^{m_i} \sim P^{m_i}(X_i, Y_i), \\ l_i &= \begin{cases} 0 & \text{if } X_i \to Y_i \\ 1 & \text{if } X_i \leftarrow Y_i \end{cases}. \end{aligned} \tag{7.14}$$

Using data of this form, we will train a neural network to classify samples from probability distributions as causal or anticausal. Since the input patterns S_i are not fixed-dimensional vectors, but bags of points, we borrow inspiration

from the literature on kernel mean embedding classifiers (Smola et al., 2007) and construct a feedforward neural network of the form

$$\mathrm{NCC}(\{(x_{ij}, y_{ij})\}_{j=1}^{m_i}) = \psi\left(\frac{1}{m_i}\sum_{j=1}^{m_i}\phi(x_{ij}, y_{ij})\right).$$

In the previous equation, ϕ is a *feature map*, and the average over all $\phi(x_{ij}, y_{ij})$ is the *mean embedding* of the empirical distribution $\frac{1}{m_i}\sum_{i=1}^{m_i}\delta_{(x_{ij},y_{ij})}$. The function ψ is a binary classifier that takes a fixed-length mean embedding as input (Section 7.2.2).

In kernel-based methods such as RCC (Section 7.4.1), ϕ is fixed a priori and defined with respect to a nonlinear kernel (Smola et al., 2007), and ψ is a separate classifier. In contrast, our feature map $\phi : \mathbb{R}^2 \to \mathbb{R}^h$ and our classifier $\psi : \mathbb{R}^h \to \{0, 1\}$ are both multilayer perceptrons, which are learned jointly from data. Figure 7.8 illustrates the proposed architecture, which we term the Neural Causation Coefficient (NCC). In short, to classify a sample S_i as causal or anticausal, NCC maps each point (x_{ij}, y_{ij}) in the sample S_i to the representation $\phi(x_{ij}, y_{ij}) \in \mathbb{R}^h$, computes the embedding vector $\phi_{S_i} := \frac{1}{m_i}\sum_{j=1}^{m_i}\phi(x_{ij}, y_{ij})$ across all points $(x_{ij}, y_{ij}) \in S_i$, and classifies the embedding vector $\phi_{S_i} \in \mathbb{R}^h$ as causal or anticausal using the neural network classifier ψ. Importantly, the proposed neural architecture is not restricted to cause-effect inference, and can be used to represent and learn from general distributions.

NCC has some attractive properties. First, predicting the cause-effect relation for a new set of samples at test time can be done efficiently with a single forward pass through the aggregate network. The complexity of this operation is linear in the number of samples. In contrast, the computational complexity of kernel-based additive noise model inference algorithms is cubic in the number of samples m_i. Second, NCC can be trained using mixtures of different causal and anticausal generative models, such as linear, non-linear, noisy, and deterministic mechanisms linking causes to their effects. This rich training allows NCC to learn a diversity of causal footprints simultaneously. Third, for differentiable activation functions, NCC is a differentiable function. This allows us to embed NCC into larger neural architectures or to use it as a regularization term to encourage the learning of causal or anticausal patterns.

The flexibility of NCC comes at a cost. In practice, labeled cause-effect data as in Equation (7.14) is scarce and laborious to collect. Because of this, we follow Section 7.4.1 and train NCC on artificially generated data.

Synthesis of training data

We will construct n synthetic observational samples, where the ith observational sample contains m_i points. The points comprising the observa-

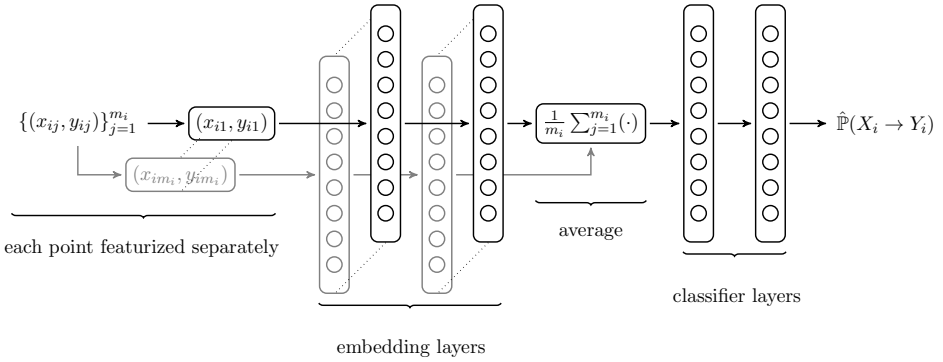


Figure 7.8: Scheme of the Neural Causation Coefficient (NCC) architecture.

tional sample $S_i = \{(x_{ij}, y_{ij})\}_{j=1}^{m_i}$ are drawn from an additive noise model $y_{ij} \leftarrow f_i(x_{ij}) + v_{ij}e_{ij}$, for all $j = 1, \ldots, m_i$.

The *cause terms* x_{ij} are drawn from a mixture of k_i Gaussians distributions. We construct each Gaussian by sampling its mean from Gaussian$(0, r_i)$, its standard deviation from Gaussian$(0, s_i)$ followed by an absolute value, and its unnormalized mixture weight from Gaussian$(0, 1)$ followed by an absolute value. We sample $k_i \sim$ RandomInteger$[1, 5]$ and $r_i, s_i \sim$ Uniform$[0, 5]$. We normalize the mixture weights to sum to one. We normalize $\{x_{ij}\}_{j=1}^{m_i}$ to zero mean and unit variance.

The *mechanism* f_i is a cubic Hermite spline with support

$$[\min(\{x_{ij}\}_{j=1}^{m_i}) - \text{std}(\{x_{ij}\}_{j=1}^{m_i}), \max(\{x_{ij}\}_{j=1}^{m_i}) + \text{std}(\{x_{ij}\}_{j=1}^{m_i})], \quad (7.15)$$

and d_i knots drawn from Gaussian$(0, 1)$, where $d_i \sim$ RandomInteger$(4, 5)$. The noiseless effect terms $\{f(x_{ij})\}_{j=1}^{m_i}$ are normalized to have zero mean and unit variance.

The *noise terms* e_{ij} are sampled from Gaussian$(0, v_i)$, where $v_i \sim$ Uniform$[0, 5]$. To slightly generalize Section 7.4.1, we allow for heteroscedastic noise: we multiply each e_{ij} by v_{ij}, where v_{ij} is the value of a smoothing spline with support defined in Equation (7.15) and d_i random knots drawn from Uniform$[0, 5]$. The noisy effect terms $\{y_{ij}\}_{j=1}^{m_i}$ are normalized to have zero mean and unit variance.

This sampling process produces a training set of $2n$ labeled observational samples

$$D = \{(\{(x_{ij}, y_{ij})\}_{j=1}^{m_i}, 0)\}_{i=1}^{n} \cup \{(\{(y_{ij}, x_{ij})\}_{j=1}^{m_i}, 1)\}_{i=1}^{n}. \quad (7.16)$$

Training NCC

We train NCC with two embedding layers and two classification layers followed by a softmax output layer. Each hidden layer is a composition of

batch normalization (Ioffe and Szegedy, 2015), 100 hidden neurons, a rectified linear unit, and 25% dropout (Srivastava et al., 2014). We train for 10000 iterations using RMSProp (Tieleman and Hinton, 2012) with the default parameters, where each minibatch is of the form given in Equation (7.16) and has size $2n = 32$. Lastly, we further enforce the symmetry $\mathbb{P}(X \to Y) = 1 - \mathbb{P}(Y \to X)$, by training the composite classifier

$$\tfrac{1}{2}\left(1 - \mathrm{NCC}(\{(x_{ij}, y_{ij})\}_{j=1}^{m_i}) + \mathrm{NCC}(\{(y_{ij}, x_{ij})\}_{j=1}^{m_i})\right), \tag{7.17}$$

where $\mathrm{NCC}(\{(x_{ij}, y_{ij})\}_{j=1}^{m_i})$ tends to zero if the classifier believes in $X_i \to Y_i$, and tends to one if the classifier believes in $X_i \leftarrow Y_i$. We chose our parameters by monitoring the validation error of NCC on a held-out set of 10000 synthetic observational samples. Using this held-out validation set, we cross-validated the percentage of dropout over $\{0.1, 0.25, 0.3\}$, the number of hidden layers over $\{2, 3\}$, and the number of hidden units in each of the layers over $\{50, 100, 500\}$.

Testing NCC

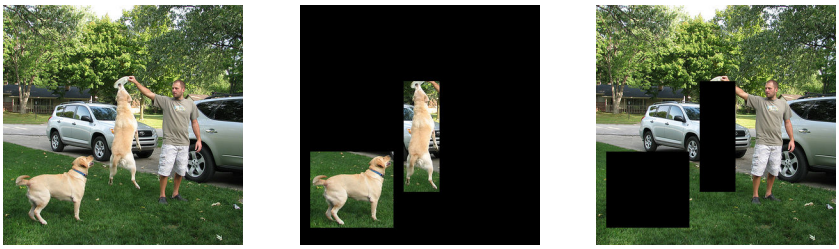
We test the performance of NCC on the Tübingen dataset, version 1.0 (Mooij et al., 2014). This is a collection of one hundred heterogeneous, hand-collected, real-world cause-effect observational samples that are widely used as a benchmark in the causal inference literature (Mooij et al., 2014). The NCC model with the highest synthetic held-out validation accuracy correctly classifies the cause-effect direction of 79% of the Tübingen dataset observational samples. We leave a detailed comparison between RCC and NCC for future work.

7.6.2 Causal signals in sets of static images

We have all the necessary tools to explore the existence of causal signals in sets of static images at our disposal. In the following, we describe the datasets that we use, the process of extracting features from these datasets, and the measurement of *object scores*, *context scores*, *causal scores*, and *anti-causal scores* for the extracted features. Finally, we validate Hypothesis 7.1 empirically.

Datasets

We conduct our experiments with the two datasets PASCAL VOC 2012 Everingham et al. (2012) and Microsoft COCO Lin et al. (2014). These datasets contain heterogeneous images collected "in the wild." Each image may contain multiple objects from different categories. The objects may appear at different scales and angles and may be partially visible or occluded. In the PASCAL dataset, we study all the twenty classes aeroplane, bicycle,



(a) Original image x_j (b) Object image x_j^o (c) Context image x_j^c

Figure 7.9: Blackout processes for object of interest "dog". Original images x_j produce features $\{f_{jl}\}_l$ and class-probabilities $\{c_{jk}\}_k$. Object images x_j^o produce features $\{f_{jl}^o\}_l$. Context images x_j^c produce features $\{f_{jl}^c\}_l$. Blackout processes are performed after image normalization, in order to obtain true zero (black) pixels.

bird, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person, potted plant, sheep, sofa, train, and television. This dataset contains 11541 images. In the COCO dataset, we study the same classes. This selection amounts to 99,309 images. We preprocess the images to have a shortest side of 224 pixels, and then take the central 224×224 crop.

Feature extraction

We use the last hidden representation (before its nonlinearity) of a residual deep convolutional neural network of 18 layers Gross (2016) as a feature extractor. This network was trained on the entire ImageNet dataset Gross (2016). In particular, we denote by $f_j = f(x_j) \in \mathbb{R}^{512}$ the vector of real-valued features obtained from the image $x_j \in \mathbb{R}^{3\times224\times224}$ using this network.

Building on top of these features and using the images from the PASCAL dataset, we train a neural network classifier formed by two hidden layers of 512 units each to distinguish between the 20 classes under study. In particular, we denote by $c_j = c(x_j) \in \mathbb{R}^{20}$ the vector of continuous log odds (activations before the classifier nonlinearity) obtained from the image $x_j \in \mathbb{R}^{3\times224\times224}$ using this classifier. We use features before their nonlinearity and log odds instead of the class probabilities or class labels because NCC has been trained on continuous data with full support on $\mathbb{R}$.

In the following we describe how to compute, for each feature $l = 1, \ldots, 512$, four different scores: its object score, context score, causal score, and anticausal score. Importantly, the object/context scores are computed independently from the causal/anticausal scores. For simplicity, the following sections describe how to compute scores for a particular object of interest k. However, our experiments will repeat this process for all the twenty objects of interest.

Computing "object" and "context" feature scores

We featurize each image x_j in the COCO dataset in three different ways, for all $j = 1 \ldots, m$. First, we featurize the original image x_j as $f_j := f(x_j)$. Second, we blackout the context of the objects of interest k in x_j by placing zero-valued pixels outside their bounding boxes. This produces the object image x_j^o, as illustrated in Figure 7.9b. We featurize x_j^o as $f_j^o = f(x_j^o)$. Third, we blackout the objects of interest k in x_j by placing zero-valued pixels inside their bounding boxes. This produces the context image x_j^c, as illustrated in Figure 7.9c. We featurize x_j^c as $f_j^c = f(x_j^c)$.

Using the previous three featurizations we compute, for each feature $l = 1, \ldots, 512$, its *object score* $s_l^o = \frac{\sum_{j=1}^m |f_{jl}^c - f_{jl}|}{\sum_{j=1}^m |f_{jl}|}$ and its *context score* $s_l^c = \frac{\sum_{j=1}^m |f_{jl}^o - f_{jl}|}{\sum_{j=1}^m |f_{jl}|}$. Intuitively, features with high *object scores* are those features that react violently when the object of interest is removed from the image.

Furthermore, we compute the log odds for the presence of the object of interest k in the original image x_j as $c_{jk} = c(x_j)_k$.

Computing "causal" and "anticausal" feature scores

For each feature l, we compute its *causal score* $1 - \mathrm{NCC}(\{(f_{jl}, c_{jk})\}_{j=1}^m)$, and its *anticausal score* $1 - \mathrm{NCC}(\{(c_{jk}, f_{jl})\}_{j=1}^m)$. Because we will be examining one feature at a time, the values taken by all other features will be an additional source of noise to our analysis, and the observed dependencies will be much weaker than in the synthetic NCC training data. To avoid detecting causation between independent random variables, we train NCC with an augmented training set: in addition to presenting each scatterplot in both causal directions as in (7.16), we pick a random permutation σ to generate an additional uncorrelated example $\{x_{i,\sigma(j)}, y_{ij}\}_{j=1}^{m_i}$ with label $\frac{1}{2}$. We use our best model of this kind which, for validation purposes, obtains 79% accuracy in the Tübingen dataset.

7.6.3 Experiments

Figure 7.10 shows the mean and standard deviation of the object scores and the context scores of the features with the top 1% anticausal scores and the top 1% causal scores. As predicted by Hypothesis 7.1, object features are related to anticausal features. In particular, the features with the highest anticausal score exhibit a higher object score than the features with the highest causal score. This effect is consistent across all 20 classes of interest when selecting the top 1% causal/anticausal features, and remains consistent across 16 out of 20 classes of interest when selecting the top 20% causal/anticausal features. These results indicate that anticausal features may be useful

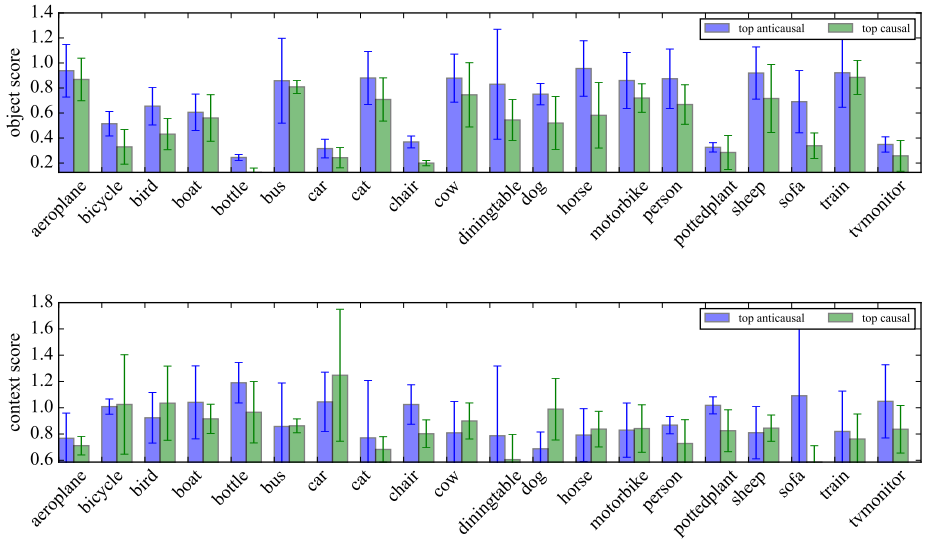


Figure 7.10: Object and context scores for top anticausal and causal features.

for detecting objects in a robust manner, regardless of their context. As stated in Hypothesis 7.1, we could not find a consistent relationship between context features and causal features. Remarkably, we remind the reader that NCC was trained to detect the arrow of causation *independently and from synthetic data*. As a sanity check, we did not obtain any similar results when replacing the NCC with the correlation coefficient or the absolute value of the correlation coefficient.

Although outside the scope of these experiments, we ran some preliminary experiments to find causal relationships between objects of interest, by computing the NCC scores between the log odds of different objects of interest. The strongest causal relationships that we found were "bus causes car," "chair causes plant," "chair causes sofa," "dining table causes bottle," "dining table causes chair," "dining table causes plant," "television causes chair," and "television causes sofa."

