Andreas Krause

Probabilistic Artificial Intelligence

Fall 2023

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute for Machine Learning
Department of Computer Science

Contents. Bayesian learning and inference. Bayesian linear regression, Gaussian processes, Bayesian Deep Learning. Variational inference, Markov chain Monte Carlo methods. Active learning, Bayesian optimization and the exploration-exploitation dilemma. Markov decision processes, and Reinforcement learning in tabular and large state-action spaces with model-based and model-free approaches.

ACKNOWLEDGEMENT

These notes were written and edited by **Jonas Hübotter**.

- Many figures were inspired by the accompanying Jupyter notebooks created by Sebastian Curi.
- Parts of section 1.1 were adapted from earlier notes by Mohammad Reza Karimi.
- Figure 12.2 was contributed by Hado van Hasselt.
- Some exercises are adapted from previous iterations of the course.
- We thank Zhiyuan Hu and Shyam Sundhar Ramesh for proofreading.

CONTRIBUTING

You are encouraged to raise issues and suggest fixes for anything you think can be improved.
CONTACT: pai-script@lists.inf.ethz.ch
REPOSITORY: `https://gitlab.inf.ethz.ch/OU-KRAUSE/pai-script`

JUPYTER NOTEBOOKS WITH EXAMPLES

`https://gitlab.inf.ethz.ch/OU-KRAUSE/pai-demos`

SYLLABUS

1. *Introduction* chapter 1

Part I: Probabilistic Machine Learning

2. *Bayesian Linear Regression & Filtering* chapter 2 through to section 2.4 and chapter 3
3. *Gaussian Processes 1* rest of chapter 2 and chapter 4 through to section 4.3
4. *Gaussian Processes 2* rest of chapter 4
5. *Variational Inference* chapter 5 up to section 5.5.1
6. *Markov Chain Monte Carlo* section 5.5.1 and chapter 6
7. *Bayesian Deep Learning* chapter 7

Part II: Sequential Decision-Making

8. *Active Learning & Bayesian Optimization* chapter 8 and chapter 9
9. *Markov Decision Processes* chapter 10
10. *Reinforcement Learning 1* chapter 11
11. *Reinforcement Learning 2* chapter 12 through to section 12.3
12. *Reinforcement Learning 3* section 12.4 through to section 12.5
13. *Reinforcement Learning 4* chapter 13

COMPILATION DATE: **December 2, 2023**

This set of notes was written for the course PROBABILISTIC ARTIFICIAL INTELLIGENCE (263-5210-00L) at ETH Zürich.

Contents

Preface

Artificial intelligence commonly refers to the science and engineering of artificial systems that can carry out tasks generally associated with requiring aspects of human intelligence, such as playing games, translating languages, and driving cars. In recent years, there have been exciting advances in learning-based, data-driven approaches towards AI, and machine learning and deep learning have enabled computer systems to perceive the world in unprecedented ways. Reinforcement learning has enabled breakthroughs in complex games such as Go and challenging robotics tasks such as quadrupedal locomotion.

A key aspect of intelligence is to not only make predictions, but reason about the *uncertainty* in these predictions, and to consider this uncertainty when making decisions. This is what the course "Probabilistic Artificial Intelligence" is about. The first part covers probabilistic approaches to machine learning. We discuss how Bayesian approaches to learning allow to differentiate between "epistemic" uncertainty due to lack of data and "aleatoric" uncertainty, which is irreducible and stems, e.g., from noisy observations and outcomes. We discuss concrete approaches towards Bayesian learning, such as Bayesian linear regression, Gaussian process models and Bayesian neural networks. Often, inference and making predictions with such models is intractable, and we discuss modern approaches to efficient approximate inference.

The second part of the course is about taking uncertainty into account in sequential decision tasks. We consider active learning and Bayesian optimization — approaches that collect data by proposing experiments that are informative for reducing the epistemic uncertainty. We then consider reinforcement learning, a rich formalism for modeling agents that learn to act in uncertain environments. After covering the basic formalism of Markov Decision Processes, we consider modern deep RL approaches that use neural network function approximation. We close by discussing modern approaches in model-based RL, which harness epistemic and aleatoric uncertainty to guide exploration, while also reasoning about safety.

Summary of Notation

We follow these general rules:

- uppercase italic for constants N
- lowercase italic for indices i and scalar variables x
- lowercase italic bold for vectors $\boldsymbol{x}$, entries are denoted $\boldsymbol{x}(i)$
- uppercase italic bold for matrices $\boldsymbol{M}$, entries are denoted $\boldsymbol{M}(i, j)$
- uppercase italic for random variables X
- uppercase bold for random vectors $\mathbf{X}$, entries are denoted $\mathbf{X}(i)$
- uppercase italic for sets A
- uppercase calligraphy for spaces (usually infinite sets) $\mathcal{A}$

$\doteq$	equality by definition
$\approx$	approximately equals
$\propto$	proportional to (up to multiplicative constants), $f \propto g$ iff $\exists k.\ \forall x.\ f(x) = k \cdot g(x)$
const	an (additive) constant
$\mathbb{N}$	set of natural numbers $\{1, 2, \dots\}$
$\mathbb{N}_0$	set of natural numbers, including 0, $\mathbb{N} \cup \{0\}$
$\mathbb{R}$	set of real numbers
$[m]$	set of natural numbers from 1 to m, $\{1, 2, \dots, m-1, m\}$
$i : j$	subset of natural numbers between i and j, $\{i, i+1, \dots, j-1, j\}$
$(a, b]$	real interval between a and b including b but not including a
$f : A \to B$	function f from elements of set A to elements of set B
$f \circ g$	function composition, $f(g(\cdot))$
$(\cdot)_+$	$\max\{0, \cdot\}$
log	logarithm with base e
$\mathcal{P}(A)$	power set (set of all subsets) of A
$\mathbb{1}\{\textit{predicate}\}$	indicator function ($\mathbb{1}\{\textit{predicate}\} \doteq 1$ if the *predicate* is true, else 0)
$\odot$	Hadamard (element-wise) product
$\leftarrow$	assignment

$\nabla f(\boldsymbol{x}) \in \mathbb{R}^n$	gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $\boldsymbol{x} \in \mathbb{R}^n$
$\boldsymbol{D}\boldsymbol{g}(\boldsymbol{x}) \in \mathbb{R}^{m \times n}$	Jacobian of a function $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^m$ at a point $\boldsymbol{x} \in \mathbb{R}^n$
$\boldsymbol{H}f(\boldsymbol{x}) \in \mathbb{R}^{n \times n}$	Hessian of a function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $\boldsymbol{x} \in \mathbb{R}^n$
$\nabla \cdot \mathbf{F}$	divergence operation on vector field $\mathbf{F}$
Δf	Laplacian of a scalar field $f : \mathbb{R}^n \to \mathbb{R}$
$f \in \mathcal{O}(g)$	f grows at most as fast as g (up to constant factors), $0 \leq \limsup_{n \to \infty} \left\lvert \frac{f(n)}{g(n)} \right\rvert < \infty$
$f \in \tilde{\mathcal{O}}(g)$	f grows at most as fast as g up to constant and logarithmic factors
$\lVert \cdot \rVert_\alpha$	α-norm
$\lVert \cdot \rVert_{\boldsymbol{A}}$	Mahalanobis norm induced by matrix $\boldsymbol{A}$

LINEAR ALGEBRA

$\boldsymbol{I}$	identity matrix
$\boldsymbol{A}^\top$	transpose of matrix $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	inverse of invertible matrix $\boldsymbol{A}$
$\boldsymbol{A}^{1/2}$	square root of a symmetric and positive semi-definite matrix $\boldsymbol{A}$
$\det(\boldsymbol{A})$	determinant of $\boldsymbol{A}$
$\operatorname{tr}(\boldsymbol{A})$	trace of $\boldsymbol{A}$, $\sum_i \boldsymbol{A}(i,i)$
$\operatorname{diag}_{i \in I}\{a_i\}$	diagonal matrix with elements a_i, indexed according to the set I

PROBABILITY

Ω	sample space
$\mathcal{A}$	event space
$\mathbb{P}$	probability measure
$X \sim P$	random variable X follows the distribution P
$X_{1:n} \overset{\text{iid}}{\sim} P$	random variables $X_{1:n}$ are independent and identically distributed according to distribution P
$x \sim P$	value x is sampled according to distribution P
P_X	cumulative distribution function of a random variable X
$\overline{P}_X$	tail distribution function of a random variable X
P_X^{-1}	quantile function of a random variable X
p_X	probability mass function (if discrete) or probability density function (if continuous) of a random variable X
$\Delta^{\mathcal{A}}$	set of all probability distributions over the set $\mathcal{A}$
δ_α	Dirac delta function, point density at α
$\boldsymbol{g}_\sharp p$	pushforward of a density p under perturbation $\boldsymbol{g}$
$X \perp Y$	random variable X is independent of random variable Y
$X \perp Y \mid Z$	random variable X is conditionally independent of random variable Y given random variable Z

$\mathbb{E}[X]$	expected value of random variable X
$\mathbb{E}_{x\sim X}[f(x)]$	expected value of the random variable $f(X)$, $\mathbb{E}[f(X)]$
$\mathbb{E}[X \mid Y]$	conditional expectation of random variable X given random variable Y
$\mathrm{Cov}[X, Y]$	covariance of random variable X and random variable Y
$\mathrm{Cor}[X, Y]$	correlation of random variable X and random variable Y
$\mathrm{Var}[X]$	variance of random variable X
$\mathrm{Var}[X \mid Y]$	conditional variance of random variable X given random variable Y
$\boldsymbol{\Sigma}_{\mathbf{X}}$	covariance matrix of random vector $\mathbf{X}$
$\boldsymbol{\Lambda}_{\mathbf{X}}$	precision matrix of random vector $\mathbf{X}$
$\mathrm{MSE}(X)$	mean squared error of random variable X
$\overline{X}_n$	sample mean of random variable X with n samples
S_n^2	sample variance of random variable X with n samples
$X_n \xrightarrow{\text{a.s.}} X$	the sequence of random variables X_n converges almost surely to X
$X_n \xrightarrow{\mathbb{P}} X$	the sequence of random variables X_n converges to X in probability
$X_n \xrightarrow{\mathcal{D}} X$	the sequence of random variables X_n converges to X in distribution
$\mathrm{S}[u]$	surprise associated with an event of probability u
$\mathrm{H}[p], \mathrm{H}[X]$	entropy of distribution p (or random variable X)
$\mathrm{H}[p\|q]$	cross-entropy of distribution q with respect to distribution p
$\mathrm{KL}(p\|q)$	KL-divergence of distribution q with respect to distribution p
$\mathrm{J}(p\|q)$	relative Fisher information of distribution q with respect to distribution p
$\mathrm{H}[X \mid Y]$	conditional entropy of random variable X given random variable Y
$\mathrm{H}[X, Y]$	joint entropy of random variables X and Y
$\mathrm{I}(X; Y)$	mutual information of random variables X and Y
$\mathrm{I}(X; Y \mid Z)$	conditional mutual information of random variables X and Y given random variable Z
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	normal distribution with mean $\boldsymbol{\mu}$ covariance $\boldsymbol{\Sigma}$
$\mathrm{Laplace}(\boldsymbol{\mu}, h)$	Laplace distribution with mean $\boldsymbol{\mu}$ scale h
$\mathrm{Unif}(S)$	uniform distribution on the set S
$\mathrm{Bern}(p)$	Bernoulli distribution with success probability p
$\mathrm{Bin}(n, p)$	binomial distribution with n trials and success probability p
$\mathrm{Beta}(\alpha, \beta)$	beta distribution with shape parameters α and β
$\mathrm{Gamma}(\alpha, \beta)$	gamma distribution with shape α and rate β
$\mathrm{Cauchy}(m, \tau)$	Cauchy distribution with location m and scale τ
$\mathrm{Pareto}(c, \alpha)$	Pareto distribution with cutoff threshold c and shape α

SUPERVISED LEARNING

$\boldsymbol{\theta}$	parameterization of a model
$\mathcal{X}$	input space
$\mathcal{Y}$	label space
$\boldsymbol{x} \in \mathcal{X}$	input

$\epsilon(\boldsymbol{x})$	zero-mean noise, sometimes assumed to be independent of $\boldsymbol{x}$
$y \in \mathcal{Y}$	(noisy) label, $f(\boldsymbol{x}) + \epsilon(\boldsymbol{x})$ where f is unknown
$\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$	labeled training data, $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$
$\boldsymbol{X} \in \mathbb{R}^{n \times d}$	design matrix when $\mathcal{X} = \mathbb{R}^d$
$\boldsymbol{\Phi} \in \mathbb{R}^{n \times e}$	design matrix in feature space $\mathbb{R}^e$
$\boldsymbol{y} \in \mathbb{R}^n$	label vector when $\mathcal{Y} = \mathbb{R}$
$p(\boldsymbol{\theta})$	prior belief about $\boldsymbol{\theta}$
$p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}, y_{1:n})$	posterior belief about $\boldsymbol{\theta}$ given training data
$p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta})$	likelihood of training data under the model parameterized by $\boldsymbol{\theta}$
$p(y_{1:n} \mid \boldsymbol{x}_{1:n})$	marginal likelihood of training data
$\hat{\boldsymbol{\theta}}_{\text{MLE}}$	maximum likelihood estimate of $\boldsymbol{\theta}$
$\hat{\boldsymbol{\theta}}_{\text{MAP}}$	maximum a posteriori estimate of $\boldsymbol{\theta}$
$\ell_{\text{nll}}(\boldsymbol{\theta}; \mathcal{D})$	negative log-likelihood of the training data $\mathcal{D}$ under model $\boldsymbol{\theta}$

BAYESIAN LINEAR MODELS

$\boldsymbol{w} \in \mathbb{R}^d$	weights of linear function $f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x}$
$\hat{\boldsymbol{w}}_{\text{ls}}$	least squares estimate of $\boldsymbol{w}$
$\hat{\boldsymbol{w}}_{\text{ridge}}$	ridge estimate of $\boldsymbol{w}$
$\hat{\boldsymbol{w}}_{\text{lasso}}$	lasso estimate of $\boldsymbol{w}$
$\mathcal{N}(\boldsymbol{0}, \sigma_p^2 \boldsymbol{I})$	prior
$\mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}, \sigma_n^2)$	likelihood
$\boldsymbol{\mu} \in \mathbb{R}^d$	posterior mean, $\sigma_n^{-2} \boldsymbol{\Sigma} \boldsymbol{X}^\top \boldsymbol{y}$
$\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$	posterior covariance matrix, $(\sigma_n^{-2} \boldsymbol{X}^\top \boldsymbol{X} + \sigma_p^{-2} \boldsymbol{I})^{-1}$
$\boldsymbol{K} \in \mathbb{R}^{n \times n}$	kernel matrix, $\sigma_p^2 \boldsymbol{X} \boldsymbol{X}^\top$
σ	logistic function
$\text{Bern}(\sigma(\boldsymbol{w}^\top \boldsymbol{x}))$	logistic likelihood
$\ell_{\log}(\cdot; \boldsymbol{x}, y)$	logistic loss of a single training example $(\boldsymbol{x}, y)$

KALMAN FILTERS

$\mathbf{X}_t$	sequence of hidden states in $\mathbb{R}^d$
$\mathbf{Y}_t$	sequence of observations in $\mathbb{R}^m$
$\boldsymbol{F} \in \mathbb{R}^{d \times d}$	motion model
$\boldsymbol{H} \in \mathbb{R}^{m \times d}$	sensor model
$\boldsymbol{\epsilon}_t$	zero-mean motion noise with covariance matrix $\boldsymbol{\Sigma}_x$
$\boldsymbol{\eta}_t$	zero-mean sensor noise with covariance matrix $\boldsymbol{\Sigma}_y$
$\boldsymbol{K}_t \in \mathbb{R}^{d \times m}$	Kalman gain

GAUSSIAN PROCESSES

$\mu : \mathcal{X} \to \mathbb{R}$	mean function
$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$	kernel function / covariance function
$f \sim \mathcal{GP}(\mu, k)$	f is a Gaussian process with mean function μ and kernel function k
$\mathcal{H}_k(\mathcal{X})$	reproducing kernel Hilbert space associated with kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

DEEP MODELS

$\boldsymbol{W}_l \in \mathbb{R}^{n_l \times n_{l-1}}$	weight matrix of layer l
$\boldsymbol{v}^{(l)} \in \mathbb{R}^{n_l}$	activations of layer l
φ	activation function
Tanh	hyperbolic tangent activation function
ReLU	rectified linear unit activation function
$\sigma_i(\boldsymbol{f})$	softmax function computing the probability mass of class i given outputs $\boldsymbol{f}$

VARIATIONAL INFERENCE

$\mathcal{Q}$	variational family
$\boldsymbol{\lambda} \in \Lambda$	variational parameters
$q_{\boldsymbol{\lambda}}$	variational posterior parameterized by $\boldsymbol{\lambda}$
$L(q, p; \mathcal{D})$	evidence lower bound for data $\mathcal{D}$ of variational posterior q and true posterior $p(\cdot \mid \mathcal{D})$