Our experiments indicate the existence of statistically observable causal signals within sets of static images. However, further research is needed to best capture and exploit causal signals for applications in image understanding and robust object detection. In particular, we stress the importance of (1) building large, real-world datasets to aid research in causal inference, (2) extending data-driven techniques like NCC to causal inference of more than two variables, and (3) exploring data with explicit causal signals, such as the arrow of time in videos Pickup et al. (2014).

7.7 Proofs

For clarity, we omit bold fonts throughout this section.

7.7.1 Distributional learning is measurable

Let $(\mathcal{Z}, \tau_{\mathcal{Z}})$ and $(\mathcal{L}, \tau_{\mathcal{L}})$ be two separable topological spaces, where we call $\mathcal{Z}$ the *input space* and we call $\mathcal{L} := \{-1, 1\}$ the *output space*. Let $\mathcal{B}(\tau)$ be the Borel σ-algebra induced by the topology τ. Let P be an unknown probability measure on $(\mathcal{Z} \times \mathcal{L}, \mathcal{B}(\tau_{\mathcal{Z}}) \otimes \mathcal{B}(\tau_{\mathcal{L}}))$. Consider also the classifiers $f \in \mathcal{F}_k$ and loss function ℓ to be measurable.

The first step to deploy our learning setup is to guarantee the existence of a measure on the space $\mu_k(\mathcal{P}) \times \mathcal{L}$, where

$$\mu_k(\mathcal{P}) = \{\mu_k(P) : P \in \mathcal{P}\} \subseteq \mathcal{H}_k$$

is the set of kernel mean embeddings of the measures in $\mathcal{P}$. The following lemma provides this guarantee, which allows learning on $\mu_k(\mathcal{P}) \times \mathcal{L}$ throughout this chapter.

Lemma 7.2 (Measurability of distributional learning)**.** *Let $(\mathcal{Z}, \tau_{\mathcal{Z}})$ and $(\mathcal{L}, \tau_{\mathcal{L}})$ be two separable topological spaces. Let $\mathcal{P}$ be the set of all Borel probability measures on $(\mathcal{Z}, \mathcal{B}(\tau_{\mathcal{Z}}))$. Let $\mu_k(\mathcal{P}) = \{\mu_k(P) : P \in \mathcal{P}\} \subseteq \mathcal{H}_k$, where μ_k is the kernel mean embedding* (7.1) *associated to some bounded continuous kernel function $k : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$. Then, there exists a measure on $\mu_k(\mathcal{P}) \times \mathcal{L}$.*

Proof. Start by endowing $\mathcal{P}$ with the weak topology $\tau_{\mathcal{P}}$, such that the map

$$L(P) = \int_{\mathcal{Z}} f(z) \mathrm{d}P(z), \tag{7.18}$$

is continuous for all $f \in C_b(\mathcal{Z})$. This makes $(\mathcal{P}, \mathcal{B}(\tau_{\mathcal{P}}))$ a measurable space.

First, we show that $\mu_k : (\mathcal{P}, \mathcal{B}(\tau_{\mathcal{P}})) \to (\mathcal{H}_k, \mathcal{B}(\tau_{\mathcal{H}}))$ is Borel measurable. Note that $\mathcal{H}_k$ is separable due to the separability of $(\mathcal{Z}, \tau_{\mathcal{Z}})$ and the continuity of k (Steinwart and Christmann, 2008, Lemma 4.33). The separability of $\mathcal{H}_k$ implies μ_k is Borel measurable if and only if it is weakly measurable (Reed and Simon, 1972, Thm. IV.22). Note that the boundedness and the continuity of k imply $\mathcal{H}_k \subseteq C_b(\mathcal{Z})$ (Steinwart and Christmann, 2008, Lemma 4.28). Therefore, (7.18) remains continuous for all $f \in \mathcal{H}_k$, which implies that μ_k is Borel measurable.

Second, $\mu_k : (\mathcal{P}, \mathcal{B}(\tau_{\mathcal{P}})) \to (\mathcal{G}, \mathcal{B}(\tau_{\mathcal{G}}))$ is Borel measurable, since the $\mathcal{B}(\tau_{\mathcal{G}}) = \{A \cap \mathcal{G} : A \in \mathcal{B}(\mathcal{H}_k)\} \subseteq \mathcal{B}(\tau_{\mathcal{H}})$, where $\mathcal{B}(\tau_{\mathcal{G}})$ is the σ-algebra induced by the topology of $\mathcal{G} \in \mathcal{B}(\mathcal{H}_k)$ (Szabó et al., 2014).

Third, we show that $g : (\mathcal{P} \times \mathcal{L}, \mathcal{B}(\tau_{\mathcal{P}}) \otimes \mathcal{B}(\tau_{\mathcal{L}})) \to (\mathcal{G} \times \mathcal{L}, \mathcal{B}(\tau_{\mathcal{G}}) \otimes \mathcal{B}(\tau_{\mathcal{L}}))$ is measurable. For that, it suffices to decompose $g(x, y) = (g_1(x, y), g_2(x, y))$ and show that g_1 and g_2 are measurable, as done by Szabó et al. (2014). □

7.7.2 Theorem 7.1

The statement (Song, 2008, Theorem 27) assumed $f \in [0,1]$, but we let these functions to take negative values. This requires some minor changes of the proof. Using the well known dual relation between the norm in RKHS and sup-norm of empirical process (Song, 2008, Theorem 28), write:

$$\|\mu_k(P) - \mu_k(P_S)\|_{\mathcal{H}_k} = \sup_{\|f\|_{\mathcal{H}_k} \leq 1} \left(\mathbb{E}_{z\sim P}[f(z)] - \frac{1}{n}\sum_{i=1}^{n} f(z_i) \right). \tag{7.19}$$

The sup-norm from the right hand side of the previous equation is real-valued function of the iid random variables $z_1, \ldots, z_n$, which we denote as $F(z_1, \ldots, z_n)$. This function F satisfies the *bounded difference* condition (Theorem 14 of (Song, 2008)). Using this fact, we fix all the values $z_1, \ldots, z_n$ except for z_j, which we replace with z'_j. Using the identity $|a-b| = (a-b)\mathbb{I}_{a>b} + (b-a)\mathbb{I}_{a\leq b}$, and noting that if $\sup_x f(x) = f(x^*)$ then $\sup_x f(x) - \sup_x g(x) \leq f(x^*) - g(x^*)$, write

$$\begin{aligned}
&|F(z_1, \ldots, z'_j, \ldots, z_n) - F(z_1, \ldots, z_j, \ldots, z_n)| \\
&\leq \frac{1}{n}\big(f(z_j) - f(z'_j)\big)\mathbb{I}_{F(z_1,\ldots,z'_j,\ldots,z_n) > F(z_1,\ldots,z_j,\ldots,z_n)} \\
&+ \frac{1}{n}\big(f(z'_j) - f(z_j)\big)\mathbb{I}_{F(z_1,\ldots,z'_j,\ldots,z_n) \leq F(z_1,\ldots,z_j,\ldots,z_n)}.
\end{aligned}$$

Since $|f(z) - f(z')| \in [0,2]$, we conclude that

$$\begin{aligned}
&|F(z_1, \ldots, z'_j, \ldots, z_n) - F(z_1, \ldots, z_j, \ldots, z_n)| \\
&\leq \frac{2}{n}\mathbb{I}_{F(z_1,\ldots,z'_j,\ldots,z_n) > F(z_1,\ldots,z_j,\ldots,z_n)} + \frac{2}{n}\mathbb{I}_{F(z_1,\ldots,z'_j,\ldots,z_n) \leq F(z_1,\ldots,z_j,\ldots,z_n)} = \frac{2}{n}.
\end{aligned}$$

Using McDiarmid's inequality (Theorem 2.8) with $c_i = 2/n$ if follows that, with probability at least $1-\delta$:

$$\begin{aligned}
&\sup_{\|f\|_{\mathcal{H}_k} \leq 1} \left(\mathbb{E}_{z\sim P}[f(z)] - \frac{1}{n}\sum_{i=1}^{n} f(z_i) \right) \\
&\qquad \leq \mathbb{E}\left[\sup_{\|f\|_{\mathcal{H}_k} \leq 1} \left(\mathbb{E}_{z\sim P}[f(z)] - \frac{1}{n}\sum_{i=1}^{n} f(z_i) \right) \right] + \sqrt{\frac{2\log(1/\delta)}{n}}.
\end{aligned}$$

Next, we use symmetrization (Theorem 2.11) to upper bound the expected value of the sup-norm of empirical process with twice the Rademacher complexity of $\{f \in \mathcal{H}_k \colon \|f\|_{\mathcal{H}_k} \leq 1\}$. Finally, we upper bound this Rademacher complexity using (Bartlett and Mendelson, 2003, Lemma 22).

The statement (Song, 2008, Theorem 27) contains extra multiplicative factor 2 under the logarithm, when compared to our result. This is because we upper bound the Rademacher complexity directly, but Song (2008) upper bounds it instead in terms of the empirical Rademacher complexity. This, in turn, requires the use of McDiarmid's inequality together with the union bound.

7.7.3 Theorem 7.3

Start by decomposing the excess risk as:

$$\begin{aligned} R_\varphi(\tilde{f}_n) - R_\varphi(f^*) &= R_\varphi(\tilde{f}_n) - \tilde{R}_\varphi(\tilde{f}_n) \\ &\quad + \tilde{R}_\varphi(\tilde{f}_n) - \tilde{R}_\varphi(f^*) \\ &\quad + \tilde{R}_\varphi(f^*) - R_\varphi(f^*) \\ &\leq 2 \sup_{f \in \mathcal{F}_k} |R_\varphi(f) - \tilde{R}_\varphi(f)| \\ &= 2 \sup_{f \in \mathcal{F}_k} |R_\varphi(f) - \hat{R}_\varphi(f) + \hat{R}_\varphi(f) - \tilde{R}_\varphi(f)| \\ &\leq 2 \sup_{f \in \mathcal{F}_k} |R_\varphi(f) - \hat{R}_\varphi(f)| + 2 \sup_{f \in \mathcal{F}_k} |\hat{R}_\varphi(f) - \tilde{R}_\varphi(f)|, \end{aligned} \tag{7.20}$$

where $\tilde{R}_\varphi(\tilde{f}_n) - \tilde{R}_\varphi(f^*) \leq 0$. We now upper bound the two terms in (7.20).

To upper bound the first term, we must translate the quantities from our distributional learning problem into the quantities from classical learning theory, as discussed in Section 7.2.1. To this end, let $\mu(\mathcal{P})$ play the role of the input space $\mathcal{Z}$. So, the input objects are kernel mean embeddings of elements of $\mathcal{P}$. According to Lemma 7.2, there is a distribution defined over $\mu(\mathcal{P}) \times \mathcal{L}$. This distribution plays the role of the data generating distribution P from classical learning theory. Finally, the iid data $\left\{\left(\mu_k(P_i), l_i\right)\right\}_{i=1}^n$ form the training sample. Thus, using Theorem 7.2 we get that, with probability not less than $1-\delta/2$ with respect to the random training sample $\left\{\left(\mu_k(P_i), l_i\right)\right\}_{i=1}^n$,

$$\sup_{f \in \mathcal{F}_k} |R_\varphi(f) - \hat{R}_\varphi(f)| \leq 2L_\varphi \mathbb{E}\left[\sup_{f \in \mathcal{F}_k} \frac{1}{n} \left|\sum_{i=1}^n \sigma_i f(z_i)\right|\right] + B\sqrt{\frac{\log(2/\delta)}{2n}}. \tag{7.21}$$

To upper bound the second term from (7.20), write

$$\begin{aligned} \sup_{f \in \mathcal{F}_k} |\hat{R}_\varphi(f) - \tilde{R}_\varphi(f)| &= \sup_{f \in \mathcal{F}_k} \left| \frac{1}{n} \sum_{i=1}^n \Big[\varphi\big(-l_i f\big(\mu_k(P_i)\big)\big) - \varphi\big(-l_i f\big(\mu_k(P_{S_i})\big)\big)\Big] \right| \\ &\leq \sup_{f \in \mathcal{F}_k} \frac{1}{n} \sum_{i=1}^n \left|\varphi\big(-l_i f\big(\mu_k(P_i)\big)\big) - \varphi\big(-l_i f\big(\mu_k(P_{S_i})\big)\big)\right| \\ &\leq L_\varphi \sup_{f \in \mathcal{F}_k} \frac{1}{n} \sum_{i=1}^n \left|f\big(\mu_k(P_i)\big) - f\big(\mu_k(P_{S_i})\big)\right|, \end{aligned}$$

where we have used the Lipschitzness of the cost function φ. Using the Lipschitzness of the functionals $f \in \mathcal{F}_k$ we obtain:

$$\sup_{f \in \mathcal{F}_k} |\hat{R}_\varphi(f) - \tilde{R}_\varphi(f)| \leq L_\varphi \sup_{f \in \mathcal{F}_k} \frac{L_f}{n} \sum_{i=1}^n \|\mu_k(P_i) - \mu_k(P_{S_i})\|_{\mathcal{H}_k}. \tag{7.22}$$

We now use 7.1 to upper bound every term in (7.22). We then combine these upper bounds using the union bound over $i = 1, \ldots, n$, and show that for any fixed $P_1, \ldots, P_n$, with probability not less than $1 - \delta/2$ with respect to the random samples $\{S_i\}_{i=1}^n$, it follows that:

$$L_\varphi \sup_{f \in \mathcal{F}} \frac{L_f}{n} \sum_{i=1}^n \|\mu_k(P_i) - \mu_k(P_{S_i})\|_{\mathcal{H}_k} \\ \leq L_\varphi \sup_{f \in \mathcal{F}} \frac{L_f}{n} \sum_{i=1}^n \left(2\sqrt{\frac{\mathbb{E}_{z \sim P}[k(z,z)]}{n_i}} + \sqrt{\frac{2 \log \frac{2n}{\delta}}{n_i}} \right). \quad (7.23)$$

The quantity $2n/\delta$ appears under the logarithm because we have used Theorem 7.1 for every i, with $\delta' = \delta/(2n)$. Combining (7.21) and (7.23) using the union bound into (7.20), we get that with probability not less than $1 - \delta$,

$$R_\varphi(\tilde{f}_n) - R_\varphi(f^*) \leq 4L_\varphi R_n(\mathcal{F}) \\ + 2B\sqrt{\frac{\log(2/\delta)}{2n}} \\ + \frac{4L_\varphi L_\mathcal{F}}{n} \sum_{i=1}^n \left(\sqrt{\frac{\mathbb{E}_{z \sim P}[k(z,z)]}{n_i}} + \sqrt{\frac{\log \frac{2n}{\delta}}{2n_i}} \right),$$

where $L_\mathcal{F} = \sup_{f \in \mathcal{F}} L_f$.

7.7.4 Theorem 7.4

Our proof is a simple combination of the duality equation (7.19) combined with the following lower bound on the suprema of empirical process (Bartlett and Mendelson, 2006, Theorem 2.3):

Theorem 7.5 (Lower bound on supremum of empirical processes)**.** *Let F be a class of real-valued functions defined on a set $\mathcal{Z}$ such that $\sup_{f \in F} \|f\|_\infty \leq 1$. Let $z_1, \ldots, z_n, z \in \mathcal{Z}$ be iid according to some probability measure P on $\mathcal{Z}$. Set $\sigma_F^2 = \sup_{f \in F} \mathbb{V}[f(z)]$. Then there are universal constants c, c', and C for which the following holds:*

$$\mathbb{E}\left[\sup_{f \in F} \left| \mathbb{E}[f(z)] - \frac{1}{n} \sum_{i=1}^n f(z_i) \right| \right] \geq c \frac{\sigma_F}{\sqrt{n}}.$$

Furthermore, for every integer $n \geq 1/\sigma_F^2$, with probability at least c',

$$\sup_{f \in F} \left| \mathbb{E}[f(z)] - \frac{1}{n} \sum_{i=1}^n f(z_i) \right| \geq C\mathbb{E}\left[\sup_{f \in F} \left| \mathbb{E}[f(z)] - \frac{1}{n} \sum_{i=1}^n f(z_i) \right| \right].$$

The constants c, c', and C appearing in the last result do not depend on any other quantities from the statement, such as n, σ_F^2, as seen in the proof provided by Bartlett and Mendelson (2006).

7.7.5 Lemma 7.1

Proof. Recall that Bochner's theorem, presented here as Theorem 3.4, allows to write any real-valued, shift-invariant kernel k on $\mathcal{Z} \times \mathcal{Z}$ as

$$k(z, z') = 2 \int_{\mathcal{Z}} \int_0^{2\pi} \frac{1}{2\pi} p_k(w) \cos(\langle w, z \rangle + b) \cos(\langle w, z' \rangle + b) \, db \, dw,$$

which was first presented as Equation 3.10. As explained in Equation 3.7, this expression mimics the expectation

$$k(z, z') = 2c_k \mathbb{E}_{b,w} \left[\cos(\langle w, z \rangle + b) \cos(\langle w, z' \rangle + b) \right], \tag{7.24}$$

where $w \sim p(w)$, $b \sim \mathcal{U}[0, 2\pi]$, and $c_k = \int_{\mathcal{Z}} p(w) dw < \infty$. Now let Q be any probability distribution defined on $\mathcal{Z}$. Then, for any $z, w \in \mathcal{Z}$ and $b \in [0, 2\pi]$, the function

$$g^z_{w,b}(\cdot) := 2c_k \cos(\langle w, z \rangle + b) \cos(\langle w, \cdot \rangle + b)$$

belongs to $L^2(Q)$. Moreover

$$\begin{aligned} \|g^z_{w,b}(\cdot)\|^2_{L^2(Q)} &= \int_{\mathcal{Z}} \Big(2c_k \cos(\langle w, z \rangle + b) \cos(\langle w, t \rangle + b) \Big)^2 dQ(t) \\ &\leq 4c_k^2 \int_{\mathcal{Z}} dQ(t) = 4c_k^2. \end{aligned}$$

For any fixed $x \in \mathcal{Z}$ and any random parameters $w \in \mathcal{Z}$ and $b \in [0, 2\pi]$, the function $g^z_{w,b}$ is a *random variable* taking values in $L^2(Q)$, which is a Hilbert Space. To study the concentration random variables in Hilbert spaces, we appeal to (Rahimi and Recht, 2008, Lemma 4):

Lemma 7.3 (Hoeffding inequality on Hilbert spaces)**.** *Let $v_1, \ldots, v_m$ be iid random variables taking values in a ball of radius M centered around origin in a Hilbert space H. Then, for any $\delta > 0$, the following holds:*

$$\left\| \frac{1}{m} \sum_{i=1}^m v_i - \mathbb{E} \left[\frac{1}{m} \sum_{i=1}^m v_i \right] \right\|_H \leq \frac{M}{m} \left(1 + \sqrt{2 \log(1/\delta)} \right).$$

with probability higher than $1 - \delta$ over the random sample $v_1, \ldots, v_m$.

Equation (7.24) hints that if w follows the distribution of the normalized Fourier transform $\frac{1}{c_k} p_k$ and $b \sim \mathcal{U}([0, 2\pi])$, then $\mathbb{E}_{w,b} \left[g^z_{w,b}(\cdot) \right] = k(z, \cdot)$. Moreover, we can show that any $h \in \mathcal{H}_k$ is also in $L^2(Q)$:

$$\begin{aligned} \|h(\cdot)\|^2_{L^2(Q)} &= \int_{\mathcal{Z}} \big(h(t) \big)^2 dQ(t) \\ &= \int_{\mathcal{Z}} \langle k(t, \cdot), h(\cdot) \rangle^2_{\mathcal{H}_k} dQ(t) \\ &\leq \int_{\mathcal{Z}} k(t, t) \|h\|^2_{\mathcal{H}_k} dQ(t) \leq \|h\|^2_{\mathcal{H}_k} < \infty, \end{aligned} \tag{7.25}$$

where we have used the reproducing property of k in $\mathcal{H}_k$, the Cauchy-Schwartz inequality, and the boundedness of k. Thus, we conclude that the function $k(z,\cdot) \in L^2(Q)$.

The previous reasoning illustrates that if we have a sample of iid data $\{(w_i,b_i)\}_{i=1}^m$, then $\mathbb{E}\Big[\frac{1}{m}\sum_{i=1}^m g^z_{w_i,b_i}(\cdot)\Big] = k(z,\cdot)$, where $\{g^z_{w_i,b_i}(\cdot)\}_{i=1}^m$ are iid elements of $L^2(Q)$. We conclude by using Lemma 7.3 together with the union bound over each element $z_i \in S$, expressed as:

$$\begin{aligned}\left\|\mu_k(P_S) - \frac{1}{n}\sum_{i=1}^n \hat{g}^{z_i}_m(\cdot)\right\|_{L^2(Q)} &= \left\|\frac{1}{n}\sum_{i=1}^n k(z_i,\cdot) - \frac{1}{n}\sum_{i=1}^n \hat{g}^{z_i}_m(\cdot)\right\|_{L^2(Q)} \\ &\leq \frac{1}{n}\sum_{i=1}^n \|k(z_i,\cdot) - \hat{g}^{z_i}_m(\cdot)\|_{L^2(Q)} \\ &= \frac{1}{n}\sum_{i=1}^n \left\|k(z_i,\cdot) - \frac{1}{m}\sum_{i=j}^m g^{z_i}_{w_j,b_j}(\cdot)\right\|_{L^2(Q)},\end{aligned}$$

where we have used the triangle inequality. □

7.7.6 Excess risk for low dimensional representations

For any $w, z \in \mathcal{Z}$ and $b \in [0, 2\pi]$, define the function

$$g^z_{w,b}(\cdot) = 2c_k \cos(\langle w, z\rangle + b)\cos(\langle w,\cdot\rangle + b) \in L^2(Q), \tag{7.26}$$

where $c_k = \int_{\mathcal{Z}} p_k(z)dz$ for $p_k \colon \mathcal{Z} \to \mathbb{R}$ is the Fourier transform of k. Sample m pairs $\{(w_i,b_i)\}_{i=1}^m$ from $\left(\frac{1}{c_k}p_k\right) \times \mathcal{U}[0,2\pi]$, and define the average

$$\hat{g}^z_m(\cdot) = \frac{1}{m}\sum_{i=1}^m g^z_{w_i,b_i}(\cdot) \in L^2(Q).$$

Given a kernel function k, the sinusoids (7.26) do not necessarily belong its RKHS $\mathcal{H}_k$. Since we are going to use such sinusoids as training data, our classifiers should act on the more general space $L^2(Q)$. To this end, we redefine the set of classifiers introduced in the Section 7.2.2 to be $\{\text{sign}\circ f \colon f \in \mathcal{F}_Q\}$, where $\mathcal{F}_Q$ is the set of functionals mapping $L^2(Q)$ to $\mathbb{R}$.

Our goal is to find a function f^* such that

$$f^* \in \arg\min_{f\in\mathcal{F}_Q} R_\varphi(f) := \arg\min_{f\in\mathcal{F}_Q} \mathbb{E}_{(P,l)\sim\mathcal{M}}\Big[\varphi\Big(-f\big(\mu_k(P)\big)l\Big)\Big]. \tag{7.27}$$

As described in Section 7.7.5, the kernel boundedness condition $\sup_{z\in\mathcal{Z}} k(z,z) \leq 1$ implies $\mathcal{H}_k \subseteq L^2(Q)$. In particular, for any $P \in \mathcal{P}$ it holds that $\mu_k(P) \in L^2(Q)$, and thus (7.27) is well defined.

We will approximate (7.27) by empirical risk minimization. This time we will replace the infinite-dimensional empirical mean embeddings $\{\mu_k(P_{S_i})\}_{i=1}^n$ with low-dimensional representations formed by random sinusoids (7.26). Namely, we propose to use the following estimator $\tilde{f}_n^m$:

$$\tilde{f}_n^m \in \arg\min_{f \in \mathcal{F}_Q} \tilde{R}_\varphi^m(f) := \arg\min_{f \in \mathcal{F}_Q} \frac{1}{n} \sum_{i=1}^n \varphi\left(-f\left(\frac{1}{n_i} \sum_{z \in S_i} \hat{g}_m^z(\cdot)\right) l_i\right).$$

The following result combines Theorem 7.3 and Lemma 7.1 to provide an excess risk bound for $\tilde{f}_n^m$, which accounts for all sources of the errors introduced in the learning pipeline: n training distributions, n_i samples from the ith training distribution, and m random features to represent empirical mean embeddings.

Theorem 7.6 (Excess risk of ERM on empirical kernel mean embeddings and random features)**.** *Let $\mathcal{Z} = \mathbb{R}^d$ and Q be any probability distribution on $\mathcal{Z}$. Consider the RKHS $\mathcal{H}_k$ associated with some bounded, continuous, characteristic and shift-invariant kernel function k, such that $\sup_{z \in \mathcal{Z}} k(z,z) \leq 1$. Consider a class $\mathcal{F}_Q$ of functionals mapping $L^2(Q)$ to $\mathbb{R}$ with Lipschitz constants uniformly bounded by L_Q. Let $\varphi\colon \mathbb{R} \to \mathbb{R}^+$ be a L_φ-Lipschitz function such that $\phi(z) \geq \mathbb{I}_{z>0}$. Let $\varphi\big(-f(h)l\big) \leq B$ for every $f \in \mathcal{F}_Q$, $h \in L^2(Q)$, and $l \in \mathcal{L}$. Then for any $\delta > 0$ the following holds:*

$$\begin{aligned} R_\varphi(\tilde{f}_n^m) - R_\varphi(f^*) \leq{}& 4L_\varphi R_n(\mathcal{F}_Q) + 2B\sqrt{\frac{\log(3/\delta)}{2n}} \\ &+ \frac{4L_\varphi L_Q}{n} \sum_{i=1}^n \left(\sqrt{\frac{\mathbb{E}_{z \sim P_i}[k(z,z)]}{n_i}} + \sqrt{\frac{\log \frac{3n}{\delta}}{2n_i}}\right) \\ &+ 2\frac{L_\varphi L_Q}{n} \sum_{i=1}^n \frac{2c_k}{\sqrt{m}} \left(1 + \sqrt{2\log(3n \cdot n_i/\delta)}\right) \end{aligned}$$

with probability not less than $1-\delta$ over all sources of randomness, which are $\{(P_i, l_i)\}_{i=1}^n$, $\{S_i\}_{i=1}^n$, $\{(w_i, b_i)\}_{i=1}^m$.

Proof. We will proceed similarly to (7.20). Decompose the excess risk as:

$$\begin{aligned}
R_\varphi(\tilde{f}_n^m) - R_\varphi(f^*) &= R_\varphi(\tilde{f}_n^m) - \tilde{R}_\varphi^m(\tilde{f}_n^m) \\
&\quad + \tilde{R}_\varphi^m(\tilde{f}_n^m) - \tilde{R}_\varphi^m(f^*) \\
&\quad + \tilde{R}_\varphi^m(f^*) - R_\varphi(f^*) \\
&\leq 2 \sup_{f \in \mathcal{F}_Q} |R_\varphi(f) - \tilde{R}_\varphi^m(f)| \\
&= 2 \sup_{f \in \mathcal{F}_Q} |R_\varphi(f) - \hat{R}_\varphi(f) + \hat{R}_\varphi(f) - \tilde{R}_\varphi(f) + \tilde{R}_\varphi(f) - \tilde{R}_\varphi^m(f)| \\
&\leq 2 \sup_{f \in \mathcal{F}_Q} |R_\varphi(f) - \hat{R}_\varphi(f)| \qquad (7.28) \\
&\quad + 2 \sup_{f \in \mathcal{F}_Q} |\hat{R}_\varphi(f) - \tilde{R}_\varphi(f)| \\
&\quad + 2 \sup_{f \in \mathcal{F}_Q} |\tilde{R}_\varphi(f) - \tilde{R}_\varphi^m(f)|.
\end{aligned}$$

The first two terms of (7.28) were upper bounded in Section 7.7.3. The upper bound of the second term (proved in Theorem 7.3) relied on the assumption that functionals in F_Q are Lipschitz on $\mathcal{H}_k$, with respect to the RKHS norm. When using bounded kernels, we have $\mathcal{H}_k \subseteq L^2(Q)$, which implies $\|h\|_{L^2(Q)} \leq \|h\|_{\mathcal{H}_k}$ for any $h \in \mathcal{H}_k$ (see (7.25)). Thus,

$$|f(h) - f(h')| \leq L_f \|h - h'\|_{L^2(Q)} \leq L_f \|h - h'\|_{\mathcal{H}_k}$$

for any $h, h' \in \mathcal{H}_k$. This means that the assumptions of Theorem 7.3 hold, and we can safely apply it to upper bound the first two terms of (7.28).

The last step is to upper bound the third term in (7.28). To this end,

$$\begin{aligned}
&\sup_{f \in \mathcal{F}_Q} |\tilde{R}_\varphi(f) - \tilde{R}_\varphi^m(f)| \\
&= \sup_{f \in \mathcal{F}_Q} \left| \frac{1}{n} \sum_{i=1}^n \varphi\left(-f\big(\mu_k(P_{S_i})\big) l_i\right) - \frac{1}{n} \sum_{i=1}^n \varphi\left(-f\left(\frac{1}{n_i} \sum_{z \in S_i} \hat{g}_m^z(\cdot)\right) l_i\right) \right| \\
&\leq \frac{1}{n} \sum_{i=1}^n \sup_{f \in \mathcal{F}_Q} \left| \varphi\left(-f\big(\mu_k(P_{S_i})\big) l_i\right) - \varphi\left(-f\left(\frac{1}{n_i} \sum_{z \in S_i} \hat{g}_m^z(\cdot)\right) l_i\right) \right| \\
&\leq \frac{L_\varphi}{n} \sum_{i=1}^n \sup_{f \in \mathcal{F}_Q} \left| f\big(\mu_k(P_{S_i})\big) - f\left(\frac{1}{n_i} \sum_{z \in S_i} \hat{g}_m^z(\cdot)\right) \right| \\
&\leq \frac{L_\varphi}{n} \sum_{i=1}^n \sup_{f \in \mathcal{F}_Q} L_f \left\| \mu_k(P_{S_i}) - \frac{1}{n_i} \sum_{z \in S_i} \hat{g}_m^z(\cdot) \right\|_{L^2(Q)}.
\end{aligned}$$

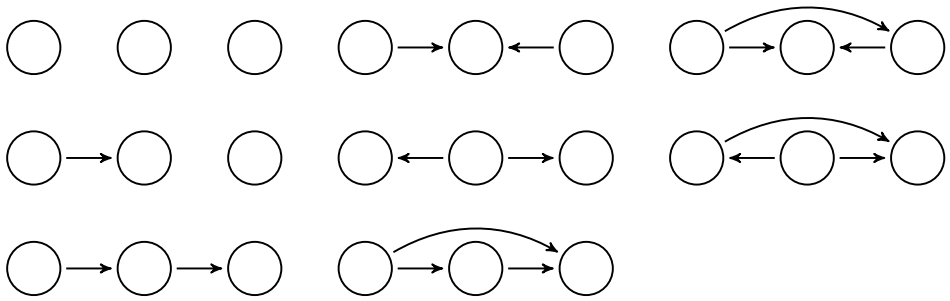


Figure 7.11: The eight possible directed acyclic graphs on three variables.

We can now use Lemma 7.1 and the union bound over $i = 1, \ldots, n$ with $\delta' = \delta/n$. This yields

$$\sup_{f \in \mathcal{F}_Q} |\tilde{R}_\varphi(f) - \tilde{R}^m_\varphi(f)| \leq \frac{L_\varphi L_Q}{n} \sum_{i=1}^{n} \frac{2c_k}{\sqrt{m}} \left(1 + \sqrt{2\log(n \cdot n_i/\delta)}\right).$$

with probability not less than $1 - \delta$ over $\{(w_i, b_i)\}_{i=1}^m$. □

7.8 Training and test protocols for Section 7.4.5

The synthesis of training data for the experiments in Section 7.4.5 resembles the one in Section 7.4.2. The main difference here is that, when trying to infer the cause-effect relationship between two variables $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ embedded in a larger set of variables $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$, we have to take into account the potential confounding effects of the variables $\boldsymbol{x}_k \subseteq \boldsymbol{x} \setminus \{\boldsymbol{x}_i, \boldsymbol{x}_j\}$. For the sake of simplicity, we will only consider one-dimensional confounding effects, that is, scalar $\boldsymbol{x}_k$.

7.8.1 Training phase

To generate cause-effect pairs that exemplify every possible type of scalar confounding, we generate data from the eight possible directed acyclic graphs on three variables, depicted in Figure 7.11.

In particular, we will sample N different causal DAGs $G_1, \ldots, G_N$, where the G_i describes the causal structure underlying $(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{z}_i)$. Given G_i, we generate the sample set $S_i = \{(x_{i,j}, y_{i,j}, z_{i,j})\}_{j=1}^n$ according to the generative process described in Section 7.4.2. Together with S_i, we annotate the triplet of labels $(l_{i,1}, l_{i,2}, l_{i,3})$, where according to G_i,

- $l_{i,1} = +1$ if "$\boldsymbol{x}_i \to \boldsymbol{y}_i$", $l_{i,1} = -1$ if "$\boldsymbol{x}_i \leftarrow \boldsymbol{y}_i$", and $l_{i,1} = 0$ else.
- $l_{i,2} = +1$ if "$\boldsymbol{y}_i \to \boldsymbol{z}_i$", $l_{i,2} = -1$ if "$\boldsymbol{y}_i \leftarrow \boldsymbol{z}_i$", and $l_{i,2} = 0$ else.

- $l_{i,3} = +1$ if "$\boldsymbol{x}_i \to \boldsymbol{z}_i$", $l_{i,1} = -1$ if "$\boldsymbol{x}_i \leftarrow \boldsymbol{z}_i$", and $l_{i,1} = 0$ else.