MARKOV CHAINS

S	set of n states
X_t	sequence of states
$p(x' \mid x)$	transition function, probability of going from state x to state x'
$p^{(t)}(x' \mid x)$	probability of reaching x' from x in exactly t steps
$\boldsymbol{P} \in \mathbb{R}^{n \times n}$	transition matrix
q_t	distribution over states at time t
π	stationary distribution
$\|\mu - \nu\|_{\text{TV}}$	total variation distance between two distributions μ and ν
τ_{TV}	mixing time with respect to total variation distance

MARKOV CHAIN MONTE CARLO METHODS

$r(\boldsymbol{x}' \mid \boldsymbol{x})$	proposal distribution, probability of proposing $\boldsymbol{x}'$ when in $\boldsymbol{x}$

$\alpha(\boldsymbol{x}' \mid \boldsymbol{x})$	acceptance distribution, probability of accepting the proposal $\boldsymbol{x}'$ when in $\boldsymbol{x}$
f	energy function

ACTIVE LEARNING

$S \subseteq \mathcal{X}$	set of observations
$I(S)$	maximization objective, quantifying the "information value" of S
$\Delta_I(\boldsymbol{x} \mid A)$	marginal gain of observation $\boldsymbol{x}$ with respect to objective I given prior observations A

BAYESIAN OPTIMIZATION

R_T	cumulative regret for time horizon T
$F(\boldsymbol{x}; \mu, \sigma)$	acquisition function
$\mathcal{C}_t(\boldsymbol{x})$	confidence interval of $f^\star(\boldsymbol{x})$ after round t
$\beta_t(\delta)$	scale of confidence interval to achieve confidence level δ
γ_T	maximum information gain after T rounds

REINFORCEMENT LEARNING

$X, \mathcal{X}$	set of states
$A, \mathcal{A}$	set of actions
$p(x' \mid x, a)$	dynamics model, probability of transitioning from state x to state x' when playing action a
r	reward function
X_t	sequence of states
A_t	sequence of actions
R_t	sequence of rewards
$\pi(a \mid x)$	policy, probability of playing action a when in state x
G_t	discounted payoff from time t
γ	discount factor
$v_t^\pi(x)$	state value function, average discounted payoff from time t starting from state x
$q_t^\pi(x, a)$	state-action value function, average discounted payoff from time t starting from state x playing action a
$a_t^\pi(x, a)$	advantage function, $q_t^\pi(x, a) - v_t^\pi(x)$
$j(\pi)$	policy value function, expected reward of policy π

1

Fundamentals

1.1 Probability

Probability is the natural extension of Boolean logic (where every statement is either true or false) to reasoning under uncertainty (Jaynes, 2003). In the frequentist interpretation, one interprets the probability of an event (say a coin coming up "heads" when flipping it) as the limit of relative frequencies in repeated independent experiments. That is,

$$\text{Probability} = \lim_{N \to \infty} \frac{\#\text{ events happening in } N \text{ trials}}{N}.$$

This interpretation is natural, but has a few issues. Often, probabilities are used to describe *beliefs* that one has about the outcome of a trial. While it is natural to consider the relative frequency of the outcome in repeated experiments as our belief, it is not very difficult to conceive of settings where repeated experiments do not make sense. Consider the outcome: "person X will live for at least 80 years." There is no way we can conduct multiple independent experiments. In these settings, our notion of probability is simply a (subjective) measure of uncertainty about occurrence of events and does not require any experiments. This notion is commonly understood as *Bayesian reasoning* or the Bayesian interpretation of probability.[1]

[1] In the early 20th century, Bruno De Finetti has done foundational work to formalize this notion.

Readings

In this section, we will briefly recall the fundamentals of probability theory, which we will need in the subsequent chapters.

For a more thorough introduction, read chapter 6 of "Mathematics for machine learning" (Deisenroth et al., 2020).

1.1.1 Probability Spaces

A probability space is a mathematical model for a random experiment. The set of all possible outcomes of the experiment Ω is called *sample space*. An *event* $A \subseteq \Omega$ of interest may be any combination of possible outcomes. The set of all events $\mathcal{A} \subseteq \mathcal{P}(\Omega)$ that we are interested in is often called the *event space* of the experiment.[2] This set of events is required to be a σ-algebra over the sample space.

[2] We use $\mathcal{P}(\Omega)$ to denote the *power set* (set of all subsets) of Ω.

Definition 1.1 (σ-algebra)**.** Given the set Ω, the set $\mathcal{A} \subseteq \mathcal{P}(\Omega)$ is a *σ-algebra* over Ω if the following properties are satisfied:

1. $\Omega \in \mathcal{A}$;
2. if $A \in \mathcal{A}$, then $\overline{A} \in \mathcal{A}$ (*closedness under complements*); and
3. if we have $A_i \in \mathcal{A}$ for all i, then $\bigcup_{i=1}^{\infty} A_i \in \mathcal{A}$ (*closedness under countable unions*).

Note that the three properties of σ-algebras correspond to characteristics we universally expect when working with random experiments. Namely, that we are able to reason about the event Ω that any of the possible outcomes occur, that we are able to reason about an event not occurring, and that we are able to reason about events that are composed of multiple (smaller) events.

Example 1.2: Event space of throwing a die

The event space $\mathcal{A}$ can also be thought of as "how much information is available about the experiment". For example, if the experiment is a throw of a die and Ω is the set of possible values on the die: $\Omega = \{1, \dots, 6\}$, then the following $\mathcal{A}$ implies that the observer cannot distinguish between 1 and 3:

$$\mathcal{A} \doteq \{\emptyset, \Omega, \{1, 3, 5\}, \{2, 4, 6\}\}.$$

Intuitively, the observer only understands the parity of the face of the die.

Definition 1.3 (Probability measure)**.** Given the set Ω and the σ-algebra $\mathcal{A}$ over Ω, the function

$$\mathbb{P} : \mathcal{A} \to \mathbb{R}$$

is a *probability measure* on $\mathcal{A}$ if the *Kolmogorov axioms* are satisfied:

1. $0 \leq \mathbb{P}(A) \leq 1$ for any $A \in \mathcal{A}$;
2. $\mathbb{P}(\Omega) = 1$; and
3. $\mathbb{P}(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mathbb{P}(A_i)$ for any countable set of mutually disjoint events $\{A_i \in \mathcal{A}\}_i$.[3]

[3] We say that a set of sets $\{A_i\}_i$ is disjoint if for all $i \neq j$ we have $A_i \cap A_j = \emptyset$.

Remarkably, all further statements about probability follow from these

three natural axioms. For an event $A \in \mathcal{A}$, we call $\mathbb{P}(A)$ the *probability* of A. We are now ready to define a probability space.

Definition 1.4 (Probability space)**.** A *probability space* is a triple $(\Omega, \mathcal{A}, \mathbb{P})$ where

- Ω is a sample space,
- $\mathcal{A}$ is a σ-algebra over Ω, and
- $\mathbb{P}$ is a probability measure on $\mathcal{A}$.

Exercise 1.5: Properties of probability

Let $(\Omega, \mathcal{A}, \mathbb{P})$ be a probability space. Derive the following properties of probability from the Kolmogorov axioms:

1. For any $A, B \in \mathcal{A}$, if $A \subseteq B$ then $\mathbb{P}(A) \leq \mathbb{P}(B)$.
2. For any $A \in \mathcal{A}$, $\mathbb{P}(\overline{A}) = 1 - \mathbb{P}(A)$.
3. For any countable set of events $\{A_i \in \mathcal{A}\}_i$,

$$\mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i\right) \leq \sum_{i=1}^{\infty} \mathbb{P}(A_i). \tag{1.1}$$

   which is called a *union bound*.

▷ *Solution*

Example 1.6: Borel σ-algebra over $\mathbb{R}$

In our context, we often have that Ω is the set of real numbers $\mathbb{R}$ or a compact subset of it. In this case, a natural event space is the σ-algebra generated by the set of events

$$A_x \doteq \{x' \in \Omega : x' \leq x\}.$$

The smallest σ-algebra $\mathcal{A}$ containing all sets A_x is called the *Borel σ-algebra*. $\mathcal{A}$ contains all "reasonable" subsets of Ω (except for some pathological examples). For example, $\mathcal{A}$ includes all singleton sets $\{x\}$, as well as all countable unions of intervals.

In the case of discrete Ω, in fact $\mathcal{A} = \mathcal{P}(\Omega)$, i.e., the Borel σ-algebra contains *all* subsets of Ω.

1.1.2 Random Variables

The set Ω is often rather complex. For example, take Ω to be the set of all possible graphs on n vertices. Then the outcome of our experiment is a graph. Usually, we are not interested in a specific graph but rather a property such as the number of edges, which is shared by many graphs. A function that maps a graph to its number of edges is a

random variable.

Definition 1.7 (Random variable)**.** A *random variable* X is a function

$$X : \Omega \to \mathcal{T}$$

where $\mathcal{T}$ is called *target space* of the random variable,[4] and where X respects the information available in the σ-algebra $\mathcal{A}$. That is,[5]

$$\forall S \subseteq \mathcal{T} : \quad \{\omega \in \Omega : X(\omega) \in S\} \in \mathcal{A}. \tag{1.2}$$

[4] For a random variable that maps a graph to its number of edges, $\mathcal{T} = \mathbb{N}_0$. For our purposes, you can generally assume $\mathcal{T} \subseteq \mathbb{R}$.

[5] In our example of throwing a die, X should assign the same value to the outcomes $1, 3, 5$.

Concrete values x of a random variable X are often referred to as *states* or *realizations* of X. The probability that X takes on a value in $S \subseteq \mathcal{T}$ is

$$\mathbb{P}(X \in S) = \mathbb{P}(\{\omega \in \Omega : X(\omega) \in S\}). \tag{1.3}$$

1.1.3 Distributions

Consider a random variable X on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$, where Ω is a compact subset of $\mathbb{R}$, and $\mathcal{A}$ the Borel σ-algebra.

In this case, we can refer to the probability that X assumes a particular state or set of states by writing

$$p_X(x) \doteq \mathbb{P}(X = x) \quad \text{(in the discrete setting)}, \tag{1.4}$$

$$P_X(x) \doteq \mathbb{P}(X \leq x). \tag{1.5}$$

Note that "$X = x$" and "$X \leq x$" are merely events (that is, they characterize subsets of the sample space Ω satisfying this condition) which are in the Borel σ-algebra, and hence their probability is well-defined.

Hereby, p_X and P_X are referred to as the probability mass function (PMF) and cumulative distribution function (CDF) of X, respectively. Note that we can also *implicitly* define probability spaces through random variables and their associated PMF/CDF, which is often very convenient.

Further, note that for continuous variables, $\mathbb{P}(X = x) = 0$. Here, instead we typically use the probability density function (PDF), to which we (with slight abuse of notation) also refer with p_X. We discuss densities in greater detail in section 1.1.4.

Remark 1.8: Notation
We use $X \sim P$ to denote that a random variable X is distributed according to the distribution P. Due to the unique characterization of random variables through distributions, the two notions are often conflated. Sometimes we use P as a random variable (but

refer to a random variable that is distributed according to P).

This notation is not to be confused with $x \sim P$, which we use to denote that a value x is sampled according to a distribution P.

Further, we often abbreviate $\mathbb{P}(X = x)$ by $\mathbb{P}(x)$ if the correspondence between X and x is clear from the context. Sometimes, we also use $p(x)$ equivalently (to denote the PMF or PDF of X).

Example 1.9: Common discrete distributions

- *Bernoulli* $\text{Bern}(p)$ describes (biased) coin flips. Its domain is $\Omega = \{0, 1\}$ where 1 corresponds to heads and 0 corresponds to tails. $p \in [0, 1]$ is the probability of the coin landing heads, that is, $\mathbb{P}(X = 1) = p$ and $\mathbb{P}(X = 0) = 1 - p$.
- *Binomial* $\text{Bin}(n, p)$ counts the number of heads in n independent Bernoulli trials, each with probability of heads p.
- *Categorical* $\text{Cat}(m, p_1, \ldots, p_m)$ is a generalization of the Bernoulli distribution and represents throwing a (biased) m-sided die. Its domain is $\Omega = [m]$ and we have $\mathbb{P}(X = i) = p_i$. We require $p_i \geq 0$ and $\sum_{i \in [m]} p_i = 1$.
- *Multinomial* $\text{Mult}(n, m, p_1, \ldots, p_m)$ counts the number of outcomes of each side in n independent Categorical trials.
- *Uniform* $\text{Unif}(S)$ assigns identical probability mass to all values in the set S. That is, $\mathbb{P}(X = x) = \frac{1}{|S|}$ $(\forall x \in S)$.

Remark 1.10: Probability simplex

We denote by Δ^m the set of all categorical distributions on m classes. Observe that Δ^m is an $(m-1)$-dimensional convex polytope.

To see this, let us consider the m-dimensional space of probabilities $[0, 1]^m$. It follows from our characterization of the categorical distribution in example 1.9 that there is a one-to-one correspondence between probability distributions over m classes and points in the space $[0, 1]^m$ where all coordinates sum to one.

This $(m-1)$-dimensional subspace of $[0, 1]^m$ is also known as the *probability simplex*.

We call the subset $S \subseteq \mathcal{T}$ of the domain of a PMF or PDF p_X such that all elements $x \in S$ have positive probability, $p_X(x) > 0$, the *support* of the distribution p_X. This quantity is denoted by $X(\Omega)$.

1.1.4 Continuous Distributions

As mentioned, a continuous random variable can be characterized by its *probability density function* (PDF). But what is a density? We can derive some intuition from physics.

Let M be a (non-homogeneous) physical object, e.g., a rock. We commonly use $m(M)$ and $\text{vol}(M)$ to refer to its mass and volume, respectively. Now, consider for a point $\boldsymbol{x} \in M$ and a ball $B_r(\boldsymbol{x})$ around $\boldsymbol{x}$ with radius r the following quantities:

$$\lim_{r \to 0} \text{vol}(B_r(\boldsymbol{x})) = 0 \quad \lim_{r \to 0} m(B_r(\boldsymbol{x})) = 0.$$

They appear utterly uninteresting at first, yet, if we divide them, we get what is called the *density* of M at $\boldsymbol{x}$.

$$\lim_{r \to 0} \frac{m(B_r(\boldsymbol{x}))}{\text{vol}(B_r(\boldsymbol{x}))} \doteq \rho(\boldsymbol{x}).$$

We know that the relationship between density and mass is described by the following formula:

$$m(M) = \int_M \rho(\boldsymbol{x}) \, d\boldsymbol{x}.$$

In other words, the density is to be integrated. For a small region I around $\boldsymbol{x}$, we can approximate $m(I) \approx \rho(\boldsymbol{x}) \cdot \text{vol}(I)$.

Crucially, observe that even though the mass of any particular point $\boldsymbol{x}$ is zero, i.e., $m(\{\boldsymbol{x}\}) = 0$, assigning a density $\rho(\boldsymbol{x})$ to $\boldsymbol{x}$ is useful for integration and approximation. The same idea applies to continuous random variables, only that volume corresponds to intervals on the real line and mass to probability. Recall that probability density functions are normalized such that their probability mass across the entire real line integrates to one.

Example 1.11: Normal distribution / Gaussian

An example of a continuous distribution is the *normal distribution*, also called *Gaussian*. We say, a random variable X is *normally distributed*, $X \sim \mathcal{N}(\mu, \sigma^2)$, if its PDF is

$$\mathcal{N}(x; \mu, \sigma^2) \doteq \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{1.6}$$

We have $\mathbb{E}[X] = \mu$ and $\text{Var}[X] = \sigma^2$. If $\mu = 0$ and $\sigma^2 = 1$, this distribution is called the *standard normal distribution*. The Gaussian CDF cannot be expressed in closed-form.

Note that the mean of a Gaussian distribution coincides with the maximizer of its PDF, also called *mode* of a distribution.

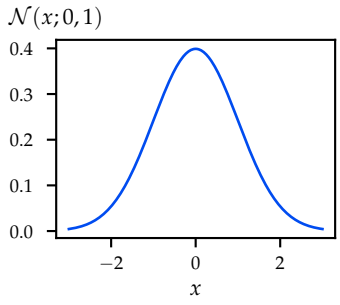


Figure 1.1: PDF of the standard normal distribution. Observe that the PDF is symmetric around the mode.

Much of the relevance of the normal distribution stems from the *central limit theorem*, which states that the sum of any i.i.d. (independent and identically distributed) samples tends to a normal distribution as the sample size goes to infinity (even if the samples themselves are not normally distributed).

Fact 1.12 (Central limit theorem by Lindeberg-Lévy)**.** *Given independent and identically distributed random variables $X_1, \dots, X_n$ with finite expectation μ and variance σ^2. Then,*

$$\frac{1}{\sqrt{n}} \sum_{i=1}^{n} (X_i - \mu) = \sqrt{n}(\overline{X}_n - \mu) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \sigma^2) \tag{1.7}$$

as $n \to \infty$.

Here, $\overline{X}_n$ denotes the sample mean of $X_1, \dots, X_n$ (see example 1.36) and $\xrightarrow{\mathcal{D}}$ denotes convergence in distribution (see eq. (1.80)).

1.1.5 *Universality of the Uniform and Sampling*

Sampling from a continuous random variable is a common and nontrivial problem. A family of techniques is motivated by the following general property of the uniform distribution, colloquially known as the "universality of the uniform".

Uniform distribution The (continuous) *uniform distribution* $\text{Unif}([a, b])$ is the only distribution that assigns constant density to all points in the support $[a, b]$. That is, it has PDF

$$p(u) = \begin{cases} \frac{1}{b-a} & \text{if } u \in [a, b] \\ 0 & \text{otherwise} \end{cases}$$

and CDF

$$P(u) = \begin{cases} \frac{u-a}{b-a} & \text{if } u \in [a, b] \\ 0 & \text{otherwise.} \end{cases}$$

First, given a random variable $X \sim P$, we call

$$P^{-1}(u) \doteq \min\{x \mid P(x) \geq u\} \quad \text{for all } 0 < u < 1. \tag{1.8}$$

the *quantile function* of X. That is, $P^{-1}(u)$ corresponds to the value x such that the probability of X being at most x is u. If the CDF P is invertible, then P^{-1} coincides with the inverse of P.

Theorem 1.13 (Universality of the uniform)**.** *If $U \sim \text{Unif}([0, 1])$ and P is invertible, then $P^{-1}(U) \sim P$.*

Proof. Let $U \sim \text{Unif}([0, 1])$. Then,

$$\mathbb{P}\Big(P^{-1}(U) \leq x\Big) = \mathbb{P}(U \leq P(x)) = P(x). \qquad \square$$

using $\mathbb{P}(U \leq u) = u$ if $U \sim \text{Unif}([0, 1])$

This implies that if we are able to sample from $\text{Unif}([0, 1])$,[6] then we are able to sample from any distribution with invertible CDF. This method is known as *inverse transform sampling*.

[6] This is done in practice using so-called *pseudo-random number generators*.

In the case of Gaussians, we learned that the CDF cannot be expressed in closed-form (and hence, is not invertible), however, for practical purposes, the quantile function of Gaussians can be approximated well.

1.1.6 Joint Probability

A joint probability (as opposed to a marginal probability) is the probability of two or more events occurring simultaneously:

$$\mathbb{P}(A, B) \doteq \mathbb{P}(A \cap B). \tag{1.9}$$

In terms of random variables, this concept extends to joint distributions. Instead of characterizing a single random variable, a *joint distribution* is a function $p_{\mathbf{X}} : \mathbb{R}^n \to \mathbb{R}$, characterizing a *random vector* $\mathbf{X} \doteq [X_1 \ \cdots \ X_n]^\top$. For example, if the X_i are discrete, the joint distribution characterizes joint probabilities of the form

$$\mathbb{P}(\mathbf{X} = [x_1, \ldots, x_n]) = \mathbb{P}(X_1 = x_1, \ldots, X_n = x_n),$$

and hence describes the relationship among all variables X_i. We use $X_{i:j}$ to denote the random vector $[X_i \ \cdots \ X_j]^\top$.

We can "sum out" (respectively "integrate out") variables from a joint distribution in a process called "marginalization".

Fact 1.14 (Sum rule). *We have that*

$$p(x_{1:i-1}, x_{i+1:n}) = \sum_{x_i \in X_i(\Omega)} p(x_{1:i-1}, x_i, x_{i+1:n}), \tag{1.10a}$$

$$p(x_{1:i-1}, x_{i+1:n}) = \int_{X_i(\Omega)} p(x_{1:i-1}, x_i, x_{i+1:n}) \, dx_i. \tag{1.10b}$$

if X_i is discrete and continuous respectively.

1.1.7 Conditional Probability

Conditional probability updates the probability of an event A given some new information, for example, after observing the event B.

Definition 1.15 (Conditional probability). Given two events A and B such that $\mathbb{P}(B) > 0$, the probability of A conditioned on B is given as

$$\mathbb{P}(A \mid B) \doteq \frac{\mathbb{P}(A, B)}{\mathbb{P}(B)}. \tag{1.11}$$

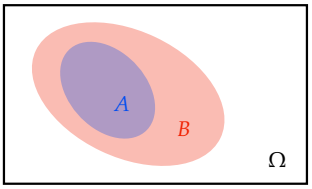


Figure 1.2: Conditioning an event A on another event B can be understood as replacing the universe of all possible outcomes Ω by the observed outcomes B. Then, the conditional probability is simply expressing the likelihood of A given that B occurred.

Simply rearranging the terms yields,

$$\mathbb{P}(A, B) = \mathbb{P}(A \mid B) \cdot \mathbb{P}(B) = \mathbb{P}(B \mid A) \cdot \mathbb{P}(A). \tag{1.12}$$

Thus, the probability that both A and B occur can be calculated by multiplying the probability of event A and the probability of B conditional on A occurring.

We say $\mathbf{Z} \sim \mathbf{X} \mid \mathbf{Y} = \boldsymbol{y}$ (or simply $\mathbf{Z} \sim \mathbf{X} \mid \boldsymbol{y}$) if $\mathbf{Z}$ follows the *conditional distribution*

$$p_{\mathbf{X}|\mathbf{Y}}(\boldsymbol{x} \mid \boldsymbol{y}) \doteq \frac{p_{\mathbf{X},\mathbf{Y}}(\boldsymbol{x}, \boldsymbol{y})}{p_{\mathbf{Y}}(\boldsymbol{y})}. \tag{1.13}$$

If X and Y are discrete, we have that $p_{X|Y}(x \mid y) = \mathbb{P}(X = x \mid Y = y)$ as one would naturally expect.

Remark 1.16: Notation
When $\mathbf{X} \sim p$, we write $p(\cdot \mid \boldsymbol{y})$ to denote the distribution of the random vector $\mathbf{X} \mid \boldsymbol{y}$. In the literature, $p(\mathbf{X} \mid \boldsymbol{y})$ is sometimes used to denote equivalent quantities.

When sampling $\boldsymbol{x} \sim p(\cdot \mid \boldsymbol{y})$ and the distribution p is clear from context, we abbreviate $\boldsymbol{x} \mid \boldsymbol{y}$.

As noted previously, if the correspondence between the random vectors $\mathbf{X}$ and $\mathbf{Y}$ and their states $\boldsymbol{x}$ and $\boldsymbol{y}$ is clear from context, we often omit the indices of the PDF p and simply write $p(\boldsymbol{x} \mid \boldsymbol{y})$, $p(\boldsymbol{x}, \boldsymbol{y})$ and $p(\boldsymbol{y})$. However, this does *not* mean that they refer to the same distribution p.

Extending eq. (1.12) to arbitrary random vectors yields the product rule (also called *chain rule of probability*).

Fact 1.17 (Product rule)**.** *Given random variables $X_{1:n}$,*

$$p(x_{1:n}) = p(x_1) \cdot \prod_{i=2}^{n} p(x_i \mid x_{1:i-1}). \tag{1.14}$$

Combining sum rule and product rule, we can compute marginal probabilities too. If $\mathbf{Y}$ is continuous, then

$$p(\boldsymbol{x}) = \int_{\mathbf{Y}(\Omega)} p(\boldsymbol{x}, \boldsymbol{y})\, d\boldsymbol{y} = \int_{\mathbf{Y}(\Omega)} p(\boldsymbol{x} \mid \boldsymbol{y}) \cdot p(\boldsymbol{y})\, d\boldsymbol{y} \tag{1.15}$$

first using the sum rule (1.10) then the product rule (1.14)

Analogously, one can condition on a discrete random variable by replacing the integral with a sum. The above is called the *law of total probability* (LOTP). Colloquially, this is often referred to as to "condition" on $\mathbf{Y}$. If it is difficult to compute $p(\boldsymbol{x})$ directly, conditioning can be a useful technique when $\mathbf{Y}$ is chosen such that the densities $p(\boldsymbol{x} \mid \boldsymbol{y})$ and $p(\boldsymbol{y})$ are straightforward to understand.

Example 1.18: Point densities and Dirac's delta function
The *Dirac delta function* δ_α is a function satisfying $\delta_\alpha(x) = 0$ for any $x \neq \alpha$ and $\int \delta_\alpha(x)\, dx = 1$. δ_α is also called a *point density* at α.

As an example, let us consider the random variable $Y = g(X)$, which is defined in terms of another random variable X and a continuously differentiable function $g : \mathbb{R} \to \mathbb{R}$. Using the sum

rule and product rule, we can express the PDF of Y as

$$p(y) = \int p(y \mid x) \cdot p(x)\, dx = \int \delta_{g(x)}(y) \cdot p(x)\, dx. \tag{1.16}$$

In section 1.1.12, we discuss how one can obtain an explicit representation of p_Y using the "change of variables" formula.

Exercise 1.19: Random walks on graphs

Let G be a simple connected finite graph. We start at a vertex u of G. At every step, we move to one of the neighbors of the current vertex uniformly at random, e.g., if the vertex has 3 neighbors, we move to one of them, each with probability $1/3$. What is the probability that the walk visits a vertex v eventually?

▷ *Solution*

1.1.8 Independence

Two random vectors $\mathbf{X}$ and $\mathbf{Y}$ are *independent* (denoted $\mathbf{X} \perp \mathbf{Y}$) iff knowledge about the state of one random vector does not affect the distribution of the other random vector, namely if their conditional CDF (or in case they have a joint density, their conditional PDF) simplifies to

$$P_{\mathbf{X}|\mathbf{Y}}(\boldsymbol{x} \mid \boldsymbol{y}) = P_{\mathbf{X}}(\boldsymbol{x}), \quad p_{\mathbf{X}|\mathbf{Y}}(\boldsymbol{x} \mid \boldsymbol{y}) = p_{\mathbf{X}}(\boldsymbol{x}). \tag{1.17}$$

For the conditional probabilities to be well-defined, we need to assume that $\mathbb{P}(\mathbf{Y} = \boldsymbol{y}) > 0$.

The more general characterization of independence is that $\mathbf{X}$ and $\mathbf{Y}$ are independent if and only if their joint CDF (or in case they have a joint density, their joint PDF) can be decomposed as follows:

$$P_{\mathbf{X},\mathbf{Y}}(\boldsymbol{x}, \boldsymbol{y}) = P_{\mathbf{X}}(\boldsymbol{x}) \cdot P_{\mathbf{Y}}(\boldsymbol{y}), \quad p_{\mathbf{X},\mathbf{Y}}(\boldsymbol{x}, \boldsymbol{y}) = p_{\mathbf{X}}(\boldsymbol{x}) \cdot p_{\mathbf{Y}}(\boldsymbol{y}). \tag{1.18}$$

The equivalence of the two characterizations (when $\mathbb{P}(\boldsymbol{y}) > 0$) is easily proven using the product rule: $P_{\mathbf{X},\mathbf{Y}}(\boldsymbol{x}, \boldsymbol{y}) = P_{\mathbf{Y}}(\boldsymbol{y}) \cdot P_{\mathbf{X}|\mathbf{Y}}(\boldsymbol{x} \mid \boldsymbol{y})$.

We often say that a set of random variables $\{X_i\}_i$ is i.i.d. to abbreviate that the X_i are "(mutually) independent and identically distributed", and we write $X_i \overset{\text{iid}}{\sim} p$.

A "weaker" notion of independence is conditional independence.[7] Two random vectors $\mathbf{X}$ and $\mathbf{Y}$ are *conditionally independent* given a random vector $\mathbf{Z}$ (denoted $\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z}$) iff, given $\mathbf{Z}$, knowledge about the

[7] It is explained in remark 1.20 how "weaker" is to be interpreted in this context.

value of one random vector $\mathbf{Y}$ does not affect the distribution of the other random vector $\mathbf{X}$, namely if

$$P_{\mathbf{X}|\mathbf{Y},\mathbf{Z}}(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z}) = P_{\mathbf{X}|\mathbf{Z}}(\boldsymbol{x} \mid \boldsymbol{z}), \tag{1.19a}$$

$$p_{\mathbf{X}|\mathbf{Y},\mathbf{Z}}(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z}) = p_{\mathbf{X}|\mathbf{Z}}(\boldsymbol{x} \mid \boldsymbol{z}). \tag{1.19b}$$

Similarly to independence, we have that $\mathbf{X}$ and $\mathbf{Y}$ are conditionally independent given $\mathbf{Z}$ if and only if their joint CDF or joint PDF can be decomposed as follows:

$$P_{\mathbf{X},\mathbf{Y}|\mathbf{Z}}(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{z}) = P_{\mathbf{X}|\mathbf{Z}}(\boldsymbol{x} \mid \boldsymbol{z}) \cdot P_{\mathbf{Y}|\mathbf{Z}}(\boldsymbol{y} \mid \boldsymbol{z}), \tag{1.20a}$$

$$p_{\mathbf{X},\mathbf{Y}|\mathbf{Z}}(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{z}) = p_{\mathbf{X}|\mathbf{Z}}(\boldsymbol{x} \mid \boldsymbol{z}) \cdot p_{\mathbf{Y}|\mathbf{Z}}(\boldsymbol{y} \mid \boldsymbol{z}). \tag{1.20b}$$

Remark 1.20: Common causes
How can conditional independence be understood as a "weaker" notion of independence? Clearly, conditional independence does not imply independence: a trivial example is $X \perp X \mid X \not\Longrightarrow X \perp X$.[8] Neither does independence imply conditional independence: for example, $X \perp Y \not\Longrightarrow X \perp Y \mid X + Y$.[9]

When we say that conditional independence is a weaker notion we mean to emphasize that X and Y can be "made" (conditionally) independent by conditioning on the "right" Z even if X and Y are dependent. This is known as *Reichenbach's common cause principle* which says that for any two random variables $X \not\perp Y$ there exists a random variable Z (which may be X or Y) that causally influences both X and Y, and which is such that $X \perp Y \mid Z$.

[8] $X \perp X \mid X$ is true trivially.

[9] Knowing X and $X + Y$ already implies the value of Y, and hence, $X \not\perp Y \mid X + Y$.

The independence and conditional independence of events is defined analogously.

1.1.9 *Directed Graphical Models*

Directed graphical models (also called *Bayesian networks*) are often used to visually denote the (conditional) independence relationships of a large number of random variables. They are a schematic representation (as a directed acyclic graph) of the factorization of the joint distribution into a product of conditional distributions. Given the sequence of random variables $\{X_i\}_{i=1}^n$, their joint distribution can be expressed as

$$\mathbb{P}(X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} \mathbb{P}(X_i = x_i \mid \text{parents}(X_i)) \tag{1.21}$$

where $\text{parents}(X_i)$ is the set of parents of the vertex X_i in the directed graphical model. In other words, the parenthood relationship encodes

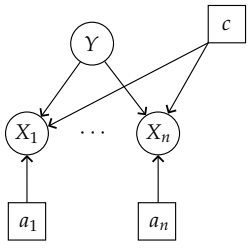


Figure 1.3: Example of a directed graphical model. The random variables $X_1, \ldots, X_n$ are mutually independent given the random variable Y. The squared rectangular nodes are used to represent dependencies on parameters $c, a_1, \ldots, a_n$.

a conditional independence of a random variable X with a random variable Y given their parents,[10]

$$X \perp Y \mid \text{parents}(X), \text{parents}(Y). \tag{1.22}$$

[10] More generally, vertices u and v are conditionally independent given a set of vertices Z if Z *d-separates* u and v, which we will not cover in depth here.

Thus, eq. (1.21) simply uses the product rule and the conditional independence relationships to factorize the joint distribution.

An example of a directed graphical model is given in fig. 1.3. Circular vertices represent random quantities (i.e., random variables). In contrast, square vertices are commonly used to represent deterministic quantities (i.e., parameters that the distributions depend on). In the given example, we have that X_i is conditionally independent of all other X_j given Y.

Plate notation is a condensed notation used to represent repeated variables of a graphical model. An example is given in fig. 1.4.

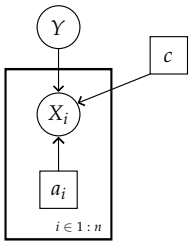


Figure 1.4: The same directed graphical model as in fig. 1.3 using plate notation.