Then, we add the following six elements to our training set:

$$
\begin{aligned}
&(\{(x_{i,j}, y_{i,j}, z_{i,j})\}_{j=1}^{n}, +l_{i,1}),\\
&(\{(y_{i,j}, z_{i,j}, x_{i,j})\}_{j=1}^{n}, +l_{i,2}),\\
&(\{(x_{i,j}, z_{i,j}, y_{i,j})\}_{j=1}^{n}, +l_{i,3}),\\
&(\{(y_{i,j}, x_{i,j}, z_{i,j})\}_{j=1}^{n}, -l_{i,1}),\\
&(\{(z_{i,j}, y_{i,j}, x_{i,j})\}_{j=1}^{n}, -l_{i,2}),\\
&(\{(z_{i,j}, x_{i,j}, y_{i,j})\}_{j=1}^{n}, -l_{i,3}),
\end{aligned}
$$

for all $1 \leq i \leq N$. Therefore, our training set will consist on $6N$ sample sets and their paired labels. At this point, and given any sample $\{(u_{i,j}, v_{i,j}, w_{i,j})\}_{j=1}^{n}$ from the training set, we propose to use as feature vectors the concatenation of the $m-$dimensional empirical kernel mean embeddings (7.12) of $\{u_{i,j}\}_{j=1}^{n}$, $\{v_{i,j}\}_{j=1}^{n}$, and $\{(u_{i,j}, v_{i,j}, w_{i,j})\}_{j=1}^{n}$.

7.8.2 Test phase

In order to estimate the causal graph underlying the test sample set S, we compute three $d \times d$ matrices $M_{\to}$, $M_{\perp\!\!\!\perp}$, and $M_{\leftarrow}$. Each of these three matrices will contain, at their coordinates i, j, the class probabilities of the labels "$\boldsymbol{x}_i \to \boldsymbol{x}_j$", "$\boldsymbol{x}_i \perp\!\!\!\perp \boldsymbol{x}_j$", and "$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_j$", when voting over all possible scalar confounders $\boldsymbol{x}_k$. Using these matrices, we estimate the underlying causal graph by selecting the type of each edge (forward, backward, or no edge) to be the one with maximal probability from the three matrices, and according to our classifier. As a post-processing step, we prune the least-confident edges until the derived graph is a DAG.

Note that our binary classifier is taught to predict the existence of an arrow in a large graph by observing only a small subset (three nodes) of such graph. Therefore, our binary classifier is taught to ignore arrows due to confounding, and to predict only arrows due to direct causal relationships.

Chapter 8

Conclusion and future directions

This chapter contains novel material. In particular, we introduce three directions for future research in artificial intelligence: machines-teaching-machines paradigms (Section 8.1, Lopez-Paz et al. (2016a)), the supervision continuum (Section 8.3), and probabilistic convexity (Section 8.2).

Learning machines excel at prediction, one integral part of intelligence. But intelligent behaviour must complete prediction with reasoning, and reasoning requires mastering causal inference. To summarize this thesis bluntly,

dependence and causation are learnable from observational data.

Such conclusion further motivates solving the dilemma introduced in this thesis, namely:

causal inference is key to intelligence, yet ignored by learning algorithms.

Prediction studies single probability distributions. In opposition, causation bridges different but related probability distributions, let them be the training and testing distributions of a learning problem; the multiple distributions involved in multitask, domain adaptation, and transfer learning; the changing distributions governing a reinforcement or online learning scenario; or the different environments over which we plan our actions and anticipate their outcomes. The differences between these different but related distributions are often *causal leaps of faith*, used to answer what could had been, but it never was. The ability to use these causal leaps to our advantage is what makes us reasoning, creative, intelligent, human agents. These causal links are the same connections that we use to tie different learning problems together, transform one piece of knowledge into another, and more generally, make learning a holistic experience rather than multiple independent tasks. Thus, the development of methods able to discover causal structures from

data, and the use of these structures in machine learning is one necessary step towards machine reasoning and artificial intelligence.

The last chapter of this thesis is a reflection on what I consider three novel and important frontiers in artificial intelligence: machine-teaching-machines paradigms, theory of nonconvex optimization, and the supervision continuum. The following exposition relies on unpublished work, not necessarily related to causation, and the reader should understand this chapter as a collection of conjectures that are currently under investigation.

8.1 Machines-teaching-machines paradigms

Humans learn much faster than machines. Vapnik and Izmailov (2015) illustrate this discrepancy with the Japanese proverb

better than a thousand days of diligent study is one day with a great teacher.

Motivated by this insight, the authors incorporate an "intelligent teacher" into machine learning. Their solution is to consider training data formed by a collection of triplets

$$\{(x_1, x_1^\star, y_1), \dots, (x_n, x_n^\star, y_n)\} \sim P^n(x, x^\star, y).$$

Here, each (x_i, y_i) is a feature-label pair, and the novel element $x_i^\star$ is additional information about the example (x_i, y_i) provided by an intelligent teacher, such as to support the learning process. Unfortunately, the learning machine will not have access to the teacher explanations $x_i^\star$ at test time. Thus, the framework of *learning using privileged information* (Vapnik and Vashist, 2009; Vapnik and Izmailov, 2015) studies how to leverage these explanations $x_i^\star$ at training time, to build a classifier for test time that outperforms those built on the regular features x_i alone. As an example, x_i could be the image of a biopsy, $x_i^\star$ the medical report of an oncologist when inspecting the image, and y_i a binary label indicating whether the tissue shown in the image is cancerous or healthy.

The previous exposition finds a mathematical justification in VC theory (Vapnik, 1998), which characterizes the speed at which machines learn using two ingredients: the capacity or flexibility of the machine, and the amount of data that we use to train it. Consider a binary classifier f belonging to a function class $\mathcal{F}$ with finite VC-Dimension $|\mathcal{F}|_{\text{VC}}$. Then, with probability $1 - \delta$, the *expected error* $R(f)$ is upper bounded by

$$R(f) \leq R_n(f) + O\left(\left(\frac{|\mathcal{F}|_{\text{VC}} - \log \delta}{n}\right)^\alpha\right),$$

where $R_n(f)$ is the training error over n data, and $\frac{1}{2} \leq \alpha \leq 1$. For difficult (*not separable*) problems the exponent is $\alpha = \frac{1}{2}$, which translates into

machines learning at a *slow* rate of $O(n^{-1/2})$. On the other hand, for easy (*separable*) problems, i.e., those on which the machine f makes no training errors, the exponent is $\alpha = 1$, which translates into machines learning at a *fast* rate of $O(n^{-1})$. The difference between these two rates is huge: the $O(n^{-1})$ learning rate potentially only requires 1000 examples to achieve the accuracy for which the $O(n^{-1/2})$ learning rate needs 10^6 examples. So, given a student who learns from a fixed amount of data n and a function class $\mathcal{F}$, a good teacher can try to ease the problem at hand by accelerating the learning rate from $O(n^{-1/2})$ to $O(n^{-1})$.

Vapnik's *learning using privileged information* is one example of what we call *machines-teaching-machines*: the paradigm where machines learn from other machines, in addition to training data. Another seemingly unrelated example is *distillation* (Hinton et al., 2015),[1] where a simple machine learns a complex task by imitating the solution of a flexible machine. In a wider context, the machines-teaching-machines paradigm is one step toward the definition of *machine reasoning* of Bottou (2014), "the algebraic manipulation of previously acquired knowledge to answer a new question". In fact, recent state-of-the-art systems compose data and supervision from multiple sources, such as object recognizers reusing convolutional neural network features (Oquab et al., 2014), and natural language processing systems operating on vector word representations extracted from unsupervised text corpora (Mikolov et al., 2013).

In the following, we frame Hinton's distillation and Vapnik's privileged information as two instances of the same machines-teaching-machines paradigm, termed *generalized distillation*. The analysis of generalized distillation sheds light to applications in semisupervised learning, domain adaptation, transfer learning, Universum learning (Weston et al., 2006), reinforcement learning, and curriculum learning (Bengio et al., 2009); some of them discussed in our numerical simulations.

8.1.1 Distillation

We focus on c-class classification, although the same ideas apply to regression. Consider the data

$$\{(x_i, y_i)\}_{i=1}^n \sim P^n(x, y),\ x_i \in \mathbb{R}^d,\ y_i \in \Delta^c. \tag{8.1}$$

Here, Δ^c is the set of c-dimensional probability vectors. Using (8.1), we target learning the representation

$$f_t = \underset{f \in \mathcal{F}_t}{\arg\min} \frac{1}{n} \sum_{i=1}^n \ell(y_i, \sigma(f(x_i))) + \Omega(\|f\|), \tag{8.2}$$

[1] Distillation relates to *model compression* (Buciluǎ et al., 2006; Ba and Caruana, 2014). We will adopt the term *distillation* throughout this section.

where $\mathcal{F}_t$ is a class of functions from $\mathbb{R}^d$ to $\mathbb{R}^c$, the function $\sigma : \mathbb{R}^c \to \Delta^c$ is the softmax operation

$$\sigma(z)_k = \frac{e^{z_k}}{\sum_{j=1}^{c} e^{z_j}},$$

for all $1 \leq k \leq c$, the function $\ell : \Delta^c \times \Delta^c \to \mathbb{R}_+$ is the cross-entropy loss

$$\ell(y, \hat{y}) = -\sum_{k=1}^{c} y_k \log \hat{y}_k,$$

and $\Omega : \mathbb{R} \to \mathbb{R}$ is an increasing function which serves as a regularizer.

When learning from real world data such as high-resolution images, f_t is often an ensemble of large deep convolutional neural networks (LeCun et al., 1998a). The computational cost of predicting new examples at test time using these ensembles is often prohibitive for production systems. For this reason, Hinton et al. (2015) propose to *distill* the learned representation $f_t \in \mathcal{F}_t$ into

$$f_s = \operatorname*{arg\,min}_{f \in \mathcal{F}_s} \frac{1}{n} \sum_{i=1}^{n} \Big[(1-\lambda)\ell(y_i, \sigma(f(x_i))) + \lambda \ell(s_i, \sigma(f(x_i))) \Big], \tag{8.3}$$

where

$$s_i = \sigma(f_t(x_i)/T) \in \Delta^c$$

are the *soft predictions* from f_t about the training data, and $\mathcal{F}_s$ is a function class simpler than $\mathcal{F}_t$. The temperature parameter $T > 0$ controls how much do we want to soften or smooth the class-probability predictions from f_t, and the imitation parameter $\lambda \in [0, 1]$ balances the importance between imitating the soft predictions s_i and predicting the true hard labels y_i. Higher temperatures lead to softer class-probability predictions s_i. In turn, softer class-probability predictions reveal label dependencies which would be otherwise hidden as extremely large or small numbers. After distillation, we can use the simpler $f_s \in \mathcal{F}_s$ for faster prediction at test time.

8.1.2 Privileged information

We now turn back to Vapnik's problem of learning in the company of an intelligent teacher, as introduced in the opening of this section. The question at hand is: How can we leverage the privileged information $x_i^\star$ to build a better classifier for test time? One naïve way to proceed would be to estimate the privileged representation $x_i^\star$ from the regular representation x_i, and then use the union of regular and *estimated* privileged representations as our test-time feature space. But this may be a cumbersome endeavour: in the example of biopsy images x_i and medical reports $x_i^\star$, it is reasonable to believe that predicting reports from images is more complicated than classifying the images into cancerous or healthy.

Alternatively, we propose to use distillation to extract useful knowledge from privileged information. The proposal is as follows. First, learn a teacher function $f_t \in \mathcal{F}_t$ by solving (8.2) using the data $\{(x_i^\star, y_i)\}_{i=1}^n$. Second, compute the teacher soft labels $s_i = \sigma(f_t(x_i^\star)/T)$, for all $1 \leq i \leq n$ and some temperature parameter $T > 0$. Third, distill $f_t \in \mathcal{F}_t$ into $f_s \in \mathcal{F}_s$ by solving (8.3) using both the hard labeled data $\{(x_i, y_i)\}_{i=1}^n$ and the softly labeled data $\{(x_i, s_i)\}_{i=1}^n$.

Comparison to prior work

Vapnik and Vashist (2009); Vapnik and Izmailov (2015) offer two strategies to learn using privileged information: similarity control and knowledge transfer. Let us briefly compare them to our distillation-based proposal.

The motivation behind *similarity control* is that SVM classification is separable after we correct for the *slack values* ξ_i, which measure the degree of misclassification of training data points x_i (Vapnik and Vashist, 2009). Since separable classification admits $O(n^{-1})$ fast learning rates, it would be ideal to have a teacher that could supply slack values to us. Unluckily, it seems quixotic to aspire for a teacher able to provide with abstract floating point number slack values. Perhaps it is more realistic to assume instead that the teacher can provide with some rich, high-level representation useful to estimate the sought-after slack values. This reasoning crystallizes into the SVM+ objective function from (Vapnik and Vashist, 2009):

$$L(w, w^\star, b, b^\star, \alpha, \beta) = \underbrace{\frac{1}{2}\|w\|^2 + \sum_{i=1}^n \alpha_i - \sum_{i=1}^n \alpha_i y_i f_i}_{\text{separable SVM objective}} + \underbrace{\frac{\gamma}{2}\|w^\star\|^2 + \sum_{i=1}^n (\alpha_i + \beta_i - C) f_i^\star}_{\text{corrections from teacher}},$$

where $f_i := \langle w, x_i \rangle + b$ is the decision boundary at x_i, and $f_i^\star := \langle w^\star, x_i^\star \rangle + b^\star$ is the teacher correcting function at the same location. The SVM+ objective function matches the objective function of not separable SVM when we replace the correcting functions $f_i^\star$ with the slacks ξ_i. Thus, skilled teachers provide with privileged information $x_i^\star$ highly informative about the slack values ξ_i. Such privileged information allows for simple correcting functions $f_i^\star$, and the easy estimation of these correcting functions is a proxy to $O(n^{-1})$ fast learning rates. Technically, this amounts to saying that a teacher is helpful whenever the capacity of her correcting functions is much smaller than the capacity of the student decision boundary.

In *knowledge transfer* (Vapnik and Izmailov, 2015) the teacher fits a function $f_t(x^\star) = \sum_{j=1}^m \alpha_j^\star k^\star(u_j^\star, x^\star)$ on the input-output pairs $\{(x_i^\star, y_i)\}_{i=1}^n$

and $f_t \in \mathcal{F}_t$, to find the best reduced set of prototype or basis points $\{u_j^\star\}_{j=1}^m$. Second, the student fits one function g_j per set of input-output pairs $\{(x_i, k^\star(u_j^\star, x_i^\star))\}_{i=1}^n$, for all $1 \leq j \leq m$. Third, the student fits a new vector of coefficients $\alpha \in \mathbb{R}^m$ to obtain the final student function $f_s(x) = \sum_{j=1}^m \alpha_j g_j(x)$, using the input-output pairs $\{(x_i, y_i)\}_{i=1}^n$ and $f_s \in \mathcal{F}_s$. Since the representation $x_i^\star$ is intelligent, we assume that the function class $\mathcal{F}_t$ has small capacity, and thus allows for accurate estimation under small sample sizes.

Distillation differs from similarity control in three ways. First, unlike SVM+, distillation is not restricted to SVMs. Second, while the SVM+ solution contains twice the amount of parameters than the original SVM, the user can choose a priori the amount of parameters in the distilled classifier. Third, SVM+ learns the teacher correcting function and the student decision boundary simultaneously, but distillation proceeds sequentially: first with the teacher, then with the student. On the other hand, knowledge transfer is closer in spirit to distillation, but the two techniques differ: while similarity control relies on a student that purely imitates the hidden representation of a low-rank kernel machine, distillation is a trade-off between imitating soft predictions and hard labels, using arbitrary learning algorithms.

The framework of learning using privileged information enjoys theoretical analysis (Pechyony and Vapnik, 2010), equivalence analysis to weighted learning (Lapin et al., 2014), and multiple applications that include ranking (Sharmanska et al., 2013), computer vision (Sharmanska et al., 2014; Lopez-Paz et al., 2014), clustering (Feyereisl and Aickelin, 2012), metric learning (Fouad et al., 2013), Gaussian process classification (Hernández-Lobato et al., 2014), and finance (Ribeiro et al., 2010).

8.1.3 Generalized distillation

We now have all the necessary background to describe *generalized distillation*. To this end, consider the data $\{(x_i, x_i^\star, y_i)\}_{i=1}^n$. Then, the process of generalized distillation is as follows:

1. Learn teacher $f_t \in \mathcal{F}_t$ using the input-output pairs $\{(x_i^\star, y_i)\}_{i=1}^n$ and Eq. 8.2.

2. Compute teacher soft labels $\{\sigma(f_t(x_i^\star)/T)\}_{i=1}^n$, using temperature parameter $T > 0$.

3. Learn student $f_s \in \mathcal{F}_s$ using the input-output pairs $\{(x_i, y_i)\}_{i=1}^n$, $\{(x_i, s_i)\}_{i=1}^n$, Eq. 8.3, and imitation parameter $\lambda \in [0, 1]$.

We say that generalized distillation reduces to *Hinton's distillation* if $x_i^\star = x_i$ for all $1 \leq i \leq n$ and $|\mathcal{F}_s|_C \ll |\mathcal{F}_t|_C$, where $|\cdot|_C$ is an appropriate function class capacity measure. Conversely, we say that generalized distillation

reduces to *Vapnik's learning using privileged information* if $x_i^\star$ is a privileged description of x_i, and $|\mathcal{F}_s|_{\mathrm{C}} \gg |\mathcal{F}_t|_{\mathrm{C}}$.