1.1.10 *Expectation*

The *expected value* $\mathbb{E}[\mathbf{X}]$ of a random vector $\mathbf{X}$ is the (asymptotic) arithmetic mean of an arbitrarily increasing number of independent realizations of $\mathbf{X}$. We have

$$\mathbb{E}[\mathbf{X}] \doteq \sum_{\boldsymbol{x} \in \mathbf{X}(\Omega)} \boldsymbol{x} \cdot p(\boldsymbol{x}) \tag{1.23a}$$

$$\mathbb{E}[\mathbf{X}] \doteq \int_{\mathbf{X}(\Omega)} \boldsymbol{x} \cdot p(\boldsymbol{x}) \, d\boldsymbol{x} \tag{1.23b}$$

if $\mathbf{X}$ is discrete or continuous, respectively.[11] A very special and often used property of expectations is their *linearity*. For any random vectors $\mathbf{X}$ and $\mathbf{Y}$ in $\mathbb{R}^n$ and any $\boldsymbol{A} \in \mathbb{R}^{m \times n}, \boldsymbol{b} \in \mathbb{R}^m$ we have

$$\mathbb{E}[\boldsymbol{A}\mathbf{X} + \boldsymbol{b}] = \boldsymbol{A}\mathbb{E}[\mathbf{X}] + \boldsymbol{b} \quad \text{and} \tag{1.24}$$

$$\mathbb{E}[\mathbf{X} + \mathbf{Y}] = \mathbb{E}[\mathbf{X}] + \mathbb{E}[\mathbf{Y}]. \tag{1.25}$$

[11] In infinite probability spaces, absolute convergence of $\mathbb{E}[\mathbf{X}]$ is necessary for the existence of $\mathbb{E}[\mathbf{X}]$.

Note that $\mathbf{X}$ and $\mathbf{Y}$ do not necessarily have to be independent! Further, if $\mathbf{X}$ and $\mathbf{Y}$ are independent then

$$\mathbb{E}\left[\mathbf{X}\mathbf{Y}^\top\right] = \mathbb{E}[\mathbf{X}] \cdot \mathbb{E}[\mathbf{Y}]^\top. \tag{1.26}$$

Remark 1.21: Notation of summary statistics

Often, when taking the expectation, the brackets around the random vector are omitted: $\mathbb{E}\mathbf{X}$. For notational convenience, we often use the state $\boldsymbol{x}$ of the random vector $\mathbf{X}$ within the brackets: $\mathbb{E}_{\boldsymbol{x} \sim \mathbf{X}}[\boldsymbol{x}]$. If the correspondence between $\boldsymbol{x}$ and $\mathbf{X}$ is clear from

context, we often abbreviate this expectation with $\mathbb{E}_{\mathbf{X}}[\boldsymbol{x}]$ or $\mathbb{E}[\boldsymbol{x}]$. The analogous notation is used for covariance and variance (cf. section 1.1.11).

Remark 1.22: Gradients of expectations

We will often encounter gradients of expectations, $\boldsymbol{\nabla}_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{X}}[f(\mathbf{X}, \boldsymbol{\theta})]$.

Fact 1.23 (Differentiation under the integral sign)**.** *Let $f(\boldsymbol{x}, t)$ be differentiable in $\boldsymbol{x}$, integrable in t, and be such that*

$$\left| \frac{\partial f(\boldsymbol{x}, t)}{\partial t} \right| \leq g(\boldsymbol{x})$$

for some integrable function g. Then, if the distribution of $\mathbf{X}$ is not parameterized by t,[12]

$$\frac{\partial \mathbb{E}[f(\mathbf{X}, t)]}{\partial t} = \mathbb{E}\left[\frac{\partial f(\mathbf{X}, t)}{\partial t} \right]. \tag{1.27}$$

Therefore, if the distribution of $\mathbf{X}$ is not parameterized by $\boldsymbol{\theta}$,

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} \mathbb{E}[f(\mathbf{X}, \boldsymbol{\theta})] = \mathbb{E}[\boldsymbol{\nabla}_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta})]. \tag{1.28}$$

[12] That is, $\mathbb{E}[f(\mathbf{X}, t)] = \int p(\boldsymbol{x}) \cdot f(\boldsymbol{x}, t)\, d\boldsymbol{x}$ and $p(\boldsymbol{x})$ does not depend on t.

Gradient The *gradient* of a function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $\boldsymbol{x} \in \mathbb{R}^n$ is

$$\boldsymbol{\nabla} f(\boldsymbol{x}) \doteq \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}(1)} & \cdots & \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}(n)} \end{bmatrix}^\top. \tag{1.29}$$

The following intuitive lemma can be used to compute expectations of transformed random variables.

Fact 1.24 (Law of the unconscious statistician, LOTUS)**.**

$$\mathbb{E}[\boldsymbol{g}(\mathbf{X})] = \int_{\mathbf{X}(\Omega)} \boldsymbol{g}(\boldsymbol{x}) \cdot p(\boldsymbol{x})\, d\boldsymbol{x} \tag{1.30}$$

where $\boldsymbol{g} : \mathbf{X}(\Omega) \to \mathbb{R}^n$ is a "nice" function[13] *and $\mathbf{X}$ is a continuous random vector. The analogous statement with a sum replacing the integral holds for discrete random variables.*

[13] $\boldsymbol{g}$ being a differentiable function whose inverse is monotonic (this is satisfied in most cases) is sufficient.

This is a nontrivial fact that can be proven using the change of variables formula discussed in section 1.1.12.

Similarly to conditional probability, we can also define conditional expectations. The expectation of a continuous random vector $\mathbf{X}$ given that $\mathbf{Y} = \boldsymbol{y}$ is defined as

$$\mathbb{E}[\mathbf{X} \mid \mathbf{Y} = \boldsymbol{y}] \doteq \int_{\mathbf{X}(\Omega)} \boldsymbol{x} \cdot p_{\mathbf{X}|\mathbf{Y}}(\boldsymbol{x} \mid \boldsymbol{y})\, d\boldsymbol{x}. \tag{1.31}$$

The definition is analogous for discrete random variables. Note that $\mathbb{E}[\mathbf{X} \mid \mathbf{Y} = \cdot]$ defines a deterministic mapping from $\boldsymbol{y}$ to $\mathbb{E}[\mathbf{X} \mid \mathbf{Y} = \boldsymbol{y}]$.

Therefore, $\mathbb{E}[\mathbf{X} \mid \mathbf{Y}]$ is itself a random vector:

$$\mathbb{E}[\mathbf{X} \mid \mathbf{Y}](\omega) = \mathbb{E}[\mathbf{X} \mid \mathbf{Y} = \mathbf{Y}(\omega)] \tag{1.32}$$

where $\omega \in \Omega$. This random vector $\mathbb{E}[\mathbf{X} \mid \mathbf{Y}]$ is called the *conditional expectation* of $\mathbf{X}$ given $\mathbf{Y}$.

Theorem 1.25 (Tower rule)**.** *Given random vectors* $\mathbf{X}$ *and* $\mathbf{Y}$*, we have*

$$\mathbb{E}_{\mathbf{Y}}[\mathbb{E}_{\mathbf{X}}[\mathbf{X} \mid \mathbf{Y}]] = \mathbb{E}[\mathbf{X}]. \tag{1.33}$$

Proof. We only prove the case where $\mathbf{X}$ and $\mathbf{Y}$ have a joint density. We have

$$\mathbb{E}[\mathbf{X} \mid \mathbf{Y} = \boldsymbol{y}] = \int \boldsymbol{x} \cdot p(\boldsymbol{x} \mid \boldsymbol{y}) \, d\boldsymbol{x}$$

and hence

$$\begin{aligned}
\mathbb{E}[\mathbb{E}[\mathbf{X} \mid \mathbf{Y}]] &= \int \left(\int \boldsymbol{x} \cdot p(\boldsymbol{x} \mid \boldsymbol{y}) \, d\boldsymbol{x} \right) p(\boldsymbol{y}) \, d\boldsymbol{y} \\
&= \int \int \boldsymbol{x} \cdot p(\boldsymbol{x}, \boldsymbol{y}) \, d\boldsymbol{x} \, d\boldsymbol{y} && \text{by definition of conditional densities (1.13)} \\
&= \int \boldsymbol{x} \int p(\boldsymbol{x}, \boldsymbol{y}) \, d\boldsymbol{y} \, d\boldsymbol{x} && \text{by Fubini's theorem} \\
&= \int \boldsymbol{x} \cdot p(\boldsymbol{x}) \, d\boldsymbol{x} && \text{using the sum rule (1.10)} \\
&= \mathbb{E}[\mathbf{X}]. && \square
\end{aligned}$$

Note that the tower rule can be used analogously to the law of total probability (1.15) to "condition" on a random vector $\mathbf{Y}$. For this reason, the tower rule is also known as the *law of total expectation* (LOTE).

Exercise 1.26: Law of total expectation

Show that if $\{A_i\}_{i=1}^k$ are a partition of Ω and $\mathbf{X}$ is a random vector, then

$$\mathbb{E}[\mathbf{X}] = \sum_{i=1}^{k} \mathbb{E}[\mathbf{X} \mid A_i] \cdot \mathbb{P}(A_i). \tag{1.34}$$

▷ *Solution*

1.1.11 Covariance and Variance

Covariance measures the linear dependence between two random vectors. Given two random vectors $\mathbf{X}$ in $\mathbb{R}^n$ and $\mathbf{Y}$ in $\mathbb{R}^m$, their *covariance* is defined as

$$\operatorname{Cov}[\mathbf{X}, \mathbf{Y}] \doteq \mathbb{E}\Big[(\mathbf{X} - \mathbb{E}[\mathbf{X}])(\mathbf{Y} - \mathbb{E}[\mathbf{Y}])^\top \Big] \tag{1.35}$$

$$= \mathbb{E}\left[\mathbf{X}\mathbf{Y}^\top\right] - \mathbb{E}[\mathbf{X}] \cdot \mathbb{E}[\mathbf{Y}]^\top \tag{1.36}$$

$$= \mathrm{Cov}[\mathbf{Y}, \mathbf{X}]^\top \in \mathbb{R}^{n \times m}. \tag{1.37}$$

A direct consequence of the definition of covariance (1.35) is that given linear maps $\boldsymbol{A} \in \mathbb{R}^{n' \times n}, \boldsymbol{B} \in \mathbb{R}^{m' \times m}$, vectors $\boldsymbol{c} \in \mathbb{R}^{n'}, \boldsymbol{d} \in \mathbb{R}^{m'}$ and random vectors $\mathbf{X}$ in $\mathbb{R}^n$ and $\mathbf{Y}$ in $\mathbb{R}^m$, we have that

$$\mathrm{Cov}[\boldsymbol{A}\mathbf{X} + \boldsymbol{c}, \boldsymbol{B}\mathbf{Y} + \boldsymbol{d}] = \boldsymbol{A}\mathrm{Cov}[\mathbf{X}, \mathbf{Y}]\boldsymbol{B}^\top. \tag{1.38}$$

Two random vectors $\mathbf{X}$ and $\mathbf{Y}$ are said to be *uncorrelated* iff $\mathrm{Cov}[\mathbf{X}, \mathbf{Y}] = \mathbf{0}$. Note that if $\mathbf{X}$ and $\mathbf{Y}$ are independent, then eq. (1.26) implies that $\mathbf{X}$ and $\mathbf{Y}$ are uncorrelated. The reverse does not hold in general.

Remark 1.27: Correlation

The *correlation* of the random vectors $\mathbf{X}$ and $\mathbf{Y}$ is a normalized covariance,

$$\mathrm{Cor}[\mathbf{X}, \mathbf{Y}](i, j) \doteq \frac{\mathrm{Cov}\left[X_i, Y_j\right]}{\sqrt{\mathrm{Var}[X_i]\mathrm{Var}\left[Y_j\right]}} \in [-1, 1]. \tag{1.39}$$

Two random vectors $\mathbf{X}$ and $\mathbf{Y}$ are uncorrelated iff $\mathrm{Cor}[\mathbf{X}, \mathbf{Y}] = \mathbf{0}$.

There is also a nice geometric interpretation of covariance and correlation. For zero mean random variables X and Y, $\mathrm{Cov}[X, Y]$ is an inner product.[14]

[14] That is,
- $\mathrm{Cov}[X, Y]$ is symmetric,
- $\mathrm{Cov}[X, Y]$ is linear (here we use $\mathbb{E}X = \mathbb{E}Y = 0$), and
- $\mathrm{Cov}[X, X] \geq 0$.

The length of a random variable X in this inner product space is called its *standard deviation*,

$$\|X\| = \sqrt{\mathrm{Cov}[X, X]} = \sqrt{\mathrm{Var}[X]} \doteq \sigma[X]. \tag{1.40}$$

That is, the longer a random variable is in the inner product space, the more "uncertain" we are about its value. If a random variable has length 0, then it is deterministic.

The cosine of the angle θ between X and Y (that are not deterministic) coincides with their correlation,

$$\cos\theta = \frac{\mathrm{Cov}[X, Y]}{\|X\| \, \|Y\|} = \mathrm{Cor}[X, Y]. \tag{1.41}$$

using the Euclidean inner product formula, $\mathrm{Cov}[X, Y] = \|X\| \, \|Y\| \cos\theta$

Thus,

$$\theta = \arccos \mathrm{Cor}[X, Y]. \tag{1.42}$$

For example, if X and Y are uncorrelated, then they are orthogonal in the inner product space. If $\mathrm{Cor}[X, Y] = -1$ then $\theta \equiv \pi$ (that is, X and Y "point in opposite directions"), whereas if $\mathrm{Cor}[X, Y] = 1$ then $\theta \equiv 0$ (that is, X and Y "point in the same direction").

The variance is a measure of uncertainty about the value of a random vector. Given the random vector $\mathbf{X}$ in $\mathbb{R}^n$, its *variance* is defined as

$$\begin{aligned}
\mathrm{Var}[\mathbf{X}] &\doteq \mathrm{Cov}[\mathbf{X}, \mathbf{X}] && (1.43)\\
&= \mathbb{E}\left[(\mathbf{X} - \mathbb{E}[\mathbf{X}])(\mathbf{X} - \mathbb{E}[\mathbf{X}])^\top\right] && (1.44)\\
&= \mathbb{E}\left[\mathbf{X}\mathbf{X}^\top\right] - \mathbb{E}[\mathbf{X}] \cdot \mathbb{E}[\mathbf{X}]^\top && (1.45)\\
&= \begin{bmatrix} \mathrm{Cov}[X_1, X_1] & \cdots & \mathrm{Cov}[X_1, X_n] \\ \vdots & \ddots & \vdots \\ \mathrm{Cov}[X_n, X_1] & \cdots & \mathrm{Cov}[X_n, X_n] \end{bmatrix}. && (1.46)
\end{aligned}$$

The variance of a random vector $\mathbf{X}$ is also called the *covariance matrix* of $\mathbf{X}$ and denoted by $\boldsymbol{\Sigma}_\mathbf{X}$ (or $\boldsymbol{\Sigma}$ if the correspondence to $\mathbf{X}$ is clear from context). A covariance matrix is symmetric by definition due to the symmetry of covariance.

Exercise 1.28: Covariance matrices are positive semi-definite

Prove that a covariance matrix $\boldsymbol{\Sigma}$ is always positive semi-definite. That is, all of its eigenvalues are greater or equal to zero, or equivalently, $\boldsymbol{x}^\top \boldsymbol{\Sigma} \boldsymbol{x} \geq 0$ for any $\boldsymbol{x} \in \mathbb{R}^n$.

▷ *Solution*

Two useful properties of variance are the following:

- It follows from eq. (1.38) that for any linear map $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and vector $\boldsymbol{b} \in \mathbb{R}^m$,

$$\mathrm{Var}[\boldsymbol{A}\mathbf{X} + \boldsymbol{b}] = \boldsymbol{A}\mathrm{Var}[\mathbf{X}]\boldsymbol{A}^\top. \tag{1.47}$$

  In particular, $\mathrm{Var}[-\mathbf{X}] = \mathrm{Var}[\mathbf{X}]$.
- It follows from the definition of variance (1.44) that for any two random vectors $\mathbf{X}$ and $\mathbf{Y}$,

$$\mathrm{Var}[\mathbf{X} + \mathbf{Y}] = \mathrm{Var}[\mathbf{X}] + \mathrm{Var}[\mathbf{Y}] + 2\mathrm{Cov}[\mathbf{X}, \mathbf{Y}]. \tag{1.48}$$

  In particular, if $\mathbf{X}$ and $\mathbf{Y}$ are independent then the covariance term vanishes and $\mathrm{Var}[\mathbf{X} + \mathbf{Y}] = \mathrm{Var}[\mathbf{X}] + \mathrm{Var}[\mathbf{Y}]$.

Analogously to conditional probability and conditional expectation, we can also define conditional variance. The *conditional variance* of a random vector $\mathbf{X}$ given another random vector $\mathbf{Y}$ is

$$\mathrm{Var}[\mathbf{X} \mid \mathbf{Y}] \doteq \mathbb{E}\left[(\mathbf{X} - \mathbb{E}[\mathbf{X} \mid \mathbf{Y}])(\mathbf{X} - \mathbb{E}[\mathbf{X} \mid \mathbf{Y}])^\top \;\middle|\; \mathbf{Y}\right]. \tag{1.49}$$

Intuitively, the conditional variance is the remaining variance when we use $\mathbb{E}[\mathbf{X} \mid \mathbf{Y}]$ to predict $\mathbf{X}$ rather than if we used $\mathbb{E}[\mathbf{X}]$.

Theorem 1.29 (Law of total variance, LOTV)**.**

$$\mathrm{Var}[\mathbf{X}] = \mathbb{E}[\mathrm{Var}[\mathbf{X} \mid \mathbf{Y}]] + \mathrm{Var}[\mathbb{E}[\mathbf{X} \mid \mathbf{Y}]]. \tag{1.50}$$

Here, the first term measures the average deviation from the mean of $\mathbf{X}$ across realizations of $\mathbf{Y}$ and the second term measures the uncertainty in the mean of $\mathbf{X}$ across realizations of $\mathbf{Y}$. In section 2.3, we will see that both terms have a meaningful characterization in the context of Bayesian learning.

Proof. To simplify the notation, we present only a proof for the univariate setting.[15]

[15] In the *univariate* setting (as opposed to the *multivariate* setting) we consider a single random variable.

$$\begin{aligned}
\mathrm{Var}[X] &= \mathbb{E}\Big[X^2\Big] - \mathbb{E}[X]^2 \\
&= \mathbb{E}\Big[\mathbb{E}\Big[X^2 \mid Y\Big]\Big] - \mathbb{E}[\mathbb{E}[X \mid Y]]^2 && \text{by the tower rule (1.33)} \\
&= \mathbb{E}\Big[\mathrm{Var}[X \mid Y] + \mathbb{E}[X \mid Y]^2\Big] - \mathbb{E}[\mathbb{E}[X \mid Y]]^2 && \text{by the definition of variance (1.45)} \\
&= \mathbb{E}[\mathrm{Var}[X \mid Y]] + \Big(\mathbb{E}\Big[\mathbb{E}[X \mid Y]^2\Big] - \mathbb{E}[\mathbb{E}[X \mid Y]]^2\Big) \\
&= \mathbb{E}[\mathrm{Var}[X \mid Y]] + \mathrm{Var}[\mathbb{E}[X \mid Y]]. && \text{by the definition of variance (1.45)}
\end{aligned}$$

□

1.1.12 *Change of Variables*

It is often useful to understand the distribution of a transformed random variable $Y = g(X)$ that is defined in terms of a random variable X, whose distribution is known. Let us first consider the univariate setting. We would like to express the distribution of Y in terms of the distribution of X, that is, we would like to find

$$P_Y(y) = \mathbb{P}(Y \leq y) = \mathbb{P}(g(X) \leq y) = \mathbb{P}\Big(X \leq g^{-1}(y)\Big). \tag{1.51}$$

When the random variables are continuous, this probability can be expressed as an integration over the domain of X. We can then use the substitution rule of integration to "change the variables" to an integration over the domain of Y. Taking the derivative yields the density p_Y.[16]

[16] The full proof of the change of variables formula in the univariate setting can be found in section 6.7.2 of "Mathematics for machine learning" (Deisenroth et al., 2020).

There is an analogous change of variables formula for the multivariate setting.

Fact 1.30 (Change of variables formula)**.** *Let $\mathbf{X}$ be a random vector in $\mathbb{R}^n$ with density $p_{\mathbf{X}}$ and let $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^n$ be a differentiable and invertible function. Then $\mathbf{Y} = \boldsymbol{g}(\mathbf{X})$ is another random variable, whose*

density can be computed based on $p_{\mathbf{X}}$ and $\boldsymbol{g}$ as follows:

$$p_{\mathbf{Y}}(\boldsymbol{y}) = p_{\mathbf{X}}(\boldsymbol{g}^{-1}(\boldsymbol{y})) \cdot \left|\det\left(\boldsymbol{D}\boldsymbol{g}^{-1}(\boldsymbol{y})\right)\right| \tag{1.52}$$

where $\boldsymbol{D}\boldsymbol{g}^{-1}(\boldsymbol{y})$ is the Jacobian of $\boldsymbol{g}^{-1}$ evaluated at $\boldsymbol{y}$.

Here, the term $\left|\det\left(\boldsymbol{D}\boldsymbol{g}^{-1}(\boldsymbol{y})\right)\right|$ measures how much a unit volume changes when applying $\boldsymbol{g}$. Intuitively, this swaps the coordinate system over which we integrate. The factor $\left|\det\left(\boldsymbol{D}\boldsymbol{g}^{-1}(\boldsymbol{y})\right)\right|$ corrects for the change in volume that is caused by this change in coordinates.

Intuitively, you can think of the vector field $\boldsymbol{g}$ as a perturbation to $\mathbf{X}$, "pushing" the probability mass around. The perturbation of a density $p_{\mathbf{X}}$ by $\boldsymbol{g}$ is commonly denoted by the *pushforward measure*

$$\boldsymbol{g}_\sharp p_{\mathbf{X}} \doteq p_{\mathbf{Y}} \quad \text{where } \mathbf{Y} = \boldsymbol{g}(\mathbf{X}). \tag{1.55}$$

1.2 Supervised Learning

In *supervised learning*, we want to learn a function $f : \mathcal{X} \to \mathcal{Y}$ from labeled training data. That is, we are given a collection of labeled examples, $\mathcal{D} \doteq \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathcal{X}$ are *inputs* and the $y_i \in \mathcal{Y}$ are *labels*, and we want to find a function $\hat{f}$ that best-approximates f. It is common to consider a class of functions $\mathcal{F}$ (called the *function class*) that serve as candidates for $\hat{f}$, where each function is described by some parameters $\boldsymbol{\theta}$.

It is often assumed that the observations are noisy, that is,

$$y_i = f(\boldsymbol{x}_i) + \epsilon_i(\boldsymbol{x}_i), \tag{1.56}$$

where $\epsilon_i(\boldsymbol{x}_i)$ is some independent zero-mean noise, for example Gaussian. Often, the dependence of the noise on $\boldsymbol{x}_i$ is omitted for brevity. When the noise distribution may depend on $\boldsymbol{x}_i$, the noise is said to be *heteroscedastic* and otherwise the noise is called *homoscedastic*. We will describe this model in greater detail when introducing Bayesian linear regression in chapter 2.

Typically, we differentiate between the task of *regression* where $\mathcal{Y} \doteq \mathbb{R}^k$ (often, we learn a scalar label, so $k = 1$), and the task of *classification* where $\mathcal{Y} \doteq \mathcal{C}$ and $\mathcal{C}$ is an m-element set of classes. In other words, regression is the task of predicting a continuous label, whereas classification is the task of predicting a discrete class label. These two tasks are intimately related: in fact, we can think of classification tasks as a regression problem where we learn a probability distribution over class labels. In this regression problem, $\mathcal{Y} \doteq \Delta^{\mathcal{C}}$ where we defined $\Delta^{\mathcal{C}}$

Jacobian Given a vector-valued function,

$$\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^m, \quad \boldsymbol{x} \mapsto \begin{bmatrix} g_1(\boldsymbol{x}) \\ \vdots \\ g_m(\boldsymbol{x}) \end{bmatrix},$$

where $g_i : \mathbb{R}^n \to \mathbb{R}$, the *Jacobian* of $\boldsymbol{g}$ at $\boldsymbol{x} \in \mathbb{R}^n$ is

$$\boldsymbol{D}\boldsymbol{g}(\boldsymbol{x}) \doteq \begin{bmatrix} \frac{\partial g_1(\boldsymbol{x})}{\partial \boldsymbol{x}(1)} & \cdots & \frac{\partial g_1(\boldsymbol{x})}{\partial \boldsymbol{x}(n)} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_m(\boldsymbol{x})}{\partial \boldsymbol{x}(1)} & \cdots & \frac{\partial g_m(\boldsymbol{x})}{\partial \boldsymbol{x}(n)} \end{bmatrix}. \tag{1.53}$$

Observe that for a function $f : \mathbb{R}^n \to \mathbb{R}$,

$$\boldsymbol{D}f(\boldsymbol{x}) = \boldsymbol{\nabla} f(\boldsymbol{x})^\top. \tag{1.54}$$

as the set of all probability distributions over the set of classes $\mathcal{C}$. Recall from remark 1.10 that $\Delta^{\mathcal{C}}$ is an $(m-1)$-dimensional convex polytope in the space of probabilities $[0,1]^m$.

1.2.1 Population Risk and Empirical Risk

We denote the underlying (and unknown) joint distribution over input-label pairs $(\boldsymbol{x}, y)$ by $\mathcal{P}$. The notion of "approximation error" is typically captured by a loss function $\ell(\hat{y}; y) \in \mathbb{R}$ which is small when the prediction $\hat{y}$ is "close" to the true label y and large otherwise. As our objective is to best-approximate the mappings $(\boldsymbol{x}, y) \sim \mathcal{P}$, we aim to minimize

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{P}}\Big[\ell(\hat{f}(\boldsymbol{x}); y)\Big]. \tag{1.57}$$

This quantity is also called the *population risk*. However, the underlying distribution $\mathcal{P}$ is unknown to us. All that we can work with is the training data for which we assume $\mathcal{D} \overset{\text{iid}}{\sim} \mathcal{P}$. It is therefore natural to consider minimizing

$$\frac{1}{n}\sum_{i=1}^{n} \ell(\hat{f}(\boldsymbol{x}_i); y_i), \quad \mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n, \tag{1.58}$$

which is known as the *empirical risk*.

However, selecting $\hat{f}$ by minimizing the empirical risk can be problematic. The reason is that in this case the model $\hat{f}$ and the empirical risk depend on the same data $\mathcal{D}$, implying that the empirical risk will not be an unbiased estimator of the population risk.[17] This can result in a model which fits the training data too closely (called *overfitting*), and which is failing to generalize to unseen data. We will discuss some solutions to this problem in section 4.4 when covering model selection in the context of Gaussian processes.

[17] In section 1.4.1, we discuss estimator bias in detail.

1.3 Bayesian Learning and Inference

Bayesian learning is the process of updating a Bayesian prior $\mathbb{P}(A)$ to a Bayesian posterior $\mathbb{P}(A \mid B)$ upon observing B. Thereby, we typically replace A with the parameterization of a model $\boldsymbol{\theta}$, and B with labeled training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$.

At this point, it is helpful to differentiate between learning and inference. By *learning* (or estimation) we refer to the aforementioned process of learning a model from data. In contrast, by *inference* (or prediction) we refer to the process of using our learned model to predict labels at new inputs.[18]

[18] In the literature, these terms are sometimes used differently, where our learning is referred to as "inference" and our inference is referred to as "prediction".

At the core of Bayesian learning is Bayes' rule.

Theorem 1.31 (Bayes' rule)**.** *Given random vectors* $\mathbf{X}$ *in* $\mathbb{R}^n$ *and* $\mathbf{Y}$ *in* $\mathbb{R}^m$*, we have for any* $\boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{y} \in \mathbb{R}^m$*,*

$$p(\boldsymbol{x} \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{x}) p(\boldsymbol{x})}{p(\boldsymbol{y})}. \tag{1.59}$$

Proof. Bayes' rule is a direct consequence of the definition of conditional densities (1.13) and the product rule (1.14). □

Remark 1.32: Notation of prior and posterior

We generally represent the prior distribution by p and the posterior distribution by $p(\cdot \mid \boldsymbol{y})$. Note that sometimes, to highlight their distinction, $p(\cdot)$ is used to refer to the prior.

It is useful to consider the meaning of each term separately:

- $p(\boldsymbol{x} \mid \boldsymbol{y})$ (*posterior*) is the updated belief about $\boldsymbol{x}$ after observing $\boldsymbol{y}$,
- $p(\boldsymbol{x})$ (*prior*) is the initial belief about $\boldsymbol{x}$,
- $p(\boldsymbol{y} \mid \boldsymbol{x})$ (*(conditional) likelihood*) describes how likely the observations $\boldsymbol{y}$ are under a given value $\boldsymbol{x}$,
- $p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{y} \mid \boldsymbol{x}) p(\boldsymbol{x})$ (*joint likelihood* or *generative model*) combines prior and likelihood,
- $p(\boldsymbol{y})$ (*marginal likelihood*) describes how likely the observations $\boldsymbol{y}$ are across all values of $\boldsymbol{x}$.

The marginal likelihood can be computed using the law of total probability (1.15),

$$p(\boldsymbol{y}) = \int_{\mathbf{X}(\Omega)} p(\boldsymbol{y} \mid \boldsymbol{x}) p(\boldsymbol{x}) \, d\boldsymbol{x}, \tag{1.60}$$

or the sum rule. Note, however, that as the marginal likelihood does not depend on $\boldsymbol{x}$,

$$p(\boldsymbol{x} \mid \boldsymbol{y}) \propto p(\boldsymbol{y} \mid \boldsymbol{x}) p(\boldsymbol{x}), \tag{1.61}$$

so we do not need to compute it to optimize for $\boldsymbol{x}$ (e.g., find the gradient with respect to $\boldsymbol{x}$). This is a helpful fact, which we will make use of quite often.

Exercise 1.33: Bayes' rule

As a result of a medical screening, one of the tests revealed a serious disease in a person. The test has a high accuracy of 99% (the probability of a positive response in the presence of a disease is 99% and the probability of a negative response in the absence of a disease is also 99%). However, the disease is quite rare and

occurs only in one person per 10 000. Calculate the probability of the examined person having the identified disease.

▷ *Solution*

Remark 1.34: Improper priors

Not always is it required that the prior $p(\boldsymbol{x})$ is a valid distribution (i.e., integrates to 1). Consider for example, the "uniform prior" $p(x) \propto \mathbb{1}\{x \in I\}$ where $I \subseteq \mathbb{R}$ is an infinitely large interval. A prior which is not a valid distribution is called an *improper prior*. We can still derive meaning from the posterior of a given likelihood and (improper) prior as long as the posterior is a valid distribution.

1.3.1 *Using Bayes' rule for supervised learning*

Returning to the task of learning a model from training data, we interpret $p(\boldsymbol{\theta})$ as the degree of our belief that the model parameterized by $\boldsymbol{\theta}$ "describes the data best". The likelihood describes how likely the training data is under a particular model. Often, we assume that our data was sampled independently, allowing us to write the likelihood as a product:

$$p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta}) = \prod_{i=1}^{n} p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}). \tag{1.62}$$

The posterior represents our belief about the best model after seeing the training data. Using Bayes' rule, we can write it as[19]

$$p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}, y_{1:n}) = \frac{1}{Z} p(\boldsymbol{\theta}) \prod_{i=1}^{n} p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}) \quad \text{where} \tag{1.63}$$

$$Z \doteq \int p(\boldsymbol{\theta}) \prod_{i=1}^{n} p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}) \, d\boldsymbol{\theta} \tag{1.64}$$

and again assuming that the training data was sampled independently. We often refer to Z as the *normalizing constant*.

[19] We generally assume that

$$p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}) = p(\boldsymbol{\theta}).$$

For our purposes, you can think of the inputs $\boldsymbol{x}_{1:n}$ as fixed deterministic parameters.

Finally, we can use our learned model for inference (predictions) at a new input $\boldsymbol{x}^\star$ by conditioning on $\boldsymbol{\theta}$,

$$\begin{aligned} p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, y_{1:n}) &= \int p(y^\star, \boldsymbol{\theta} \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, y_{1:n}) \, d\boldsymbol{\theta} && \text{by the sum rule (1.10)} \\ &= \int p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}, y_{1:n}) \, d\boldsymbol{\theta}. && \text{by the product rule (1.14) and } y^\star \perp \boldsymbol{x}_{1:n}, y_{1:n} \mid \boldsymbol{\theta} \end{aligned} \tag{1.65}$$

Here, the distribution over models $p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}, y_{1:n})$ is called the *posterior* and the distribution over predictions $p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, y_{1:n})$ is called the *predictive posterior*.

1.3.2 *Conjugate Priors*

In general, the integrals in the normalizing constant Z (1.64) and the predictive distribution (1.65) cannot be expressed in closed-form.

If the prior $p(\boldsymbol{x})$ and posterior $p(\boldsymbol{x} \mid \boldsymbol{y})$ are of the same family of distributions, the prior is called a *conjugate prior* to the likelihood $p(\boldsymbol{y} \mid \boldsymbol{x})$. This is a very desirable property, as it allows us to apply the same learning mechanism recursively.

Under some conditions the Gaussian is self-conjugate (cf. section 2.2). That is, if we have a Gaussian prior and a Gaussian likelihood, then our posterior will also be a Gaussian.