This comparison reveals a subtle difference between Hinton's distillation and Vapnik's privileged information. In Hinton's distillation, $\mathcal{F}_t$ is *flexible*, for the teacher to exploit her *general purpose* representation $x_i^\star = x_i$ to learn intricate patterns from *large* amounts of labeled data. In Vapnik's privileged information, $\mathcal{F}_t$ is *simple*, for the teacher to exploit her *rich* representation $x_i^\star \neq x_i$ to learn intricate patterns from *small* amounts of labeled data. The space of privileged information is thus a specialized space, one of "metaphoric language". In our running example of biopsy images, the space of medical reports is much more specialized than the space of pixels, since the space of pixels can also describe buildings, animals, and other unrelated concepts. In any case, the teacher must develop a language that effectively communicates information to help the student come up with better representations. The teacher may do so by incorporating invariances, or biasing them towards being robust with respect to the kind of distribution shifts that the teacher may expect at test time. In general, having a teacher is one opportunity to learn characteristics about the decision boundary which are not contained in the training sample, in analogy to a good Bayesian prior.

Why does generalized distillation work?

Recall our three actors: the student function $f_s \in \mathcal{F}_s$, the teacher function $f_t \in \mathcal{F}_t$, and the real target function of interest to both the student and the teacher, $f \in \mathcal{F}$. For simplicity, consider *pure distillation* (set the imitation parameter to $\lambda = 1$). Furthermore, we will place some assumptions about how the student, teacher, and true function interplay when learning from n data. First, assume that the student may learn the true function at a slow rate

$$R(f_s) - R(f) \leq O\left(\frac{|\mathcal{F}_s|_{\mathrm{C}}}{\sqrt{n}}\right) + \varepsilon_s,$$

where the $O(\cdot)$ term is the estimation error, and ε_s is the approximation error of the student function class $\mathcal{F}_s$ with respect to $f \in \mathcal{F}$. Second, assume that the better representation of the teacher allows her to learn at the fast rate

$$R(f_t) - R(f) \leq O\left(\frac{|\mathcal{F}_t|_{\mathrm{C}}}{n}\right) + \varepsilon_t,$$

where ε_t is the approximation error of the teacher function class $\mathcal{F}_t$ with respect to $f \in \mathcal{F}$. Finally, assume that when the student learns from the teacher, she does so at the rate

$$R(f_s) - R(f_t) \leq O\left(\frac{|\mathcal{F}_s|_{\mathrm{C}}}{n^\alpha}\right) + \varepsilon_l,$$

where ε_l is the approximation error of the student function class $\mathcal{F}_s$ with respect to $f_t \in \mathcal{F}_t$, and $\frac{1}{2} \leq \alpha \leq 1$. Then, the rate at which the student learns the true function f admits the alternative expression

$$\begin{aligned} R(f_s) - R(f) &= R(f_s) - R(f_t) + R(f_t) - R(f) \\ &\leq O\left(\frac{|\mathcal{F}_s|_{\mathrm{C}}}{n^\alpha}\right) + \varepsilon_l + O\left(\frac{|\mathcal{F}_t|_{\mathrm{C}}}{n}\right) + \varepsilon_t \\ &\leq O\left(\frac{|\mathcal{F}_s|_{\mathrm{C}} + |\mathcal{F}_t|_{\mathrm{C}}}{n^\alpha}\right) + \varepsilon_l + \varepsilon_t, \end{aligned}$$

where the last inequality follows because $\alpha \leq 1$. Thus, the question at hand is to argue, for a given learning problem, if the inequality

$$O\left(\frac{|\mathcal{F}_s|_{\mathrm{C}} + |\mathcal{F}_t|_{\mathrm{C}}}{n^\alpha}\right) + \varepsilon_l + \varepsilon_t \leq O\left(\frac{|\mathcal{F}_s|_{\mathrm{C}}}{\sqrt{n}}\right) + \varepsilon_s$$

holds. The inequality highlights that the benefits of learning with a teacher arise due to i) the capacity of the teacher being small, ii) the approximation error of the teacher being smaller than the approximation error of the student, and iii) the coefficient α being greater than $\frac{1}{2}$. Remarkably, these factors embody the assumptions of privileged information from Vapnik and Izmailov (2015). The inequality is also reasonable under the main assumption in (Hinton et al., 2015), which is $\varepsilon_s \gg \varepsilon_t + \varepsilon_l$. Moreover, the inequality highlights that the teacher is most helpful in low data regimes; for instance, when working with small datasets, or in the initial stages of online and reinforcement learning.

We believe that the "$\alpha > \frac{1}{2}$ case" is a general situation, since soft labels (dense vectors with a real number of information per class) contain more information than hard labels (one-hot-encoding vectors with one bit of information per class) per example, and should allow for faster learning. This additional information, also understood as label uncertainty, relates to the acceleration in SVM+ due to the knowledge of slack values. Since a good teacher smoothes the decision boundary and instructs the student to fail on difficult examples, the student can focus on the remaining body of data. Although this translates into the unambitious "whatever my teacher could not do, I will not do", the imitation parameter $\lambda \in [0, 1]$ in (8.3) allows to follow this rule safely, and fall back to regular learning if necessary.

Extensions

Semi-supervised learning We now extend generalized distillation to the situation where examples lack regular features, privileged features, labels, or a combination of the three. In the following, we denote missing elements by $\square$. For instance, the example $(x_i, \square, y_i)$ has no privileged features, and the

example $(x_i, x_i^\star, \square)$ is missing its label. Using this convention, we introduce the *clean subset* notation

$$c(S) = \{v : v \in S, v_i \neq \square \ \forall i\}.$$

Then, semisupervised generalized distillation walks the same three steps as generalized distillation, enumerated at the beginning of Section 8.1.3, but uses the appropriate clean subsets instead of the whole data. For example, the semisupervised extension of distillation allows the teacher to prepare soft labels for all the unlabeled data $c(\{(x_i, x_i^\star)\}_{i=1}^n)$. These additional soft-labels are additional information available to the student to learn the teacher representation f_t.

Learning with the Universum The unlabeled data $c(\{x_i, x_i^\star\}_{i=1}^n)$ can belong to one of the classes of interest, or be *Universum* data (Weston et al., 2006). Universum data may have labels: in this case, one can exploit these additional labels by i) training a teacher that distinguishes amongst all classes (those of interest and those from the Universum), ii) computing soft class-probabilities only for the classes of interest, and iii) distilling these soft probabilities into a student function.

Learning from multiple tasks Generalized distillation applies to some domain adaptation, transfer learning, or multitask learning scenarios. On the one hand, if the multiple tasks share the same labels y_i but differ in their input modalities, the input modalities from the source tasks are privileged information. On the other hand, if the multiple tasks share the same input modalities x_i but differ in their labels, the labels from the source tasks are privileged information. In both cases, the regular student representation is the input modality from the target task.

Curriculum and reinforcement learning We conjecture that the uncertainty in the teacher soft predictions is a mechanism to rank the difficulty of training examples, and use these ranks for curriculum learning (Bengio et al., 2009). Furthermore, distillation resembles imitation, a technique that learning agents could exploit in *reinforcement learning* environments.

A causal perspective on generalized distillation

The assumption of *independence of cause and mechanisms* states that "the probability distribution of a cause is often independent from the process mapping this cause into its effects" (Schölkopf et al., 2012). Under this assumption, for instance, *causal learning problems* —i.e., those where the features cause the labels— do not benefit from semisupervised learning, since by the independence assumption, the marginal distribution of the

features contains no information about the function mapping features to labels. Conversely, *anticausal learning problems* —those where the labels cause the features— may benefit from semisupervised learning.

Causal implications also arise in generalized distillation. First, if the privileged features $x_i^\star$ only add information about the marginal distribution of the regular features x_i, the teacher should be able to help only in anticausal learning problems. Second, if the teacher provides additional information about the conditional distribution of the labels y_i given the inputs x_i, it should also help in the causal setting. We will confirm this hypothesis in the next section.

8.1.4 Numerical simulations

We now present some experiments to illustrate when the distillation of privileged information is effective, and when it is not.

We start with four synthetic experiments, designed to minimize modeling assumptions and to illustrate different prototypical types of privileged information. These are simulations of logistic regression models repeated over 100 random partitions, where we use $n_{\text{tr}} = 200$ samples for training, and $n_{\text{te}} = 10,000$ samples for testing. The dimensionality of the regular features x_i is $d = 50$, and the involved separating hyperplanes $\alpha \in \mathbb{R}^d$ follow the distribution $\mathcal{N}(0, I_d)$. For each experiment, we report the test accuracy when i) using the teacher explanations $x_i^\star$ at both train and test time, ii) using the regular features x_i at both train and test time, and iii) distilling the teacher explanations into the student classifier with $\lambda = T = 1$.

1. Clean labels as privileged information. We sample triplets $(x_i, x_i^\star, y_i)$ from:

$$\begin{aligned} x_i &\sim \mathcal{N}(0, I_d) \\ x_i^\star &\leftarrow \langle \alpha, x_i \rangle \\ \varepsilon_i &\sim \mathcal{N}(0, 1) \\ y_i &\leftarrow \mathbb{I}((x_i^\star + \varepsilon_i) > 0). \end{aligned}$$

Here, each teacher explanation $x_i^\star$ is the exact distance to the decision boundary for each x_i, but the data labels y_i are corrupt. This setup aligns with the assumptions about slacks in the similarity control framework of Vapnik and Vashist (2009). We obtained a privileged test classification accuracy of $96 \pm 0\%$, a regular test classification accuracy of $88 \pm 1\%$, and a distilled test classification accuracy of $95 \pm 1\%$. This illustrates that distillation of privileged information is an effective mean to detect outliers in label space.

2. Clean features as privileged information We sample triplets $(x_i, x_i^\star, y_i)$ from:

$$\begin{aligned} x_i^\star &\sim \mathcal{N}(0, I_d) \\ \varepsilon_i &\sim \mathcal{N}(0, I_d) \\ x_i &\leftarrow x_i^\star + \varepsilon \\ y_i &\leftarrow \mathbb{I}\left(\langle \alpha, x_i^\star \rangle > 0\right). \end{aligned}$$

In this setup, the teacher explanations $x_i^\star$ are clean versions of the regular features x_i available at test time. We obtained a privileged test classification accuracy of $90 \pm 1\%$, a regular test classification accuracy of $68 \pm 1\%$, and a distilled test classification accuracy of $70 \pm 1\%$. This improvement is not statistically significant. This is because the intelligent explanations $x_i^\star$ are independent from the noise ε_i polluting the regular features x_i. Therefore, there exists no additional information transferable from the teacher to the student.

3. Relevant features as privileged information We sample triplets $(x_i, x_i^\star, y_i)$ from:

$$\begin{aligned} x_i &\sim \mathcal{N}(0, I_d) \\ x_i^\star &\leftarrow x_{i,J} \\ y_i &\leftarrow \mathbb{I}(\langle \alpha_J, x_i^\star \rangle > 0), \end{aligned}$$

where the set J, with $|J| = 3$, is a subset of the variable indices $\{1, \ldots, d\}$ chosen at random but common for all samples. In another words, the teacher explanations indicate the values of the variables relevant for classification, which translates into a reduction of the dimensionality of the data that we have to learn from. We obtained a privileged test classification accuracy of $98 \pm 0\%$, a regular test classification accuracy of $89 \pm 1\%$, and a distilled test classification accuracy of $97 \pm 1\%$. This illustrates that distillation on privileged information is an effective tool for feature selection.

4. Sample-dependent relevant features as privileged information Sample triplets

$$\begin{aligned} x_i &\sim \mathcal{N}(0, I_d) \\ x_i^\star &\leftarrow x_{i,J_i} \\ y_i &\leftarrow \mathbb{I}(\langle \alpha_{J_i}, x_i^\star \rangle > 0), \end{aligned}$$

where the sets J_i, with $|J_i| = 3$ for all i, are a subset of the variable indices $\{1, \ldots, d\}$ chosen at random for each sample $x_i^\star$. One interpretation of such model is the one of bounding boxes in computer vision: each high-dimensional

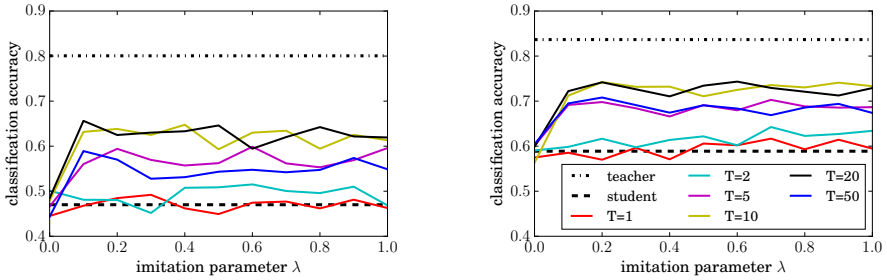


Figure 8.1: Distillation results on MNIST for 300 and 500 samples.

vector x_i would be an image, and each teacher explanation $x_i^\star$ would be the pixels inside a bounding box locating the concept of interest (Sharmanska et al., 2013). We obtained a privileged test classification accuracy of $96\pm2\%$, a regular test classification accuracy of $55\pm3\%$, and a distilled test classification accuracy of $0.56 \pm 4\%$. Note that although the classification is linear in $x^\star$, this is not the case in terms of x. Therefore, although we have misspecified the function class $\mathcal{F}_s$ for this problem, the distillation approach did not deteriorate the final performance.

The previous four experiments set up causal learning problems. In the second experiment, the privileged features $x_i^\star$ add no information about the target function mapping the regular features to the labels, so the causal hypothesis from Section 8.1.3 justifies the lack of improvement. The first and third experiments provide privileged information that adds information about the target function, and therefore is beneficial to distill this information. The fourth example illustrates that the privileged features adding information about the target function is not a sufficient condition for improvement.

5. MNIST handwritten digit image classification The privileged features are the original 28x28 pixels MNIST handwritten digit images (LeCun et al., 1998b), and the regular features are the same images downscaled to 7x7 pixels. We use 300 or 500 samples to train both the teacher and the student, and test their accuracies at multiple levels of temperature and imitation on the full test set. Both student and teacher are neural networks of composed by two hidden layers of 20 rectifier linear units and a softmax output layer (as in the remaining experiments). Figure 8.1 summarizes the results of this experiment, where we see a significant improvement in classification accuracy when distilling the privileged information, with respect to using the regular features alone. As expected, the benefits of distillation diminished as we further increased the sample size.

6. Semisupervised learning We explore the semisupervised capabilities of generalized distillation on the CIFAR10 dataset (Krizhevsky, 2009). Here,

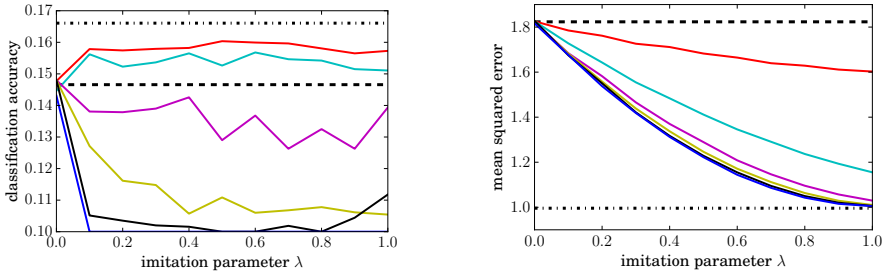


Figure 8.2: Distillation results on CIFAR 10 and SARCOS.

the privileged features are the original 32x32 pixels CIFAR10 color images, and the regular features are the same images when polluted with additive Gaussian noise. We provide labels for 300 images, and unlabeled privileged and regular features for the rest of the training set. Thus, the teacher trains on 300 images, but computes the soft labels for the whole training set of $50,000$ images. The student then learns by distilling the 300 original hard labels and the $50,000$ soft predictions. As seen in Figure 8.2, the soft labeling of unlabeled data results in a significant improvement with respect to pure student supervised classification. Distillation on the 300 labeled samples did not improve the student performance. This illustrates the importance of semisupervised distillation in this data. We believe that the drops in performance for some distillation temperatures are due to the lack of a proper weighting between labeled and unlabeled data in (8.3).

7. Multitask learning The SARCOS dataset[2] characterizes the 7 joint torques of a robotic arm given 21 real-valued features. Thus, this is a multitask learning problem, formed by 7 regression tasks. We learn a teacher on 300 samples to predict each of the 7 torques given the other 6, and then distill this knowledge into a student who uses as her regular input space the 21 real-valued features. Figure 8.2 illustrates the performance improvement in mean squared error when using generalized distillation to address the multitask learning problem. When distilling at the proper temperature, distillation allowed the student to match her teacher performance.

Machine adapters

Consider a machine $f : \mathbb{R}^d \to \mathbb{R}^c$ trained on some task, and a collection of unlabeled data $\{x_i\}_{i=1}^n$, $x_i \in \mathbb{R}^q$, related to a new but related task. For instance, we may exploit the knowledge contained in f by learning an *adapter*

[2] `http://www.gaussianprocess.org/gpml/data/`

$a : \mathbb{R}^q \to \mathbb{R}^d$ that minimizes the reconstruction loss

$$L(a, g; x, f) = \frac{1}{n} \sum_{i=1}^{n} \|x_i - g(f(a(x_i)))\|,$$

where $g : \mathbb{R}^c \to \mathbb{R}^q$. The resulting machine $f(a(x))$ would simultaneously contain knowledge from f (for instance, high-level visual features) and most of the information from the new data $\{x_i\}_{i=1}^n$. Alternatively, one could also train the adapter a by using labeled data and a supervised objective, or a generative adversarial network (see Section 4.2.3), or an unsupervised objective function on the output statistics of f.

8.2 Theory of nonconvex optimization

When learning a function g using empirical risk minimization over a function class $\mathcal{F}$ and dataset $D \sim P^n$, the error of a computed solution $\tilde{f} \in \mathcal{F}$ is

$$\begin{aligned}
\mathcal{E} &= \mathbb{E}\Big[R(\tilde{f}) - R(g)\Big] \\
&= \mathbb{E}\Big[R(\tilde{f}) - R(\hat{f})\Big] + \mathbb{E}\Big[R(\hat{f}) - R(f^\star)\Big] + \mathbb{E}[R(f^\star) - R(g)] \\
&= \mathcal{E}_{\text{opt}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{app}}.
\end{aligned}$$

First, the term $\mathcal{E}_{\text{app}}$ is the *approximation error* due to the difference between the expected risk minimizer $f^\star \in \mathcal{F}$ and the target function g. Second, the term $\mathcal{E}_{\text{est}}$ is the *estimation error* due to the difference between the expected risk minimizer $f^\star$ and the empirical risk minimizer $\hat{f} \in \mathcal{F}$. Third, the term $\mathcal{E}_{\text{opt}}$ is the *optimization error* due to the difference between the empirical risk minimizer $\hat{f}$ and the computed solution $\tilde{f}$. Optimization errors arise due to the imprecisions of the numerical computation of $\tilde{f} \in \mathcal{F}$, such as the local minima of nonconvex empirical risk minimization problems and limited computational budgets.

Observe that if $g \in \mathcal{F}$ or $\mathcal{F}$ is universally consistent, then $\mathcal{E}_{\text{app}} = 0$. Second, the generalization error $\mathcal{E}_{\text{est}}$ is inversely proportional to the amount of available training data, and directly proportional to the flexibility of $\mathcal{F}$ as measured, for instance, using Rademacher complexities (Bartlett and Mendelson, 2003) or stability criteria (Hardt et al., 2015). For convex learning problems and gradient-based numerical optimization routines (Section 2.4), the optimization error $\mathcal{E}_{\text{opt}}$ is inversely proportional to the number of iterations (Bousquet and Bottou, 2008). However, for general nonconvex learning problems, such as deep or convolutional neural networks (Bengio et al., 2015), we have no guarantees about the optimization error, that is, the difference between $\hat{f}$ and $\tilde{f}$. This is a big caveat: nonconvex empirical risk minimization is NP-hard, and the theory of empirical risk minimization only holds if we can find the empirical risk minimizer (Section 2.3.1).

Let us exemplify the goal of this section by using the language of neural networks. To this end, assume data $\mathcal{D} = \{(x_i, g(x_i))\}_{i=1}^n$, where g is a neural network with h hidden layers of w neurons each. Using empirical risk minimization over the data $\mathcal{D}$ and a neural network with $H \geq h$ hidden layers of $W \geq w$ neurons each, we obtain the solution $\tilde{f}$. Because of nonconvexity, the solution $\tilde{f}$ may be worse than the empirical risk minimizer $\hat{f}$. Also, since we are in a realizable learning situation, the empirical risk minimizer has zero approximation error. Therefore, we are interested in characterizing the optimization error as the tail probability

$$\mathbb{P}\left(R(\tilde{f}) > t\right) \leq g(n, w, h, W, H), \tag{8.4}$$

where the randomness is due to the random initialization of the neural network parameters provided to the gradient-based optimizer. We propose to study (8.4) by sketching two novel concepts: convexity generalizations and continuation methods.

8.2.1 Convexity generalizations

One way to study nonconvex functions is to compare the quality of their local minima. We do this by introducing two generalizations of convexity: α-convexity and ε-convexity. The first one, α-convexity, measures how much does the quality of two random local minima of f differ.

Definition 8.1 (α-convexity)**.** *A function $f : \Omega \to [0, 1]$ is α-convex if, for two local minima $w, w' \in \Omega$, it follows that*

$$\mathbb{P}_{w,w'}\left(\left|f(w) - f(w')\right| > t\right) \leq C_f \exp(-c_f t^2),$$

for some constants $C_f, c_f > 0$.

Therefore, the local minima of functions with an α-convexity profile that decays fast will be similar in value, and in particular, similar in value to the global minima. Alternatively, ε-convexity measures how much does a differentiable function f depart from a convex function.

Definition 8.2 (ε-convexity)**.** *A differentiable function $f : \Omega \to [0, 1]$ is ε-convex if, for all $w, w' \in \Omega$, it follows that*

$$\mathbb{P}_{w,w'}\left(f(w') - f(w) - \nabla f(w)^\top (w' - w) > t\right) \leq C_f \exp(-c_f t^2),$$

for some constants $C_f, c_f > 0$.

Similar definitions for ε-convexity follow by using zero-order or second-order conditions. We conjecture that optimizing a function f with a ε-convexity profile that decays fast will be similar to optimizing a convex function; this may translate into guarantees about the relationship between the local and global minima of f.

The convexity of deep neural networks

We hope that α-convexity and ε-convexity will aid the investigation of the loss surface of multilayer neural networks (Choromanska et al., 2015). We believe this because of two intuitions. First, the local minima of large neural networks have better value than the global minima of small neural networks. This should translate into a good α-convexity profile, and the fast decay of the tail probability (8.4). In practice, to obtain the quality of the empirical risk minimizer from a set of small neural networks, practitioners simply train to local optimality a large neural network. Second, large neural networks are highly redundant (Denil et al., 2013). Thus, it is not critical to misconfigure some of the parameters of these networks, since we can leverage the redundancy provided by the remaining parameters to keep descending down the loss surface. Actually, it is known that the amount of local minima decreases exponentially with higher optimization dimensionality (Dauphin et al., 2014), and that the challenges of high-dimensional nonconvex optimization are mostly due to saddle points. Our intuition is that these thoughts relate to the α-convexity and ε-convexity profiles of neural network empirical risk minimization, as well as to the tail probability (8.4). To turn intuition into mathematics, it would be desirable to obtain expressions for the ε-convexity and the α-convexity of deep neural networks in terms of their number of their hidden layers and neurons. These results would be a remarkable achieving, and would provide deep neural networks with the necessary theory for their empirical risk minimization.

8.2.2 Continuation methods

Continuation methods (Mobahi and Fisher III, 2015) tackle nonconvex optimization problems by first solving an easy optimization problem, and then progressively morphing this easy problem into the nonconvex problem of interest. Here, we propose a simple way of implementing continuation methods in neural networks. Our proposal is based on two observations. First, the only nonlinear component in neural networks is their activation function $\sigma : \mathbb{R} \to \mathbb{R}$. Second, for linear activation functions, we can solve neural networks optimally (Baldi and Hornik, 1989). Therefore, let us replace the activation functions σ in a neural network with

$$\sigma_\alpha(z) = (1 - \alpha)z + \alpha\sigma(z),$$

where $0 \leq \alpha \leq 1$. For $\alpha = 0$, the neural network is linear. For $\alpha = 1$, the neural network is nonlinear. For $0 < \alpha < 1$, the neural network has an intermediate degree of nonlinearity. The continuation scheme would be to first minimize our neural network equipped with activation functions σ_0, and then reuse the solution to minimize the same neural network with activation functions $\sigma_\epsilon, \sigma_{2\epsilon}, \ldots, \sigma_1$, for some small $0 < \epsilon < 1$.

We believe that investigating the quality of neural networks obtained with a continuation method like the one described above is an interesting research question. How different are two solutions obtained with this continuation method? How do these solutions compare to the solutions obtained from usual backpropagation? Can we relate the solutions obtained using our continuation method to the global minima, under additional assumptions and for very small ϵ?

8.3 The supervision continuum

The mathematical difference between supervised and unsupervised learning is subtle: in the end, both are the minimization of a loss function. For instance, in the supervised task of *classification* we "learn a function $f : \mathbb{R}^d \to \Delta^c$ using the loss ℓ_{sup} and the *labeled* data $\{(x_i, y_i)\}_{i=1}^n$". On the other hand, in the unsupervised task of *clustering* we "learn a function $f : \mathbb{R}^d \to \Delta^c$ using the loss ℓ_{unsup} and the *unlabeled* data $\{x_i\}_{i=1}^n$".

The main difference between the previous is that in supervised learning we have a clear picture about how ℓ_{sup} should look like, but in unsupervised learning, the shape of ℓ_{unsup} depends on the type of learning tasks that we expect to confront in the future. Supervised and unsupervised learning are the two ends of the *supervision continuum*. Everything in between is a situation where our training data is a mixture of labeled and unlabeled examples. We formalize this by writing the examples comprising our data as

$$(x_i, y_i) \in (\mathbb{R}^d \cup \square) \times (\Delta^c \cup \square),$$

where $x_i = \square$ or $y_i = \square$ means "not available".

As the percentage of labeled examples in our data grows, so does the *supervision level* of the learning problem at hand. So, supervision is not a matter of two extremes, but characterized as a *continuum*. Therefore, it makes sense to ask if there exists a single learning machine that can deal efficiently with the whole supervision continuum, or if we need fundamentally different algorithms to deal with different levels of supervision. One example of an algorithm dealing with the supervision continuum is the *ladder network* of Rasmus et al. (2015), which mixes a cross-entropy objective for labeled examples with a reconstruction objective for unlabeled examples. However, it would be interesting to develop unsupervised objectives alternative to reconstruction error, which do not involve learning a whole complicated decoder function (for instance, favour low density decision boundaries or large margin for unlabeled samples as in transductive learning).

The supervision continuum extends to multitask learning. To see this, write the examples comprising our data as

$$(x_i, y_i, t_i) \in (\mathbb{R} \cup \square)^{d_{t_i}} \times (\Delta \cup \square)^{c_{t_i}} \times (\mathbb{Z} \cup \square),$$

where $t_i \in (\mathbb{Z}, \square)$ is the "task identification number for the i-th example", and two examples (x_i, y_i, t_i) and (x_j, y_j, t_j) may have inputs and outputs defined on different spaces. Said differently, we may not know to which task some of the examples in our data belong. This is similar to human learning: we are constantly presented with a stream of data, that we exploit to get better at different but related learning tasks. However, in many cases these tasks are not explicitly identified. How can a machine deal with this additional continuum of supervision?

Bibliography

Aas, K., Czado, C., Frigessi, A., and Bakken, H. Pair-copula constructions of multiple dependence. *Insurance: Mathematics and economics*, 44(2): 182–198, 2009. (pp. 95 and 100.)

Acar, E. F., Genest, C., and Nešlehová, J. Beyond simplified pair-copula constructions. *Journal of Multivariate Analysis*, 110:74–90, 2012. (pp. 95, 100, and 101.)

Achlioptas, D., McSherry, F., and Schölkopf, B. Sampling techniques for kernel methods. In *NIPS*, volume 1, page 335. MIT Press, 2002. (pp. 109.)

Altobelli, N., Lopez-Paz, D., Pilorz, S., Spilker, L. J., Morishima, R., Brooks, S., Leyrat, C., Deau, E., Edgington, S., and Flandes, A. Two numerical models designed to reproduce saturn ring temperatures as measured by Cassini-CIRS. *Icarus*, 238:205–220, 2014. (pp. 7.)

Andrew, G., Arora, R., Bilmes, J., and Livescu, K. Deep canonical correlation analysis. In *ICML*, pages 1247–1255, 2013. (pp. 125 and 126.)

Asuncion, A. and Newman, D. UCI machine learning repository, 2007. (pp. 180.)

Avron, H., Boutsidis, C., Toledo, S., and Zouzias, A. Efficient dimensionality reduction for canonical correlation analysis. *SIAM Journal on Scientific Computing*, 36(5):S111–S131, 2014. (pp. 109.)

Ba, J. and Caruana, R. Do deep nets really need to be deep? In *NIPS*, 2014. (pp. 203.)

Bach, F. R. and Jordan, M. I. Kernel independent component analysis. *JMLR*, 3:1–48, 2002. (pp. 108, 112, 113, 117, and 130.)

Baldi, P. and Hornik, K. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1): 53–58, 1989. (pp. 136 and 216.)

Bartlett, P. L. and Mendelson, S. Rademacher and Gaussian complexities: risk bounds and structural results. *JMLR*, 3:463–482, 2003. (pp. 171, 192, and 214.)

Bartlett, P. L. and Mendelson, S. Empirical minimization. *Probability Theory and Related Fields*, 135(3):311–334, 2006. (pp. 194.)

Bartlett, P. L., Bousquet, O., and Mendelson, S. Local Rademacher complexities. *The Annals of Statistics*, pages 1497–1537, 2005. (pp. 25 and 173.)

Bartlett, P. L., Jordan, M. I., and McAuliffe, J. D. Convexity, classification, and risk bounds. *JASA*, 101(473):138–156, 2006. (pp. 170.)

Băzăvan, E. G., Li, F., and Sminchisescu, C. Fourier kernel learning. In *ECCV*, pages 459–473. Springer, 2012. (pp. 47.)

Bedford, T. and Cooke, R. M. Probability density decomposition for conditionally dependent random variables modeled by vines. *The Annals of Mathematics and Artificial intelligence*, 32(1-4):245–268, 2001. (pp. 95 and 100.)

Bedford, T. and Cooke, R. M. Vines: A new graphical model for dependent random variables. *The Annals of Statistics*, pages 1031–1068, 2002. (pp. 95 and 100.)

Beebee, H., Hitchcock, C., and Menzies, P. *The Oxford handbook of causation*. Oxford Handbooks Online, 2009. (pp. 143.)

Bell, R. M., Koren, Y., and Volinsky, C. The BellKor solution to the Netflix prize, 2008. (pp. 58.)

Bellman, R. Dynamic programming and lagrange multipliers. *PNAS*, 42(10): 767, 1956. (pp. 60.)

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010. (pp. 103.)

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *ICML*, 2009. (pp. 203 and 209.)

Bengio, Y., Goodfellow, I. J., and Courville, A. Deep learning. Book in preparation for MIT Press, 2015. URL `http://www.iro.umontreal.ca/~bengioy/dlbook`. (pp. 36, 49, 51, 52, 55, 56, 59, and 214.)

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *JMLR*, 13(1):281–305, 2012. (pp. 58.)

Berlinet, A. and Thomas-Agnan, C. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011. (pp. 38.)

Bishop, C. M. *Pattern recognition and machine learning*. Springer, 2006. (pp. 41 and 91.)

Blankertz, B. BCI Competition III data, experiment 4a, subject 3, 1000Hz, 2005. URL `http://bbci.de/competition/iii/download/`. (pp. 179.)

Bottou, L. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, pages 177–186. Springer, 2010. (pp. 34.)

Bottou, L. From machine learning to machine reasoning. *Machine Learning*, 94(2):133–149, 2014. (pp. 2 and 203.)

Boucheron, S., Lugosi, G., and Massart, P. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford University Press, 2013. (pp. 19.)

Boucheron, S., Bousquet, O., and Lugosi, G. Theory of classification: A survey of some recent advances. *ESAIM: Probability and Statistics*, 9: 323–375, 2005. (pp. 24, 25, 170, and 171.)

Bousquet, O. and Bottou, L. The tradeoffs of large scale learning. In *NIPS*, pages 161–168, 2008. (pp. 214.)

Bousquet, O., Boucheron, S., and Lugosi, G. Introduction to statistical learning theory. In *Advanced Lectures on Machine Learning*, pages 169–207. Springer, 2004. (pp. 26.)

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge University Press, 2004. (pp. 32.)

Breiman, L. Random forests. *Machine learning*, 45(1):5–32, 2001. (pp. 59.)

Breiman, L. and Friedman, J. H. Estimating optimal transformations for multiple regression and correlation. *JASA*, 80(391):580–598, 1985. (pp. 119.)

Bubeck, S. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015. (pp. 32 and 35.)

Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. Model compression. In *KDD*, pages 535–541. ACM, 2006. (pp. 6 and 203.)

Cao, B., Pan, S. J., Zhang, Y., Yeung, D.-Y., and Yang, Q. Adaptive transfer learning. In *AAAI*, 2010. (pp. 105.)

Chaudhuri, K., Kakade, S. M., Livescu, K., and Sridharan, K. Multi-view clustering via canonical correlation analysis. In *ICML*, pages 129–136. ACM, 2009. (pp. 113.)

Chen, S. S. and Gopinath, R. A. Gaussianization. In *NIPS*, pages 423–429, 2001. (pp. 72 and 77.)

Cherubini, U., Luciano, E., and Vecchiato, W. *Copula methods in finance*. John Wiley & Sons, 2004. (pp. 77.)

Cho, Y. and Saul, L. K. Analysis and extension of arc-cosine kernels for large margin classification. *arXiv preprint arXiv:1112.3712*, 2011. (pp. 41.)

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. The loss surfaces of multilayer networks. In *AISTATS*, 2015. (pp. 216.)

Cunningham, J. P. and Ghahramani, Z. Linear dimensionality reduction: Survey, insights, and generalizations. *To appear in JMLR*, 2015. (pp. 109.)