Example 1.35: Bernoulli and beta distribution

As an example for conjugacy, we will show that the beta distribution is a conjugate prior to a binomial likelihood. Recall the PMF of the binomial distribution

$$\mathrm{Bin}(k; n, \theta) = \binom{n}{k} \theta^k (1-\theta)^{n-k} \tag{1.66}$$

and the PDF of the beta distribution,

$$\mathrm{Beta}(\theta; \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{\alpha-1}(1-\theta)^{\beta-1}, \tag{1.67}$$

where Γ denotes the gamma function. We assume $\theta \sim \mathrm{Beta}(\alpha, \beta)$ and $k \mid \theta \sim \mathrm{Bin}(n, \theta)$. Let $n_H = k$ be the number of heads and $n_T = n - k$ the number of tails in the binomial trial k. Now,

$$\begin{aligned} p(\theta \mid k) &\propto p(k \mid \theta) p(\theta) && \text{using Bayes' rule (1.59)} \\ &\propto \theta^{n_H}(1-\theta)^{n_T} \theta^{\alpha-1}(1-\theta)^{\beta-1} \\ &= \theta^{\alpha+n_H-1}(1-\theta)^{\beta+n_T-1}. \end{aligned}$$

Thus, $\theta \mid k \sim \mathrm{Beta}(\alpha + n_H, \beta + n_T)$.

1.4 *Parameter Estimation*

A very common — albeit non-Bayesian — approach to learning is to reduce the posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}, y_{1:n})$ to a point estimate of $\boldsymbol{\theta}$. We will now take a brief look at this approach, but first define the concept of an estimator more formally.

1.4.1 Estimators

Suppose we are given a collection of independent samples $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(n)}$ from some random vector $\mathbf{X}$. Often, the exact distribution of $\mathbf{X}$ is unknown to us, but we still want to "estimate" some property of this distribution, for example its mean. We denote the property that we aim to estimate from our sample by $\boldsymbol{\theta}$. For example, if our goal is estimating the mean of $\mathbf{X}$, then $\boldsymbol{\theta} \doteq \mathbb{E}[\mathbf{X}]$.

An *estimator* for a parameter $\boldsymbol{\theta}$ is a random vector $\mathbf{U}$ that is a function of n sample variables $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(n)}$ whose distribution is identical to the distribution of $\mathbf{X}$. Any concrete sample $\boldsymbol{x}^{(i)} \sim \mathbf{X}^{(i)}$ yields a concrete estimate $\boldsymbol{u} \sim \mathbf{U}$ of $\boldsymbol{\theta}$. How these samples are composed is not restricted, but their composition should lead to a "good estimate" of $\boldsymbol{\theta}$. Mainly, there are two measures of goodness of an estimator: its expectation and its variance.

Clearly, we want $\mathbb{E}[\mathbf{U}] = \boldsymbol{\theta}$. Estimators that satisfy this property are called *unbiased*. The *bias*, $\mathbb{E}_{\mathbf{U}}[\mathbf{U} - \boldsymbol{\theta}]$, of an unbiased estimator is $\mathbf{0}$.

Example 1.36: Estimating expectations

Perhaps the most common estimator is the *sample mean* $\overline{\mathbf{X}}_n$, which is an unbiased estimator for $\mathbb{E}[\mathbf{X}]$. It is defined as

$$\overline{\mathbf{X}}_n \doteq \frac{1}{n} \sum_{i=1}^{n} \mathbf{X}^{(i)} \tag{1.68}$$

where $\mathbf{X}^{(i)} \overset{\text{iid}}{\sim} \mathbf{X}$ are independent samples of the random vector $\mathbf{X}$.

Using a sample mean to estimate an expectation is often also called *Monte Carlo sampling*, and the resulting estimate is referred to as a *Monte Carlo average*. In section 1.4.2, we will see that the variance of the sample mean is small for reasonably large n.

Example 1.37: Estimating variances

Similarly to the sample mean, the *sample variance* (or *sample covariance matrix*) of a random vector $\mathbf{X}$,

$$S_n^2 \doteq \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{X}^{(i)} - \overline{\mathbf{X}}_n)(\mathbf{X}^{(i)} - \overline{\mathbf{X}}_n)^\top \tag{1.69}$$

$$= \frac{n}{n-1} \left(\frac{1}{n} \sum_{i=1}^{n} \mathbf{X}^{(i)} {\mathbf{X}^{(i)}}^\top - \overline{\mathbf{X}}_n \overline{\mathbf{X}}_n^\top \right) \tag{1.70}$$

where $\mathbf{X}^{(i)} \overset{\text{iid}}{\sim} \mathbf{X}$ are independent samples of $\mathbf{X}$, is an unbiased

estimator of $\mathrm{Var}[\mathbf{X}]$.

Intuitively, the normalizing factor is $1/n-1$ because S_n^2 depends on the sample mean $\overline{\mathbf{X}}_n$, which is obtained using the same samples. For this reason, S_n^2 has $n-1$ degrees of freedom.[20]

[20] That is, any sample $\mathbf{X}^{(i)}$ can be recovered using all other samples and the sample mean.

Exercise 1.38: Sample variance

Let X be a zero-mean random variable. Confirm that the sample variance of X is indeed unbiased.

▷ *Solution*

The second desirable property of an estimator U is that its variance is small.[21] A common measure for the variance of an estimator of θ is the *mean squared error*,

[21] The variance of estimators $\mathbf{U}$ is typically studied component wise.

$$\mathrm{MSE}(U) \doteq \mathbb{E}_U\Big[(U-\theta)^2\Big]. \tag{1.71}$$

We say that an estimator is *consistent* if its mean squared error converges to zero as $n \to \infty$.

Remark 1.39: The bias-variance tradeoff

Often, bias and variance of an estimator have to be traded as it is difficult to achieve both a small bias and a small variance. This is known as the *bias-variance tradeoff*.

There exists a natural characterization of the mean squared error in terms of bias and variance that highlights this tradeoff. Namely,

$$\mathrm{MSE}(U) = \mathbb{E}_U\Big[(U-\theta)^2\Big] = \mathrm{Var}[U] + \mathbb{E}_U[U-\theta]^2, \tag{1.72}$$

using (1.45) and $\mathrm{Var}_U[U-\theta] = \mathrm{Var}[U]$

the mean squared error of an estimator can be written as the sum of its variance and its squared bias.

We further say that an estimator U is *sharply concentrated* around θ if for any $\epsilon > 0$,

$$\mathbb{P}(|U-\theta| > \epsilon) \leq \exp(-\Omega(\epsilon)) \doteq \delta, \tag{1.73}$$

where $\Omega(\epsilon)$ denotes the class of functions that grow at least linearly in ϵ.[22] Thus, if an estimator is sharply concentrated, its absolute error is bounded by an exponentially quickly decaying error probability δ as ϵ grows.

[22] That is, $h \in \Omega(\epsilon)$ iff $\lim_{\epsilon\to\infty} \frac{h(\epsilon)}{\epsilon} > 0$. With slight abuse of notation, we force h to be positive (so as to ensure that the argument to the exponential function is negative) whereas in the traditional definition of Landau symbols, h is only required to grow linearly in absolute value.

Remark 1.40: Heavy tails

It is often said that a sharply concentrated estimator U has *small tails*, where "tails" refer to the "ends" of a PDF. Let us examine the difference between a *light-tailed* and a *heavy-tailed* distribution more closely.

Definition 1.41. A distribution P_X is said to have a *heavy (right) tail* iff its *tail distribution function*

$$\overline{P}_X(x) \doteq 1 - P_X(x) = \mathbb{P}(X > x) \tag{1.74}$$

decays slower than that of the exponential distribution, that is,

$$\limsup_{x\to\infty} \frac{\overline{P}_X(x)}{e^{-\lambda x}} = \infty \tag{1.75}$$

for all $\lambda > 0$. When $\limsup_{x\to\infty} \frac{\overline{P}_X(x)}{\overline{P}_Y(x)} > 1$, the (right) tail of X is said to be *heavier* than the (right) tail of Y.

It is immediate from the definitions that the distribution of an unbiased estimator is light-tailed if and only if the estimator is sharply concentrated, so both notions are equivalent.

A Gaussian $X \sim \mathcal{N}(0,1)$ is light-tailed since its tail distribution function is bounded by

$$\overline{P}_X(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-t^2/2}\, dt \le \int_x^\infty \frac{t}{x}\frac{1}{\sqrt{2\pi}} e^{-t^2/2}\, dt = \frac{e^{-x^2/2}}{\sqrt{2\pi}x}. \tag{1.76}$$

using $\frac{t}{x} \ge 1$

Another light-tailed distribution is the *Laplace distribution*

$$\text{Laplace}(x;\mu,h) \propto \exp\left(-\frac{|x-\mu|}{h}\right) \tag{1.77}$$

with mean μ and so-called length scale parameter h. It places more probability mass directly at the mean at the expense of heavier tails. An example of a heavy-tailed distribution is the *log-normal distribution*. A random variable Y is logarithmically normal distributed with parameters μ and σ^2 iff $\log Y \sim \mathcal{N}(\mu, \sigma^2)$.

For more details and examples of heavy-tailed distributions, refer to "The Fundamentals of Heavy Tails" (Nair et al., 2022).

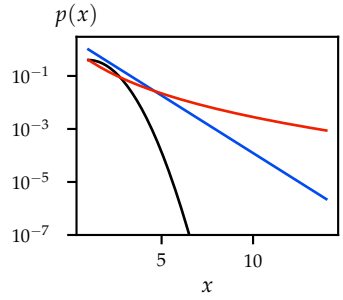


Figure 1.5: Shown are the right tails of the PDFs of a **Gaussian** with mean 1 and variance 1, a **exponential distribution** with mean 1 and parameter $\lambda = 1$, and a **log-normal distribution** with mean 1 and variance 1 on a log-scale.

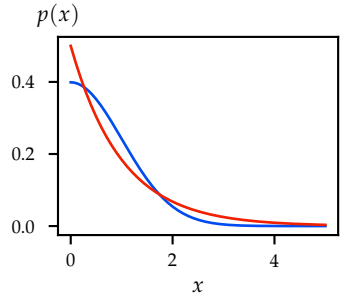


Figure 1.6: Shown are the right tails of the PDFs of a **Laplace distribution** with mean 0 and length scale 1 and a **Gaussian** with mean 0 and variance 1.

1.4.2 *Mean Estimation and Concentration Inequalities*

We have seen that we desire two properties in estimators: namely, (1) that they are unbiased; and (2) that their variance is small.[23] For

[23] That is, they are consistent and, ideally, their variance converges quickly.

estimating expectations with the sample mean $\overline{X}_n$ which is also known as a *Monte Carlo approximation*, we will see that both properties follow from standard results in probability theory.

It is immediate from the definition of the sample mean (1.68) that it is an unbiased estimator for $\mathbb{E}[X]$. We will now see that the sample mean is also consistent, and even better, converges with exponentially decaying error probability.

First, let us recall the common notions of convergence of a sequence of random variables X_n.

Definition 1.42 (Convergence of random variables)**.** Let $\{X_n\}_{n \in \mathbb{N}}$ be a sequence of random variables and X another random variable. We say that,

1. X_n converges to X *almost surely* (also called *convergence with probability* 1) if
$$\mathbb{P}\Big(\Big\{\omega \in \Omega : \lim_{n \to \infty} X_n(\omega) = X(\omega)\Big\}\Big) = 1, \tag{1.78}$$
and we write $X_n \xrightarrow{\text{a.s.}} X$ as $n \to \infty$.
2. X_n converges to X *in probability* if for any $\epsilon > 0$,
$$\lim_{n \to \infty} \mathbb{P}(|X_n - X| > \epsilon) = 0, \tag{1.79}$$
and we write $X_n \xrightarrow{\mathbb{P}} X$ as $n \to \infty$.
3. X_n converges to X *in distribution* if for all points $x \in X(\Omega)$ at which P_X is continuous,
$$\lim_{n \to \infty} P_{X_n}(x) = P_X(x), \tag{1.80}$$
and we write $X_n \xrightarrow{\mathcal{D}} X$ as $n \to \infty$.

It can be shown that as $n \to \infty$,

$$X_n \xrightarrow{\text{a.s.}} X \implies X_n \xrightarrow{\mathbb{P}} X \implies X_n \xrightarrow{\mathcal{D}} X. \tag{1.81}$$

In the following, we will use some classical concentration inequalities.

Exercise 1.43: Simple concentration inequalities

1. Prove *Markov's inequality* which says that if X is a non-negative random variable, then for any $\epsilon > 0$,
$$\mathbb{P}(X \geq \epsilon) \leq \frac{\mathbb{E}X}{\epsilon}. \tag{1.82}$$
2. Prove *Chebyshev's inequality* which says that if X is a random

variable with finite and non-zero variance, then for any $\epsilon > 0$,

$$\mathbb{P}(|X - \mathbb{E}X| \geq \epsilon) \leq \frac{\mathrm{Var}X}{\epsilon^2}. \tag{1.83}$$

▷ *Solution*

The law of large numbers is a classical result in statistics and directly implies that the sample mean is consistent.

Fact 1.44 (Strong law of large numbers, SLLN)**.** *Given the random variable $X : \Omega \to \mathbb{R}$ with independent samples $x_i \sim X$ and finite variance. Let f be a "nice" function and $Y \doteq f(X)$. We have,*

$$\overline{Y}_n = \frac{1}{n}\sum_{i=1}^{n} f(x_i) \overset{\text{a.s.}}{\to} \mathbb{E}[f(X)], \tag{1.84}$$

as $n \to \infty$.

Exercise 1.45: Weak law of large numbers

Prove this weaker version of the SLLN which is known as the *weak law of large numbers* (WLLN).

Let $X : \Omega \to \mathbb{R}$ be a random variable with finite variance. Then,

$$\overline{X}_n \overset{\mathbb{P}}{\to} \mathbb{E}X. \tag{1.85}$$

▷ *Solution*

Under more restrictive assumptions it is even possible to show that the sample mean $\overline{X}_n$ is sharply concentrated around the mean $\mathbb{E}X$, which is a much stronger property than consistency. One such assumption is provided by the notion of a sub-Gaussian random variable.

Definition 1.46 (Sub-Gaussian random variable)**.** A random variable $X : \Omega \to \mathbb{R}$ is called *σ-sub-Gaussian* for $\sigma > 0$ if for all $\lambda \in \mathbb{R}$,

$$\mathbb{E}\left[e^{\lambda(X - \mathbb{E}X)}\right] \leq \exp\left(\frac{\sigma^2\lambda^2}{2}\right). \tag{1.86}$$

Intuitively, a random variable is sub-Gaussian if its tail probabilities decay at least as fast as those of a Gaussian. The class of sub-Gaussian random variables is therefore a subclass of the light-tailed distributions discussed in remark 1.40.

Example 1.47: Examples of sub-Gaussian random variables

- If a random variable X is σ-sub-Gaussian, then so is $-X$.
- If $X \sim \mathcal{N}(\mu, \sigma^2)$ then a simple calculation shows that

$$\varphi_X(\lambda) \doteq \mathbb{E}\left[e^{\lambda X}\right] = \exp\left(\mu\lambda + \frac{\sigma^2\lambda^2}{2}\right) \qquad \text{for any } \lambda \in \mathbb{R} \tag{1.87}$$

  which is called the "moment-generating function" of the normal distribution, and implying that X is σ-sub-Gaussian.
- If the random variable $X : \Omega \to \mathbb{R}$ satisfies $a \leq X \leq b$ with probability 1 then X is $\frac{b-a}{2}$-sub-Gaussian. This is also known as *Hoeffding's lemma*.

Theorem 1.48 (Hoeffding's inequality)**.** *Let $X : \Omega \to \mathbb{R}$ be a σ-sub-Gaussian random variable. Then,*

$$\mathbb{P}\left(\left|\overline{X}_n - \mathbb{E}X\right| \geq \epsilon\right) \leq 2\exp\left(-\frac{n\epsilon^2}{2\sigma^2}\right). \tag{1.88}$$

In words, the absolute error of the expectation estimate is bounded by an exponentially quickly decaying error probability δ. Solving for n, we obtain that for

$$n \geq \frac{2\sigma^2}{\epsilon^2}\log\frac{2}{\delta} \tag{1.89}$$

the probability that the absolute error is greater than ϵ is at most δ.

Proof. Let $S_n \doteq n\overline{X}_n = X_1 + \cdots + X_n$. We have for any $\lambda, \epsilon > 0$ that

$$\begin{aligned}
\mathbb{P}\left(\overline{X}_n - \mathbb{E}X \geq \epsilon\right) &= \mathbb{P}(S_n - \mathbb{E}S_n \geq n\epsilon) \\
&= \mathbb{P}\left(e^{\lambda(S_n - \mathbb{E}S_n)} \geq e^{n\epsilon\lambda}\right) && \text{using that } z \mapsto e^{\lambda z} \text{ is increasing} \\
&\leq e^{-n\epsilon\lambda}\mathbb{E}\left[e^{\lambda(S_n - \mathbb{E}S_n)}\right] && \text{using Markov's inequality (1.82)} \\
&= e^{-n\epsilon\lambda}\prod_{i=1}^{n}\mathbb{E}\left[e^{\lambda(X_i - \mathbb{E}X)}\right] && \text{using independence of the } X_i \\
&\leq e^{-n\epsilon\lambda}\prod_{i=1}^{n} e^{\sigma^2\lambda^2/2} && \text{using the characterizing property of a } \sigma\text{-sub-Gaussian random variable (1.86)} \\
&= \exp\left(-n\epsilon\lambda + \frac{n\sigma^2\lambda^2}{2}\right).
\end{aligned}$$

Minimizing the expression with respect to λ, we set $\lambda = \epsilon/\sigma^2$, and obtain

$$\mathbb{P}\left(\overline{X}_n - \mathbb{E}X \geq \epsilon\right) \leq \min_{\lambda > 0}\left\{\exp\left(-n\epsilon\lambda + \frac{n\sigma^2\lambda^2}{2}\right)\right\} = \exp\left(-\frac{n\epsilon^2}{2\sigma^2}\right).$$

The theorem then follows from

$$\mathbb{P}\big(\big|\overline{X}_n - \mathbb{E}X\big| \geq \epsilon\big) = \mathbb{P}\big(\overline{X}_n - \mathbb{E}X \geq \epsilon\big) + \mathbb{P}\big(\overline{X}_n - \mathbb{E}X \leq -\epsilon\big)$$

and noting that the second term can be bounded analogously to the first term by considering the random variables $-X_1, \ldots, -X_n$. □

The law of large numbers and Hoeffding's inequality show that we can estimate $\mathbb{E}[f(X)]$ very precisely with "few" samples using a sample mean. Crucially, we require that the samples x_i are from the same distribution as the random variable X and that the samples are independent.

1.4.3 *Maximum Likelihood Estimation*

Let us return to finding point estimates of our model. Intuitively, the "best model" should have the property that the training data is as likely as possible under this model when compared to all other models. This is precisely the *maximum likelihood estimate* (or MLE):

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} \doteq \arg\max_{\boldsymbol{\theta}} p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta}) \tag{1.90}$$

$$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}) \qquad \text{using the independence of the training data (1.62)}$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}). \tag{1.91} \qquad \text{using that log is a monotonic function}$$

Taking the logarithm of the likelihood is a standard technique. The resulting $\log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta})$ is called *log-likelihood*, and its negative is known as the *negative log-likelihood* $\ell_{\text{nll}}(\boldsymbol{\theta}; \mathcal{D})$,

$$= \arg\min_{\boldsymbol{\theta}} \ell_{\text{nll}}(\boldsymbol{\theta}; \mathcal{D}). \tag{1.92}$$

Example 1.49: MLE of Bernoulli random variables

Let us consider an i.i.d. sample $x_{1:n}$ of $\text{Bern}(p)$ distributed random variables. We want to estimate the parameter p using a MLE. We have,

$$\hat{p}_{\text{MLE}} = \arg\max_{p} \mathbb{P}(x_{1:n} \mid p)$$

$$= \arg\max_{p} \sum_{i=1}^{n} \log \mathbb{P}(x_i \mid p)$$

$$= \arg\max_{p} \sum_{i=1}^{n} \log p^{x_i}(1-p)^{1-x_i} \qquad \text{using the Bernoulli PMF, see example 1.9}$$

$$= \arg\max_p \sum_{i=1}^{n} x_i \log p + (1 - x_i) \log(1 - p).$$

Computing the first derivative with respect to p, we see that the objective is maximized by

$$= \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{1.93}$$

Thus, the maximum likelihood estimate for the parameter p of $X \sim \text{Bern}(p)$ coincides with the sample mean $\overline{X}$.

1.4.4 *Maximum A Posteriori Estimation*

Often, the MLE overfits to the training data. The danger of overfitting can be reduced by taking a "more Bayesian" approach and maximizing over the entire posterior distribution instead of only maximizing the likelihood. This results in the *maximum a posteriori estimate* (or MAP estimate):

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} \doteq \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}, y_{1:n}) \tag{1.94}$$

$$= \arg\max_{\boldsymbol{\theta}} p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta}) \tag{1.95}$$

by Bayes' rule (1.59)

$$= \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) + \sum_{i=1}^{n} \log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}). \tag{1.96}$$

using that log is a monotonic function

$$= \arg\min_{\boldsymbol{\theta}} \underbrace{-\log p(\boldsymbol{\theta})}_{\text{regularization}} + \underbrace{\ell_{\text{nll}}(\boldsymbol{\theta}; \mathcal{D})}_{\text{quality of fit}}. \tag{1.97}$$

Here, the *log-prior* $\log p(\boldsymbol{\theta})$ acts as a regularizer. For example,

- if $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \sigma_p^2 \boldsymbol{I})$ then $-\log p(\boldsymbol{\theta}) = \frac{\sigma_p^2}{2} \|\boldsymbol{\theta}\|_2^2$,
- if $p(\boldsymbol{\theta}) = \text{Laplace}(\boldsymbol{\theta}; \mathbf{0}, h)$ then $-\log p(\boldsymbol{\theta}) = \frac{h}{2} \|\boldsymbol{\theta}\|_1^2$, and
- for a uniform prior (i.e., a prior that is independent of $\boldsymbol{\theta}$), MAP estimation reduces to likelihood maximization.

Note that in case the posterior is Gaussian, the MAP estimate (i.e., mode of the posterior distribution) corresponds to the mean.

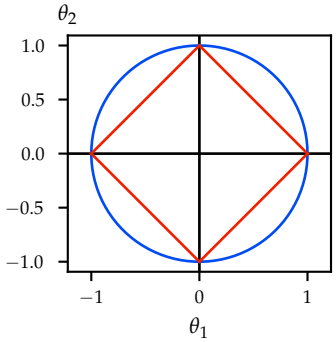


Figure 1.7: Level sets of **Gaussian** and **Laplace** regularization. It can be seen that Laplace regularization is more effective in encouraging sparse solutions (that is, solutions where many components are set to exactly 0).

1.5 *Optimization*

Finding parameter estimates is one of the many examples where we seek to minimize some function ℓ.[24] The field of optimization has a rich history, which we will not explore in much detail here. What will be important for us is that given that the function to be optimized (called the *objective* or *loss*) fulfills certain smoothness properties, optimization is a well-understood problem and can often be solved exactly

[24] W.l.o.g. we assume that we want to minimize ℓ. If we wanted to maximize the objective, we can simply minimize its negation.

(e.g., when the objective is convex) or "approximately" when the objective is non-convex. In fact, we will see that it is often advantageous to frame problems as optimization problems when suitable because the machinery to solve these problems is so extensive.

Readings

For a more thorough reminder of optimization methods, read chapter 7 of "Mathematics for machine learning" (Deisenroth et al., 2020).

1.5.1 *Stationary Points*

In this section, we derive some basic facts about unconstrained optimization problems. Given some function $f : \mathbb{R}^n \to \mathbb{R}$, we want to find

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x}). \tag{1.98}$$

We say that a point $\boldsymbol{x}^\star \in \mathbb{R}^n$ is a *(global) optimum* of f if $f(\boldsymbol{x}^\star) \leq f(\boldsymbol{x})$ for any $\boldsymbol{x} \in \mathbb{R}^n$.

Consider the more general problem of minimizing f over some subset $S \subseteq \mathbb{R}^n$, that is, to minimize the function $f_S : S \to \mathbb{R}, \boldsymbol{x} \mapsto f(\boldsymbol{x})$. If there exists some open subset $S \subseteq \mathbb{R}^n$ including $\boldsymbol{x}^\star$ such that $\boldsymbol{x}^\star$ is optimal with respect to the function f_S, then $\boldsymbol{x}^\star$ is called a *local optimum* of f.

Remark 1.50: Differentiability

We will generally assume that f is continuously (Fréchet) differentiable on $\mathbb{R}^n$. That is, at any point $\boldsymbol{x} \in \mathbb{R}^n$, there exists $\boldsymbol{\nabla} f(\boldsymbol{x})$ such that for any $\boldsymbol{\delta} \in \mathbb{R}^n$,

$$f(\boldsymbol{x} + \boldsymbol{\delta}) = f(\boldsymbol{x}) + \boldsymbol{\nabla} f(\boldsymbol{x})^\top \boldsymbol{\delta} + o(\|\boldsymbol{\delta}\|_2), \tag{1.99}$$

where $\lim_{\boldsymbol{\delta} \to \mathbf{0}} \frac{o(\|\boldsymbol{\delta}\|_2)}{\|\boldsymbol{\delta}\|_2} = 0$, and $\boldsymbol{\nabla} f(\boldsymbol{x})$ is continuous on $\mathbb{R}^n$.

Equation (1.99) is also called a *first-order expansion* of f at $\boldsymbol{x}$.

Definition 1.51 (Stationary point)**.** Given a function $f : \mathbb{R}^n \to \mathbb{R}$, a point $\boldsymbol{x} \in \mathbb{R}^n$ where $\boldsymbol{\nabla} f(\boldsymbol{x}) = \mathbf{0}$ is called a *stationary point* of f.

Being a stationary point is not sufficient for optimality. Take for example the point $x \doteq 0$ of $f(x) \doteq x^3$. Such a point that is stationary but not (locally) optimal is called a *saddle point*.

Theorem 1.52 (First-order optimality condition)**.** *If $\boldsymbol{x} \in \mathbb{R}^n$ is a local extremum of a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, then $\boldsymbol{\nabla} f(\boldsymbol{x}) = \mathbf{0}$.*

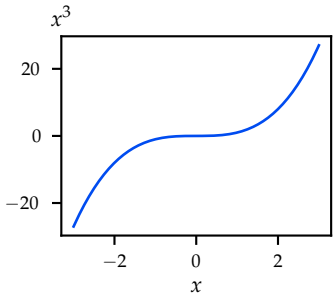


Figure 1.8: Example of a saddle point at $x = 0$.

Proof. Assume $\boldsymbol{x}$ is a local minimum of f. Then, for all $\boldsymbol{d} \in \mathbb{R}^n$ and for all small enough $\lambda \in \mathbb{R}$, we have $f(\boldsymbol{x}) \leq f(\boldsymbol{x} + \lambda \boldsymbol{d})$, so

$$\begin{aligned} 0 &\leq f(\boldsymbol{x} + \lambda \boldsymbol{d}) - f(\boldsymbol{x}) \\ &= \lambda \nabla f(\boldsymbol{x})^\top \boldsymbol{d} + o(\lambda \left\| \boldsymbol{d} \right\|_2). \end{aligned}$$

using a first-order expansion of f around $\boldsymbol{x}$

Dividing by λ and taking the limit $\lambda \to 0$, we obtain

$$0 \leq \nabla f(\boldsymbol{x})^\top \boldsymbol{d} + \lim_{\lambda \to 0} \frac{o(\lambda \left\| \boldsymbol{d} \right\|_2)}{\lambda} = \nabla f(\boldsymbol{x})^\top \boldsymbol{d}.$$

Take $\boldsymbol{d} \doteq -\nabla f(\boldsymbol{x})$. Then, $0 \leq -\left\| \nabla f(\boldsymbol{x}) \right\|_2^2$, so $\nabla f(\boldsymbol{x}) = \boldsymbol{0}$. □

1.5.2 *Convexity*

Convex functions are a subclass of functions where finding global minima is substantially easier than for general functions.

Definition 1.53 (Convex function)**.** A function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* iff

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n : \forall \theta \in [0, 1] : f(\theta \boldsymbol{x} + (1 - \theta)\boldsymbol{y}) \leq \theta f(\boldsymbol{x}) + (1 - \theta) f(\boldsymbol{y}). \tag{1.100}$$

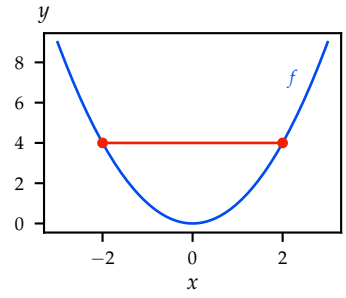


Figure 1.9: Example of a convex function. Any line between two points on f lies "above" f.

That is, any line between two points on f lies "above" f. If the inequality of eq. (1.100) is strict, we say that f is *strictly convex*.

If the function f is convex, we say that the function $-f$ is *concave*. The above is also known as the 0*th-order characterization of convexity*.

Theorem 1.54 (First-order characterization of convexity)**.** *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, then f is convex iff*

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n : f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}). \tag{1.101}$$

Observe that the right-hand side of the inequality is an affine function with slope $\nabla f(\boldsymbol{x})$ based at $f(\boldsymbol{x})$.

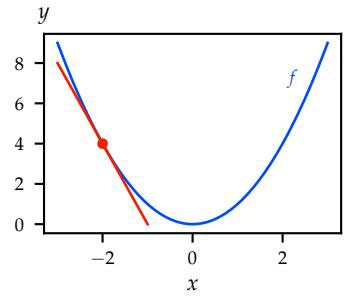


Figure 1.10: The first-order characterization characterizes convexity in terms of affine lower bounds. Shown is an affine lower bound based at $x = -2$.

Proof. In the following, we will make use of directional derivatives.

Exercise 1.55: Directional derivatives

Given a function $f : \mathbb{R}^n \to \mathbb{R}$ that is differentiable at $\boldsymbol{x} \in \mathbb{R}^n$, the *directional derivative* of f at $\boldsymbol{x}$ in the direction $\boldsymbol{d} \in \mathbb{R}^n$ is

$$Df(\boldsymbol{x})[\boldsymbol{d}] \doteq \lim_{\lambda \to 0} \frac{f(\boldsymbol{x} + \lambda \boldsymbol{d}) - f(\boldsymbol{x})}{\lambda}. \tag{1.102}$$

Show that

$$Df(\boldsymbol{x})[\boldsymbol{d}] = \nabla f(\boldsymbol{x})^\top \boldsymbol{d}. \tag{1.103}$$

Hint: Consider a first-order expansion of f at $\boldsymbol{x}$ in direction $\lambda \boldsymbol{d}$.

▷ *Solution*

- "⇒": Fix any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$. As f is convex,

  $$f((1-\theta)\boldsymbol{x} + \theta \boldsymbol{y}) \leq (1-\theta) f(\boldsymbol{x}) + \theta f(\boldsymbol{y}),$$

  for all $\theta \in [0,1]$. We can rearrange to

  $$f(\underbrace{(1-\theta)\boldsymbol{x} + \theta \boldsymbol{y}}_{\boldsymbol{x}+\theta(\boldsymbol{y}-\boldsymbol{x})}) - f(\boldsymbol{x}) \leq \theta (f(\boldsymbol{y}) - f(\boldsymbol{x})).$$

  Dividing by θ yields,

  $$\frac{f(\boldsymbol{x} + \theta(\boldsymbol{y}-\boldsymbol{x})) - f(\boldsymbol{x})}{\theta} \leq f(\boldsymbol{y}) - f(\boldsymbol{x}).$$

  Taking the limit $\theta \to 0$ on both sides gives the directional derivative at $\boldsymbol{x}$ in direction $\boldsymbol{y} - \boldsymbol{x}$,

  $$\boldsymbol{\nabla} f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}) = Df(\boldsymbol{x})[\boldsymbol{y} - \boldsymbol{x}] \leq f(\boldsymbol{y}) - f(\boldsymbol{x}).$$

- "⇐": Fix any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and let $\boldsymbol{z} \doteq \theta \boldsymbol{y} + (1-\theta)\boldsymbol{x}$. We have,

  $$\begin{aligned} f(\boldsymbol{y}) &\geq f(\boldsymbol{z}) + \boldsymbol{\nabla} f(\boldsymbol{z})^\top (\boldsymbol{y} - \boldsymbol{z}), \quad \text{and} \\ f(\boldsymbol{x}) &\geq f(\boldsymbol{z}) + \boldsymbol{\nabla} f(\boldsymbol{z})^\top (\boldsymbol{x} - \boldsymbol{z}). \end{aligned}$$

  We also have $\boldsymbol{y} - \boldsymbol{z} = (1-\theta)(\boldsymbol{y} - \boldsymbol{x})$ and $\boldsymbol{x} - \boldsymbol{z} = \theta(\boldsymbol{x} - \boldsymbol{y})$. Hence,

  $$\begin{aligned} \theta f(\boldsymbol{y}) + (1-\theta) f(\boldsymbol{x}) &\geq f(\boldsymbol{z}) + \boldsymbol{\nabla} f(\boldsymbol{z})^\top (\underbrace{\theta(\boldsymbol{y} - \boldsymbol{z}) + (1-\theta)(\boldsymbol{x} - \boldsymbol{z})}_{\boldsymbol{0}}) \\ &= f(\theta \boldsymbol{y} + (1-\theta)\boldsymbol{x}). \end{aligned}$$

  □

Theorem 1.56. *Let $f : \mathbb{R}^n \to R$ be a convex and differentiable function. Then, if $\boldsymbol{x}^\star \in \mathbb{R}^n$ is a stationary point of f, $\boldsymbol{x}^\star$ is a global minimum of f.*

Proof. By the first-order characterization of convexity, we have for any $\boldsymbol{y} \in \mathbb{R}^n$,

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}^\star) + \underbrace{\boldsymbol{\nabla} f(\boldsymbol{x}^\star)^\top}_{\boldsymbol{0}} (\boldsymbol{y} - \boldsymbol{x}^\star) = f(\boldsymbol{x}^\star).$$

□

Generally, the main difficulty in solving convex optimization problems lies therefore in finding stationary points (or points that are sufficiently close to stationary points).