Cuturi, M., Fukumizu, K., and Vert, J.-P. Semigroup kernels on measures. In *JMLR*, pages 1169–1198, 2005. (pp. 166.)

Daniusis, P., Janzing, D., Mooij, J., Zscheischler, J., Steudel, B., Zhang, K., and Schölkopf, B. Inferring deterministic causal relations. In *UAI*, 2010. (pp. 160.)

Daumé III, H. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009. (pp. 105.)

Daumé III, H., Kumar, A., and Saha, A. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 53–59. Association for Computational Linguistics, 2010. (pp. 105.)

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *NIPS*, pages 2933–2941, 2014. (pp. 216.)

Dawid, A. P. Causal inference without counterfactuals. *JASA*, 95(450): 407–424, 2000. (pp. 142.)

Dawid, A. P. Beware of the dag! In *NIPS Causality: Objectives and Assessment*, volume 6, pages 59–86, 2010. (pp. 148, 149, and 152.)

De Bie, T., Cristianini, N., and Rosipal, R. Eigenproblems in pattern recognition. In *Handbook of Geometric Computing*, pages 129–167. Springer, 2005. (pp. 115.)

Demarta, S. and McNeil, A. J. The t copula and related copulas. *International Statistical Review*, 73(1):111–129, 2005. (pp. 84.)

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, pages 1–38, 1977. (pp. 75.)

Denil, M., Shakibi, B., Dinh, L., de Freitas, N., et al. Predicting parameters in deep learning. In *NIPS*, pages 2148–2156, 2013. (pp. 216.)

Dissmann, J., Brechmann, E. C., Czado, C., and Kurowicka, D. Selecting and estimating regular vine copulae and application to financial returns. *Computational Statistics & Data Analysis*, 59:52–69, 2013. (pp. 81, 95, 97, and 100.)

Drineas, P. and Mahoney, M. W. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *JMLR*, 6:2153–2175, 2005. (pp. 42.)

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011. (pp. 34.)

Duvenaud, D., Lloyd, J. R., Grosse, R., Tenenbaum, J. B., and Ghahramani, Z. Structure discovery in nonparametric regression through compositional kernel search. In *ICML*, pages 1166–1174, 2013. (pp. 41.)

Efron, B. Bootstrap methods: another look at the jackknife. *The Annals of Statistics*, pages 1–26, 1979. (pp. 62.)

Elidan, G. Copula Bayesian networks. In *NIPS*, pages 559–567, 2010. (pp. 78.)

Elidan, G. Copulas in machine learning. In *Copulae in Mathematical and Quantitative Finance*, pages 39–60. Springer, 2013. (pp. 78.)

Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes challenge: A retrospective. *IJCV*, 111(1):98–136, 2012. (pp. 187.)

Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. Do we need hundreds of classifiers to solve real world classification problems? *JMLR*, 15(1):3133–3181, 2014. (pp. 59.)

Feyereisl, J. and Aickelin, U. Privileged information for data clustering. *Information Sciences*, 194:4–23, 2012. (pp. 206.)

Fouad, S., Tino, P., Raychaudhury, S., and Schneider, P. Incorporating privileged information through metric learning. *Neural Networks and Learning Systems*, 24(7):1086–1098, 2013. (pp. 206.)

Friedman, J. H. Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, pages 1189–1232, 2001. (pp. 58.)

Fujimaki, R., Sogawa, Y., and Morinaga, S. Online heterogeneous mixture modeling with marginal and copula selection. In *KDD*, pages 645–653. ACM, 2011. (pp. 78.)

Fukumizu, K., Bach, F. R., and Gretton, A. Statistical consistency of kernel canonical correlation analysis. *JMLR*, 8:361–383, 2007. (pp. 133.)

Gebelein, H. Das statistische problem der korrelation als variations-und eigenwertproblem und sein zusammenhang mit der ausgleichsrechnung. *Zeitschrift für Angewandte Mathematik und Mechanik*, 21(6):364–379, 1941. (pp. 116.)

Georges, P., Lamy, A.-G., Nicolas, E., Quibel, G., and Roncalli, T. Multivariate survival modelling: a unified approach with copulas. *SSRN 1032559*, 2001. (pp. 77.)

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010. (pp. 56.)

Golub, G. H. and Van Loan, C. F. *Matrix computations*, volume 3. JHU Press, 2012. (pp. 13.)

Gönen, M. and Alpaydın, E. Multiple kernel learning algorithms. *JMLR*, 12: 2211–2268, 2011. (pp. 47.)

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014. (pp. 67 and 74.)

Goodman, N. D., Ullman, T. D., and Tenenbaum, J. B. Learning a theory of causality. *Psychological Review*, 118(1):110, 2011. (pp. 169.)

Granger, C. W. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, pages 424–438, 1969. (pp. 161.)

Gretton, A., Herbrich, R., Smola, A., Bousquet, O., and Schölkopf, B. Kernel methods for measuring independence. *JMLR*, 2005a. (pp. 117.)

Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. Measuring statistical dependence with Hilbert-Schmidt norms. In *ALT*, pages 63–77. Springer, 2005b. (pp. 117 and 128.)

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *JMLR*, 13(1):723–773, 2012a. (pp. 105 and 123.)

Gretton, A., Sejdinovic, D., Strathmann, H., Balakrishnan, S., Pontil, M., Fukumizu, K., and Sriperumbudur, B. K. Optimal kernel choice for large-scale two-sample tests. In *NIPS*, pages 1205–1213, 2012b. (pp. 122.)

Gross, S. ResNet training in Torch, 2016. URL `https://github.com/facebook/fb.resnet.torch`. (pp. 188.)

Guyon, I. Cause-effect pairs kaggle competition, 2013. URL `https://www.kaggle.com/c/cause-effect-pairs/`. (pp. 165 and 166.)

Guyon, I. Chalearn fast causation coefficient challenge, 2014. URL `https://www.codalab.org/competitions/1381`. (pp. 165, 179, and 180.)

Haff, I. H., Aas, K., and Frigessi, A. On the simplified pair-copula construction: simply useful or too simplistic? *Journal of Multivariate Analysis*, 101(5):1296–1310, 2010. (pp. 95 and 100.)

Hardoon, D. R., Szedmak, S., and Shawe-Taylor, J. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664, 2004. (pp. 113.)

Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015. (pp. 58 and 214.)

Heckerman, D., Meek, C., and Cooper, G. A Bayesian approach to causal discovery. Technical report, Microsoft Research, 1997. (pp. 157.)

Hein, M. and Bousquet, O. Hilbertian metrics and positive definite kernels on probability measures. In *AISTATS*, 2005. (pp. 166.)

Hernández-Lobato, D., Sharmanska, V., Kersting, K., Lampert, C. H., and Quadrianto, N. Mind the nuisance: Gaussian process classification using privileged noise. In *NIPS*, pages 837–845, 2014. (pp. 206.)

Hernández-Lobato, D., Morales-Mombiela, P., Lopez-Paz, D., and Suárez, A. Non-linear Causal Inference using Gaussianity Measures. *JMLR*, 2016. (pp. 4, 7, and 158.)

Hernández-Lobato, J. M., Lloyd, J. R., and Hernández-Lobato, D. Gaussian process conditional copulas with applications to financial time series. In *NIPS*, pages 1736–1744, 2013. (pp. 78 and 101.)

Hinton, G., McClelland, J., and Rumelhart, D. Distributed representations. In *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1*, pages 77–109. MIT Press, 1986. (pp. 51.)

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv*, 2015. (pp. 6, 203, 204, and 208.)

Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. (pp. 136.)

Hoeffding, W. Scale-invariant correlation theory. In *The collected works of Wassily Hoeffding*, pages 57–107. Springer, 1994. (pp. 77.)

Horn, R. A. and Johnson, C. R. *Matrix analysis*. Cambridge University Press, 2012. (pp. 12 and 13.)

Hotelling, H. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):498–520, 1933. (pp. 110.)

Hotelling, H. Relations between two sets of variates. *Biometrika*, pages 321–377, 1936. (pp. 108, 112, and 113.)

Hoyer, P. O., Janzing, D., Mooij, J. M., Peters, J., and Schölkopf, B. Nonlinear causal discovery with additive noise models. In *NIPS*, pages 689–696, 2009. (pp. 158.)

Huang, J., Gretton, A., Borgwardt, K. M., Schölkopf, B., and Smola, A. J. Correcting sample selection bias by unlabeled data. In *NIPS*, pages 601–608, 2006. (pp. 105.)

Huang, P.-S., Avron, H., Sainath, T. N., Sindhwani, V., and Ramabhadran, B. Kernel methods match deep neural networks on timit. In *ICASSP*, pages 205–209. IEEE, 2014. (pp. 47.)

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. (pp. 58 and 187.)

Janzing, D. and Schölkopf, B. Causal inference using the algorithmic Markov condition. *Information Theory, IEEE Transactions on*, 56(10):5168–5194, 2010. (pp. 155.)

Janzing, D., Mooij, J., Zhang, K., Lemeire, J., Zscheischler, J., Daniušis, P., Steudel, B., and Schölkopf, B. Information-geometric approach to inferring causal directions. *Artificial Intelligence*, 182:1–31, 2012. (pp. 160 and 178.)

Jaworski, P., Durante, F., Härdle, W., and Rychlik, T. *Copula Theory and its Applications*. Lecture Notes in Statistics. Springer Berlin Heidelberg, 2010. (pp. 77.)

Jebara, T., Kondor, R., and Howard, A. Probability product kernels. *JMLR*, 5:819–844, 2004. (pp. 166.)

Joe, H. Families of m-variate distributions with given margins and $m(m-1)/2$ bivariate dependence parameters. *Lecture Notes-Monograph Series*, pages 120–141, 1996. (pp. 95.)

Joe, H. *Multivariate models and multivariate dependence concepts*. CRC Press, 1997. (pp. 77.)

Jolliffe, I. *Principal component analysis*. Wiley Online Library, 2002. (pp. 110.)

Jordan, M. I. *Learning in Graphical Models*, volume 89. Springer Science & Business Media, 1998. (pp. 72.)

Kakade, S. M. and Foster, D. P. Multi-view regression via canonical correlation analysis. In *Learning Theory*, pages 82–96. Springer, 2007. (pp. 113.)

Kanamori, T., Suzuki, T., and Sugiyama, M. Statistical analysis of kernel-based least-squares density-ratio estimation. *Machine Learning*, 86(3): 335–367, 2012. (pp. 105.)

Karpathy, A. Convolutional neural networks for visual recognition, 2015. URL `http://cs231n.github.io/`. (pp. 54.)

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013. (pp. 73 and 137.)

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. Semi-supervised learning with deep generative models. In *NIPS*, pages 3581–3589, 2014. (pp. 73.)

Kirshner, S. Learning with tree-averaged densities and distributions. In *NIPS*, 2007. (pp. 77.)

Kirshner, S. and Póczos, B. ICA and ISA using Schweizer-Wolff measure of dependence. In *ICML*, pages 464–471. ACM, 2008. (pp. 77.)

Kleiner, A., Talwalkar, A., Sarkar, P., and Jordan, M. I. A scalable bootstrap for massive data. *Journal of the Royal Statistical Society*, 76(4):795–816, 2014. (pp. 62.)

Koltchinskii, V. Rademacher penalties and structural risk minimization. *Information Theory, IEEE Transactions on*, 47(5):1902–1914, 2001. (pp. 23.)

Koltchinskii, V. *Oracle Inequalities in Empirical Risk Minimization and Sparse Recovery Problems*, volume 38 of *Ecole de Probabilités de Saint-Flour*. Springer Science & Business Media, 2011. (pp. 24.)

Koltchinskii, V. and Panchenko, D. Rademacher processes and bounding the risk of function learning. In *High dimensional probability II*, pages 443–457. Springer, 2000. (pp. 171.)

Kpotufe, S., Sgouritsa, E., Janzing, D., and Schölkopf, B. Consistency of causal inference under the additive noise model. In *ICML*, pages 478–486, 2014. (pp. 159.)

Kramer, M. A. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991. (pp. 136.)

Krizhevsky, A. The CIFAR-10 and CIFAR-100 datasets, 2009. URL `http://www.cs.toronto.edu/~kriz/cifar.html`. (pp. 212.)

Kumar, S., Mohri, M., and Talwalkar, A. Sampling methods for the Nyström method. *JMLR*, 13(1):981–1006, 2012. (pp. 42.)

Kurowicka, D. *Dependence modeling: vine copula handbook*. World Scientific, 2011. (pp. 95, 97, and 100.)

Lacerda, G., Spirtes, P. L., Ramsey, J., and Hoyer, P. O. Discovering cyclic causal models by independent components analysis. *arXiv preprint arXiv:1206.3273*, 2012. (pp. 159.)

Lai, P. L. and Fyfe, C. Kernel and nonlinear canonical correlation analysis. *International Journal of Neural Systems*, 10(05):365–377, 2000. (pp. 108 and 113.)

Laparra, V., Camps-Valls, G., and Malo, J. Iterative Gaussianization: from ICA to random rotations. *Neural Networks, IEEE Transactions on*, 22(4): 537–549, 2011. (pp. 72.)

Lapin, M., Hein, M., and Schiele, B. Learning using privileged information: SVM+ and weighted SVM. *Neural Networks*, 53:95–108, 2014. (pp. 206.)

Le, Q., Sarlos, T., and Smola, A. Fastfood: computing Hilbert space expansions in loglinear time. In *ICML*, pages 244–252, 2013. (pp. 43 and 46.)

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998a. (pp. 204.)

LeCun, Y., Cortes, C., and Burges, C. J. The MNIST database of handwritten digits, 1998b. URL `http://yann.lecun.com/exdb/mnist/`. (pp. 125 and 212.)

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553): 436–444, 2015. (pp. 51.)

Ledoux, M. and Talagrand, M. *Probability in Banach Spaces: isoperimetry and processes*, volume 23. Springer Science & Business Media, 2013. (pp. 175.)

Lemeire, J. and Dirkx, E. Causal models as minimal descriptions of multivariate systems, 2006. (pp. 155.)

Lewis, D. *Counterfactuals*. John Wiley & Sons, 1974. (pp. 141.)

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft COCO: Common objects in context. In *Computer Vision–ECCV 2014*, pages 740–755. Springer, 2014. (pp. 187.)

Liu, H., Lafferty, J., and Wasserman, L. The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *JMLR*, 10:2295–2328, 2009. (pp. 78 and 83.)

Lopez-Paz, D., Hernández-Lobato, J. M., and Schölkopf, B. Semi-supervised domain adaptation with non-parametric copulas. In *NIPS*, pages 674–682, 2012. (pp. 3, 4, 7, 65, 93, 96, and 105.)

Lopez-Paz, D., Hennig, P., and Schölkopf, B. The randomized dependence coefficient. In *NIPS*, pages 1–9, 2013a. (pp. 4, 5, 7, 77, 79, 107, and 117.)

Lopez-Paz, D., Hernández-Lobato, J. M., and Ghahramani, Z. Gaussian process vine copulas for multivariate dependence. In *ICML*, pages 10–18, 2013b. (pp. 3, 5, 7, 65, 87, 95, 96, and 102.)

Lopez-Paz, D., Sra, S., Smola, A. J., Ghahramani, Z., and Schölkopf, B. Randomized nonlinear component analysis. In *ICML*, pages 1359–1367, 2014. (pp. 4, 5, 7, 107, 110, 113, and 206.)

Lopez-Paz, D., Muandet, K., Schölkopf, B., and Tolstikhin, I. Towards a learning theory of cause-effect inference. In *ICML*, pages 1452–1461, 2015. (pp. 4, 5, 7, 162, and 163.)

Lopez-Paz, D., Bottou, L., Schölkopf, B., and Vapnik, V. Unifying distillation and privileged information. *ICLR*, 2016a. (pp. 4, 5, 7, and 201.)

Lopez-Paz, D., Muandet, K., and Recht, B. The randomized causation coefficient. *JMLR*, 2016b. (pp. 4, 5, 7, and 162.)

Lopez-Paz, D., Nishihara, R., Chintala, S., Schölkopf, B., and Bottou, L. Discovering causal signals in images. *Under review*, 2016c. (pp. 4, 5, 7, and 163.)

Ma, J. and Sun, Z. Copula component analysis. In *Independent Component Analysis and Signal Separation*, pages 73–80. Springer, 2007. (pp. 77.)

Mardia, K. V., Kent, J. T., and Bibby, J. M. *Multivariate analysis*. Academic Press, 1979. (pp. 121.)

Martins, A. F., Smith, N. A., Xing, E. P., Aguiar, P. M., and Figueiredo, M. A. Nonextensive information theoretic kernels on measures. *JMLR*, 10: 935–975, 2009. (pp. 166.)

Massart, P. The tight constant in the Dvoretzky-Kiefer-Wolfowitz inequality. *The Annals of Probability*, pages 1269–1283, 1990. (pp. 16.)

Massart, P. Some applications of concentration inequalities to statistics. *Annales de la Faculté des sciences de Toulouse: Mathématiques*, 9(2): 245–303, 2000. (pp. 25.)

Maurer, A. The Rademacher complexity of linear transformation classes. In *Learning Theory*, pages 65–78. Springer, 2006. (pp. 173.)

McWilliams, B., Balduzzi, D., and Buhmann, J. M. Correlated random features for fast semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 440–448, 2013. (pp. 109.)

Mercer, J. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions of the royal society of London*, pages 415–446, 1909. (pp. 43.)

Messerli, F. H. Chocolate consumption, cognitive function, and nobel laureates. *New England Journal of Medicine*, 367(16):1562–1564, 2012. (pp. 1 and 139.)

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. In *ICLR Workshops*, 2013. (pp. 203.)

Minka, T. P. Expectation propagation for approximate Bayesian inference. In *UAI*, pages 362–369. Morgan Kaufmann Publishers Inc., 2001. (pp. 88 and 89.)

Mobahi, H. and Fisher III, J. W. A theoretical analysis of optimization by Gaussian continuation. In *AAAI*, 2015. (pp. 216.)

Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of machine learning*. MIT Press, 2012. (pp. 21.)

Mooij, J. M., Janzing, D., Heskes, T., and Schölkopf, B. On causal discovery with cyclic additive noise models. In *NIPS*, pages 639–647, 2011. (pp. 153 and 159.)

Mooij, J. M., Peters, J., Janzing, D., Zscheischler, J., and Schölkopf, B. Distinguishing cause from effect using observational data: methods and benchmarks. *JMLR*, 2014. (pp. 178 and 187.)

Muandet, K. *From Points to Probability Measures: A Statistical Learning on Distributions with Kernel Mean Embedding*. PhD thesis, University of Tübingen, Germany, September 2015. (pp. 166.)

Muandet, K., Fukumizu, K., Dinuzzo, F., and Schölkopf, B. Learning from distributions via support measure machines. In *NIPS*, pages 10–18, 2012. (pp. 166.)

Mumford, S. and Anjum, R. L. *Causation: A Very Short Introduction*. Oxford University Press, 2013. (pp. 144.)

Murphy, K. P. *Machine learning: a probabilistic perspective*. MIT Press, 2012. (pp. 21.)

Nelsen, R. B. *An introduction to copulas*, volume 139. Springer, 2006. (pp. 77, 78, 79, 83, 84, and 85.)

Nesterov, Y. *Introductory lectures on convex optimization: a basic course*. Applied optimization. Kluwer Academic Publ., 2004. (pp. 32, 34, and 35.)

Nishihara, R., Lopez-Paz, D., and Bottou, L. No regret bound for extreme bandits. *AISTATS*, 2016. (pp. 7 and 58.)

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, pages 1717–1724. IEEE, 2014. (pp. 203.)

Panagiotelis, A., Czado, C., and Joe, H. Pair copula constructions for multivariate discrete data. *Journal of the American Statistical Association*, 107(499):1063–1072, 2012. (pp. 95.)

Parzen, E. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, pages 1065–1076, 1962. (pp. 75.)

Patton, A. J. Modelling asymmetric exchange rate dependence. *International Economic Review*, 47(2):527–556, 2006. (pp. 77 and 85.)

Patton, A. J. *Applications of copula theory in financial econometrics*. PhD thesis, University of California, San Diego, 2002. (pp. 87.)

Pearl, J. *Bayesian networks: A model of self-activated memory for evidential reasoning*. University of California (Los Angeles). Computer Science Department, 1985. (pp. 94.)

Pearl, J. Causal inference in statistics: An overview. *Statistics Surveys*, 3: 96–146, 2009a. (pp. 142.)

Pearl, J. *Causality*. Cambridge University Press, 2009b. (pp. 144, 148, 149, 150, 174, 182, and 183.)

Pearson, K. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. (pp. 108 and 110.)

Pechyony, D. and Vapnik, V. On the theory of learning with privileged information. In *NIPS*, pages 1894–1902, 2010. (pp. 206.)

Peters, J. Causality. Technical report, ETH Zurich, 2015. (pp. 2 and 151.)

Peters, J., Janzing, D., Gretton, A., and Schölkopf, B. Detecting the direction of causal time series. In *ICML*, pages 801–808. ACM, 2009. (pp. 179.)

Peters, J., Janzing, D., and Schölkopf, B. Causal inference on discrete data using additive noise models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(12):2436–2450, 2011. (pp. 159.)

Peters, J., Mooij, J. M., Janzing, D., and Schölkopf, B. Causal discovery with continuous additive noise models. *JMLR*, 15(1):2009–2053, 2014. (pp. 158 and 183.)

Peters, J. M. *Restricted structural equation models for causal inference*. PhD thesis, ETH Zürich, 2012. (pp. 144, 148, 150, 157, and 161.)

Petersen, K. B. and Pedersen, M. S. The matrix cookbook, 2012. (pp. 13.)

Pickup, L. C., Pan, Z., Wei, D., Shih, Y., Zhang, C., Zisserman, A., Schölkopf, B., and Freeman, W. T. Seeing the arrow of time. In *CVPR*, 2014. (pp. 190.)

Plataniotis, K. Gaussian mixtures and their applications to signal processing. *Advanced Signal Processing Handbook*, 2000. (pp. 76.)

Póczos, B., Rinaldo, A., Singh, A., and Wasserman, L. Distribution-free distribution regression. *AISTATS*, 2013. (pp. 166.)

Póczos, B., Ghahramani, Z., and Schneider, J. G. Copula-based kernel dependency measures. In *ICML*, 2012. (pp. 77, 79, and 81.)

Quinn, G. E., Shin, C. H., Maguire, M. G., and Stone, R. A. Myopia and ambient lighting at night. *Nature*, 399(6732):113–114, 1999. (pp. 140.)

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *NIPS*, 2007. (pp. 43 and 44.)

Rahimi, A. and Recht, B. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *NIPS*, 2008. (pp. 43, 44, 175, and 195.)

Rank, J. *Copulas: From theory to application in finance*. Risk books, 2007. (pp. 77 and 93.)

Rao, B. R. Partial canonical correlations. *Trabajos de estadística y de investigación operativa*, 20(2):211–219, 1969. (pp. 120.)

Rasmus, A., Valpola, H., Honkala, M., Berglund, M., and Raiko, T. Semi-supervised learning with ladder network. In *NIPS*, 2015. (pp. 217.)

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. MIT Press, 2006. (pp. 30, 61, and 87.)

Reed, M. and Simon, B. Functional analysis, volume 1 of methods of modern mathematical physics, 1972. (pp. 13 and 191.)

Reichenbach, H. *The direction of time*. Dover, 1956. (pp. 1, 143, and 183.)

Rényi, A. On measures of dependence. *Acta Mathematica Hungarica*, 10 (3-4):441–451, 1959. (pp. 79 and 116.)

Reshef, D. N., Reshef, Y. A., Finucane, H. K., Grossman, S. R., McVean, G., Turnbaugh, P. J., Lander, E. S., Mitzenmacher, M., and Sabeti, P. C. Detecting novel associations in large data sets. *Science*, 334(6062):1518–1524, 2011. (pp. 128.)

Rey, M. and Roth, V. Copula mixture model for dependency-seeking clustering. In *ICML*, 2012. (pp. 78.)

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014. (pp. 73.)

Ribeiro, B., Silva, C., Vieira, A., Gaspar-Cunha, A., and das Neves, J. C. Financial distress model prediction using SVM+. In *IJCNN*. IEEE, 2010. (pp. 206.)

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML*, pages 833–840, 2011. (pp. 137.)

Rifkin, R. M. and Lippert, R. A. Notes on regularized least squares. Technical report, MIT, 2007. (pp. 30.)

Rosenbaum, P. R. and Rubin, D. B. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983. (pp. 154.)

Rosenblatt, M. Remarks on a multivariate transformation. *The Annals of Mathematical Statistics*, pages 470–472, 1952. (pp. 79.)

Roweis, S. Gaussian identities. *University of Toronto*, 1999. (pp. 71.)

Roweis, S. and Brody, C. Linear heteroencoders. Technical report, Gatsby Computational Neuroscience Unit, 1999. (pp. 137.)

Rudin, W. *Fourier Analysis on Groups*. Wiley, 1962. (pp. 44.)

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Cognitive Modeling*, 5:3, 1986. (pp. 56.)

Salvadori, G., De Michele, C., Kottegoda, N. T., and Rosso, R. *Extremes in nature: an approach using copulas*, volume 56. Springer Science & Business Media, 2007. (pp. 77, 82, and 85.)

Schepsmeier, U. and Stöber, J. Derivatives and fisher information of bivariate copulas. *Statistical Papers*, 55(2):525–542, 2014. (pp. 83.)

Schölkopf, B. and Smola, A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001. (pp. 37, 38, and 166.)

Schölkopf, B., Smola, A., and Müller, K.-R. Kernel principal component analysis. In *ICANN*, pages 583–588. Springer, 1997. (pp. 108 and 110.)

Schölkopf, B., Janzing, D., Peters, J., Sgouritsa, E., Zhang, K., and Mooij, J. M. On causal and anticausal learning. In *ICML*, pages 1255–1262, July 2012. (pp. 155, 162, 182, 183, and 209.)

Schweizer, B. and Sklar, A. *Probabilistic metric spaces*. Courier Corporation, 1983. (pp. 77 and 84.)

Schweizer, B. and Wolff, E. F. On nonparametric measures of dependence for random variables. *The Annals of Statistics*, pages 879–885, 1981. (pp. 77 and 79.)

Seeger, M. Expectation propagation for exponential families. Technical report, EPFL Report 161464, 2005. (pp. 89 and 91.)

Shalev-Shwartz, S. and Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. (pp. 21 and 25.)

Sharmanska, V., Quadrianto, N., and Lampert, C. H. Learning to rank using privileged information. In *ICCV*, pages 825–832. IEEE, 2013. (pp. 206 and 212.)

Sharmanska, V., Quadrianto, N., and Lampert, C. H. Learning to transfer privileged information. *arXiv*, 2014. (pp. 206.)

Shimizu, S., Hoyer, P. O., Hyvärinen, A., and Kerminen, A. A linear non-Gaussian acyclic model for causal discovery. *JMLR*, 7:2003–2030, 2006. (pp. 157, 158, and 159.)

Sklar, A. *Fonctions de répartition à n dimensions et leurs marges*. Université Paris 8, 1959. (pp. 77.)

Smola, A., Gretton, A., Song, L., and Schölkopf, B. A Hilbert space embedding for distributions. In *ALT*, pages 13–31. Springer, 2007. (pp. 166 and 185.)

Snelson, E. and Ghahramani, Z. Sparse Gaussian processes using pseudo-inputs. In *NIPS*, pages 1257–1264, 2005. (pp. 92.)

Sober, E. Venetian sea levels, british bread prices, and the principle of the common cause. *The British Journal for the Philosophy of Science*, 52(2): 331–346, 2001. (pp. 140.)

Song, L. *Learning via Hilbert Space Embedding of Distributions*. PhD thesis, The University of Sydney, 2008. (pp. 167 and 192.)

Song, L., Smola, A., Gretton, A., Bedo, J., and Borgwardt, K. Feature selection via dependence maximization. *JMLR*, 13(1):1393–1434, 2012. (pp. 130.)

Souza, C. R. Kernel functions for machine learning applications, 2010. URL `http://crsouza.com/2010/03/kernel-functions-for-machine-learning-applications/`. (pp. 40.)

Spirtes, P., Glymour, C. N., and Scheines, R. *Causation, prediction, and search*, volume 81. MIT Press, 2000. (pp. 156.)

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014. (pp. 55.)

Sriperumbudur, B. K. and Szabó, Z. Optimal rates for random Fourier features. In *NIPS*, 2015. (pp. 46 and 175.)

Sriperumbudur, B. K., Gretton, A., Fukumizu, K., Schölkopf, B., and Lanckriet, G. R. Hilbert space embeddings and metrics on probability measures. *JMLR*, 11:1517–1561, 2010. (pp. 166.)

Sriperumbudur, B. K., Fukumizu, K., and Lanckriet, G. R. G. Universality, characteristic kernels and rkhs embedding of measures. *The Journal of Machine Learning Research*, 12:2389–2410, 2011. (pp. 117.)

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014. (pp. 58 and 187.)

Stegle, O., Janzing, D., Zhang, K., Mooij, J. M., and Schölkopf, B. Probabilistic latent variable models for distinguishing between cause and effect. In *NIPS*, pages 1687–1695, 2010. (pp. 159.)

Steinwart, I. and Christmann, A. *Support Vector Machines*. Springer Science & Business Media, 2008. (pp. 171 and 191.)

Sugiyama, M., Suzuki, T., and Kanamori, T. *Density ratio estimation in machine learning*. Cambridge University Press, 2012. (pp. 121.)

Sutskever, I. *Training recurrent neural networks*. PhD thesis, University of Toronto, 2013. (pp. 55.)

Szabó, Z., Gretton, A., Póczos, B., and Sriperumbudur, B. Two-stage sampled learning theory on distributions. *arXiv preprint arXiv:1402.1754*, 2014. (pp. 166 and 191.)

Székely, G. J., Rizzo, M. L., Bakirov, N. K., et al. Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, 35(6): 2769–2794, 2007. (pp. 130.)

Tewari, A., Giering, M. J., and Raghunathan, A. Parametric characterization of multimodal distributions with non-Gaussian modes. In *ICDMW*, pages 286–292. IEEE, 2011. (pp. 78.)

Theis, L., Oord, A. v. d., and Bethge, M. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015. (pp. 67.)

Tieleman, T. and Hinton, G. Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4, 2012. (pp. 34 and 187.)

Tolstikhin, I., Sriperumbudur, B., and Muandet, K. Minimax Estimation of Kernel Mean Embeddings. *ArXiv e-prints*, 2016. (pp. 174.)

Tolstikhin, I. and Lopez-Paz, D. Lower bounds for realizable transductive learning. *arXiv*, 2016. (pp. 7.)

Trivedi, P. K. and Zimmer, D. M. *Copula modeling: an introduction for practitioners*. Now Publishers Inc, 2007. (pp. 77 and 78.)

Tropp, J. A. An introduction to matrix concentration inequalities. *Foundations and Trends in Machine Learning*, 8(1-2):1–230, 2015. (pp. 21 and 111.)

Vallentin, M. The probability and statistics cookbook, 2015. URL `http://statistics.zone/`. (pp. 15.)

Van Gerven, M. A., Cseke, B., De Lange, F. P., and Heskes, T. Efficient Bayesian multivariate fMRI analysis using a sparsifying spatio-temporal prior. *NeuroImage*, 50(1):150–161, 2010. (pp. 89 and 90.)

Vapnik, V. *Estimation of dependences based on empirical data*, volume 40. Springer-verlag New York, 1982. (pp. 1.)

Vapnik, V. *Statistical learning theory*. Wiley New York, 1998. (pp. 27, 66, and 202.)

Vapnik, V. and Chervonenkis, A. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & its Applications*, 16(2):264–280, 1971. (pp. 17.)

Vapnik, V. and Izmailov, R. Learning using privileged information: Similarity control and knowledge transfer. *JMLR*, 16:2023–2049, 2015. (pp. 202, 205, and 208.)

Vapnik, V. and Vashist, A. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5):544–557, 2009. (pp. 6, 113, 127, 128, 202, 205, and 210.)

Verma, T. and Pearl, J. Equivalence and synthesis of causal models. In *UAI*, pages 255–270, 1991. (pp. 147.)

Villani, C. *Topics in optimal transportation*. American Mathematical Soc., 2003. (pp. 180.)

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103. ACM, 2008. (pp. 137.)

Vinokourov, A., Cristianini, N., and Shawe-taylor, J. Inferring a semantic representation of text via cross-language correlation analysis. *NIPS*, 2002. (pp. 112.)

Von Luxburg, U. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007. (pp. 111.)

Wasserman, L. *All of Statistics*. Springer, 2010. (pp. 75.)

Weston, J., Collobert, R., Sinz, F., Bottou, L., and Vapnik, V. Inference with the Universum. In *ICML*, pages 1009–1016. ACM, 2006. (pp. 203 and 209.)

Williams, C. and Seeger, M. Using the Nyström method to speed up kernel machines. In *NIPS*, pages 682–688, 2001. (pp. 42.)

Wilson, A. and Ghahramani, Z. Copula processes. In *NIPS*, pages 2460–2468, 2010. (pp. 78.)

Wilson, A. G. *Covariance kernels for fast automatic pattern discovery and extrapolation with Gaussian processes*. PhD thesis, University of Cambridge, 2014. (pp. 47.)

Wolpert, D. H. and Macready, W. G. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997. (pp. 26 and 60.)

Wright, S. Correlation and Causation. *J. Agric. Res.*, 20:557–585, 1921. (pp. 148.)

Yang, J., Sindhwani, V., Fan, Q., Avron, H., and Mahoney, M. Random Laplace feature maps for semigroup kernels on histograms. In *CVPR*, pages 971–978. IEEE, 2014. (pp. 46.)

Yang, T., Li, Y.-F., Mahdavi, M., Jin, R., and Zhou, Z.-H. Nyström method vs random Fourier features: A theoretical and empirical comparison. In *NIPS*, pages 476–484, 2012. (pp. 42.)

Yudkowsky, E. Artificial intelligence as a positive and negative factor in global risk. *Global Catastrophic Risks*, 1:303, 2008. (pp. 184.)

Zhang, K. and Hyvärinen, A. On the identifiability of the post-nonlinear causal model. In *UAI*, pages 647–655. AUAI Press, 2009. (pp. 159.)