Remark 1.57: Second-order characterization of convexity

We say that $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously (Fréchet) differentiable iff for any point $\boldsymbol{x} \in \mathbb{R}^n$, there exists $\boldsymbol{\nabla} f(\boldsymbol{x})$ and $\boldsymbol{H}_f(\boldsymbol{x})$ such that for any $\boldsymbol{\delta} \in \mathbb{R}^n$,

$$f(\boldsymbol{x}+\boldsymbol{\delta}) = f(\boldsymbol{x}) + \boldsymbol{\nabla} f(\boldsymbol{x})^\top \boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^\top \boldsymbol{H}_f(\boldsymbol{x})\boldsymbol{\delta} + o(\|\boldsymbol{\delta}\|_2^2), \quad (1.104)$$

where $\lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{o(\|\boldsymbol{\delta}\|_2^2)}{\|\boldsymbol{\delta}\|_2^2} = 0$, and $\boldsymbol{\nabla} f(\boldsymbol{x})$ and $\boldsymbol{H}_f(\boldsymbol{x})$ are continuous on $\mathbb{R}^n$. Equation (1.104) is also called a *second-order expansion* of f at $\boldsymbol{x}$.

Twice continuously differentiable functions admit a natural characterization of convexity in terms of the Hessian.

Fact 1.58 (Second-order characterization of convexity)**.** *Consider a twice continuously differentiable function $f : \mathbb{R}^n \to R$. Then, f is convex iff $\boldsymbol{H}_f(\boldsymbol{x})$ is positive semi-definite for all $\boldsymbol{x} \in \mathbb{R}^n$.*

It follows that f is concave iff $\boldsymbol{H}_f(\boldsymbol{x})$ is negative semi-definite for all $\boldsymbol{x} \in \mathbb{R}^n$.

Hessian The *Hessian* of a function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $\boldsymbol{x} \in \mathbb{R}^n$ is

$$\begin{aligned}\boldsymbol{H}_f(\boldsymbol{x}) &\doteq \boldsymbol{H} f(\boldsymbol{x}) \\ &\doteq \begin{bmatrix} \frac{\partial^2 f(\boldsymbol{x})}{\partial \boldsymbol{x}(1)\,\partial \boldsymbol{x}(1)} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial \boldsymbol{x}(1)\,\partial \boldsymbol{x}(n)} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial \boldsymbol{x}(n)\,\partial \boldsymbol{x}(1)} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial \boldsymbol{x}(n)\,\partial \boldsymbol{x}(n)} \end{bmatrix} \quad (1.105) \\ &= (\boldsymbol{D}\boldsymbol{\nabla} f(\boldsymbol{x}))^\top \quad (1.106) \\ &\in \mathbb{R}^{n\times n}.\end{aligned}$$

Thus, a Hessian captures the curvature of f. If the Hessian of f is positive definite at a point $\boldsymbol{x}$, then f is "curved up around $\boldsymbol{x}$".

Hessians are symmetric when the second partial derivatives are continuous, due to Schwartz's theorem.

1.5.3 Stochastic Gradient Descent

In this course, we primarily employ so-called first-order methods, which rely on (estimates of) the gradient of the objective function to determine a direction of local improvement. The main idea behind these methods is to repeatedly take a step in the opposite direction of the gradient scaled by a learning rate η_t, which may depend on the current iteration t.

We will often want to minimize a stochastic optimization objective

$$L(\boldsymbol{\theta}) \doteq \mathbb{E}_{\boldsymbol{x}\sim p}[\ell(\boldsymbol{\theta}; \boldsymbol{x})] \quad (1.107)$$

where ℓ and its gradient $\boldsymbol{\nabla}\ell$ are known.

Based on our discussion in previous subsections, it is a natural first step to look for stationary points of L, that is, the roots of $\boldsymbol{\nabla} L$.

Fact 1.59 (Robbins-Monro (RM) algorithm)**.** *Let $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^m$ be an unknown function of which we want to find a root and suppose that we have access to unbiased noisy observations $\mathbf{G}(\boldsymbol{\theta})$ of $\boldsymbol{g}(\boldsymbol{\theta})$. The scheme*

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t - \eta_t \boldsymbol{g}^{(t)}(\boldsymbol{\theta}_t), \quad (1.108)$$

where $\boldsymbol{g}^{(t)}(\boldsymbol{\theta}_t) \sim \mathbf{G}(\boldsymbol{\theta}_t)$ are independent and unbiased estimates of

$g(\theta_t)$, *is known as the* Robbins-Monro algorithm.

It can be shown that if the sequence of learning rates $\{\eta_t\}_{t=0}^{\infty}$ *is chosen such that the* Robbins-Monro conditions,[25]

$$\eta_t \geq 0 \quad \forall t, \quad \sum_{t=0}^{\infty} \eta_t = \infty \quad \textit{and} \quad \sum_{t=0}^{\infty} \eta_t^2 < \infty, \tag{1.109}$$

and additional regularity assumptions are satisfied,[26] *then we have that*

$$g(\theta_t) \xrightarrow{\text{a.s.}} \mathbf{0} \quad \textit{as } t \to \infty. \tag{1.110}$$

That is, the RM-algorithm converges to a root almost surely.[27]

[25] $\eta_t = 1/t$ is an example of a learning rate satisfying the RM-conditions.

[26] For more details, refer to "Online learning and stochastic approximations" (Bottou, 1998).

[27] See eq. (1.78) for a definition of almost sure convergence of a sequence of random variables.

Using Robbins-Monro to find a root of ∇L is known as *stochastic gradient descent*. In particular, when ℓ is convex and the RM-conditions are satisfied, Robbins-Monro converges to a stationary point (and hence, a global minimum) of L almost surely.

Moreover, it can be shown that for general ℓ and satisfied RM-conditions, stochastic gradient descent converges to a local minimum almost surely. Intuitively, the randomness in the gradient estimates allows the algorithm to "jump past" saddle points.

A commonly used strategy to obtain unbiased gradient estimates is to take the sample mean of the gradient with respect to some set of samples B (also called a *batch*) as is shown in alg. 1.60.

Algorithm 1.60: Stochastic gradient descent, SGD

```
initialize θ
repeat
    draw mini-batch B ≐ {x^(1), ..., x^(m)}, x^(i) ~ p
    θ ← θ − η_t (1/m) Σ_{i=1}^{m} ∇_θ ℓ(θ; x^(i))
until converged
```

Example 1.61: Minimizing training loss

A common application of SGD in the context of machine learning is the following: $p \doteq \text{Unif}(\{x_1, \ldots, x_n\})$ is a uniform distribution over the training inputs, yielding the objective

$$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(\theta; x_i), \tag{1.111}$$

where $\ell(\theta; x_i)$ is the loss of a model parameterized by θ for train-

ing input $\boldsymbol{x}_i$ and m is a fixed *batch size*. Here, computing the gradients exactly (i.e., choosing $m = n$) is expensive when the size of the training set is large.

A commonly used alternative to sampling each batch independently, is to split the training set into equally sized batches and perform a gradient descent step with respect to each of them. Typically, the optimum is not found after a single pass through the training data, so the same procedure is repeated multiple times. One such iteration is called an *epoch*.

Remark 1.62: Regularization via weight decay

A common technique to improve the "stability" of minima found through gradient-based methods is to "regularize" the loss by adding an explicit bias favoring "simpler" solutions. That is, given a loss ℓ measuring the quality of fit, we consider the regularized loss

$$\ell'(\boldsymbol{\theta}; \boldsymbol{x}) \doteq \underbrace{\ell(\boldsymbol{\theta}; \boldsymbol{x})}_{\text{quality of fit}} + \underbrace{r(\boldsymbol{\theta})}_{\text{regularization}} \tag{1.112}$$

where $r(\boldsymbol{\theta})$ is large for "complex" and small for "simple" choices of $\boldsymbol{\theta}$, respectively. A common choice is $r(\boldsymbol{\theta}) = \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2$ for some $\lambda > 0$, which is known as *ℓ_2-regularization*.

Recall from section 1.4.4 that in the context of likelihood maximization, this choice of r corresponds to imposing the Gaussian prior $\mathcal{N}(\mathbf{0}, \lambda \boldsymbol{I})$, and finding the MAP estimate. Imposing, for example, a Laplace prior leads to *ℓ_1-regularization*.

Using ℓ_2-regularization, we obtain for the gradient,

$$\nabla_{\boldsymbol{\theta}} \ell'(\boldsymbol{\theta}; \boldsymbol{x}) = \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \boldsymbol{x}) + \lambda \boldsymbol{\theta}. \tag{1.113}$$

Thus, a gradient descent step changes to

$$\boldsymbol{\theta} \leftarrow (1 - \lambda \eta_t) \boldsymbol{\theta} - \eta_t \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \boldsymbol{x}). \tag{1.114}$$

That is, in each step, $\boldsymbol{\theta}$ decays towards zero at the rate $\lambda \eta_t$. This regularization method is also called *weight decay*.

Remark 1.63: Adaptive learning rates and momentum

Commonly, in optimization, constant learning rates are used to accelerate mixing. The performance can be further improved by taking an adaptive gradient step with respect to the geometry of the cost function, which is done by commonly used algorithms such as *Adagrad* and *Adam*. The methods employed by these algorithms are known as *adaptive learning rates* and *momentum*.

For more details on momentum read section 7.1.2 of "Mathematics for machine learning" (Deisenroth et al., 2020) and for an overview of the aforementioned optimization algorithms refer to "An overview of gradient descent optimization algorithms" (Ruder, 2016).

We will have a brief look at other approaches to stochastic optimization in section 6.3.

1.6 Multivariate Normal Distribution

Using general distributions for learning and inference is computationally very expensive when the number of dimensions is large — even in the discrete setting. Computing marginal distributions using the sum rule yields an exponentially long sum in the size of the random vector. Similarly, the normalizing constant of the conditional distribution is a sum of exponential length. Even to represent any discrete joint probability distribution requires space that is exponential in the number of dimensions (cf. fig. 1.11). One strategy to get around this computational blowup is to restrict the class of distributions.

X_1	$\cdots$	X_{n-1}	X_n	$\mathbb{P}(X_{1:n})$
0	$\cdots$	0	0	0.01
0	$\cdots$	0	1	0.001
0	$\cdots$	1	0	0.213
$\cdots$	$\cdots$	$\cdots$	$\cdots$	
1	$\cdots$	1	1	0.0003

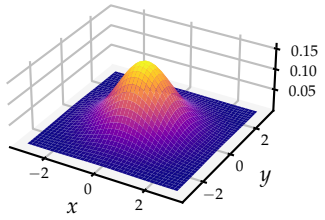
Figure 1.11: A table representing a joint distribution of n binary random variables. The number of parameters is $2^n - 1$.

It turns out that Gaussians have many extremely useful properties: they have a compact representation and — as we will see in later chapters — they allow for closed-form learning and inference.

In eq. (1.6), we have already seen the PDF of the univariate Gaussian distribution. A random vector $\mathbf{X}$ in $\mathbb{R}^n$ is *normally distributed*, $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, if its PDF is

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \doteq \frac{1}{\sqrt{\det(2\pi\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Lambda}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{1.115}$$

where $\boldsymbol{\mu} \in \mathbb{R}^n$ is the mean, $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$ the covariance matrix, and $\boldsymbol{\Lambda} \doteq \boldsymbol{\Sigma}^{-1}$ the so-called *precision matrix*. $\mathbf{X}$ is also called a *Gaussian random vector* (GRV). $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is the multivariate *standard normal distribution*. In this definition, we assume that the covariance matrix $\boldsymbol{\Sigma}$ is invertible,



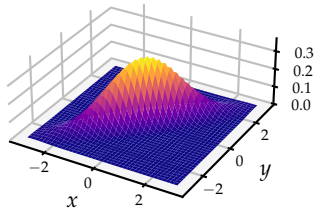


Figure 1.12: Shown are two-dimensional Gaussians with mean $\mathbf{0}$ and covariance matrices

$$\boldsymbol{\Sigma}_1 \doteq \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \boldsymbol{\Sigma}_2 \doteq \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}$$

respectively.

i.e., does not have the eigenvalue 0.

Remark 1.64: Zero eigenvalues of covariance matrices

It can be shown that a covariance matrix has a zero eigenvalue iff there exists a deterministic affine[28] relationship between some of the variables in the joint distribution.

[28] An *affine transformation* is a function $f : \mathbb{R}^n \to \mathbb{R}^m, \boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is an invertible linear map and $\boldsymbol{b} \in \mathbb{R}^m$ is called a translation.

As we have already seen that a covariance matrix does not have negative eigenvalues,[29] this ensures that $\boldsymbol{\Sigma}$ and $\boldsymbol{\Lambda}$ are positive definite.[30]

[29] see exercise 1.28

[30] The inverse of a positive definite matrix is also positive definite.

We call a Gaussian *isotropic* iff its covariance matrix is of the form $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}$ for some $\sigma^2 \in \mathbb{R}$. In this case, the sublevel sets of the PDF are perfect spheres.

Note that a Gaussian can be represented using only $\mathcal{O}(n^2)$ parameters. In the case of a diagonal covariance matrix (which as we will see corresponds to independent univariate Gaussians) we just need $\mathcal{O}(n)$ parameters.

Exercise 1.65: Product of Gaussian PDFs

Let $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2 \in \mathbb{R}^n$ be mean vectors and $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2 \in \mathbb{R}^{n \times n}$ be covariance matrices. Prove that

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \tag{1.116}$$

for some mean vector $\boldsymbol{\mu} \in \mathbb{R}^n$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$. That is, show that the product of two Gaussian PDFs is proportional to the PDF of a Gaussian.

▷ *Solution*

1.6.1 *Properties of Gaussians*

In this section, we study some of the special properties of Gaussians.

Theorem 1.66 (Marginal and conditional distribution)**.** *Consider the Gaussian random vector* $\mathbf{X}$ *and fix index sets* $A \subseteq [n]$ *and* $B \subseteq [n]$*. Then, we have that for any such* marginal distribution*,*

$$\mathbf{X}_A \sim \mathcal{N}(\boldsymbol{\mu}_A, \boldsymbol{\Sigma}_{AA}), \tag{1.117}$$

By $\boldsymbol{\mu}_A$ we denote $[\mu_{i_1}, \dots, \mu_{i_k}]$ where $A = \{i_1, \dots i_k\}$. $\boldsymbol{\Sigma}_{AA}$ is defined analogously.

and that for any such conditional distribution*,*

$$\mathbf{X}_A \mid \mathbf{X}_B = \boldsymbol{x}_B \sim \mathcal{N}(\boldsymbol{\mu}_{A|B}, \boldsymbol{\Sigma}_{A|B}) \quad \textit{where} \tag{1.118}$$

$$\boldsymbol{\mu}_{A|B} \doteq \boldsymbol{\mu}_A + \boldsymbol{\Sigma}_{AB}\boldsymbol{\Sigma}_{BB}^{-1}(\boldsymbol{x}_B - \boldsymbol{\mu}_B), \tag{1.119}$$

Here, $\boldsymbol{\mu}_A$ characterizes the prior belief and $\boldsymbol{\Sigma}_{AB}\boldsymbol{\Sigma}_{BB}^{-1}(\boldsymbol{x}_B - \boldsymbol{\mu}_B)$ represents "how different" $\boldsymbol{x}_B$ is from what is expected.

$$\boldsymbol{\Sigma}_{A|B} \doteq \boldsymbol{\Sigma}_{AA} - \boldsymbol{\Sigma}_{AB}\boldsymbol{\Sigma}_{BB}^{-1}\boldsymbol{\Sigma}_{BA}. \tag{1.120}$$

Observe, that the variance can only shrink! Moreover, how much the variance is reduced depends purely on *where* the observations are made (e.g., the choice of B) but not on *what* the observations are. Also observe that $\boldsymbol{\mu}_{A|B}$ depends affinely on $\boldsymbol{\mu}_B$.

Exercise 1.67: Marginal / conditional distribution of a Gaussian

Prove theorem 1.66. That is, show that

1. every marginal of a Gaussian is Gaussian; and
2. conditioning on a subset of variables of a joint Gaussian is Gaussian

by finding their corresponding PDFs.

Hint: You may use that for matrices $\boldsymbol{\Sigma}$ and $\boldsymbol{\Lambda}$ such that $\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Lambda}$,

- *if $\boldsymbol{\Sigma}$ and $\boldsymbol{\Lambda}$ are symmetric,*

$$\begin{aligned}
&\begin{bmatrix}\boldsymbol{x}_A\\ \boldsymbol{x}_B\end{bmatrix}^\top \begin{bmatrix}\boldsymbol{\Lambda}_{AA} & \boldsymbol{\Lambda}_{AB}\\ \boldsymbol{\Lambda}_{BA} & \boldsymbol{\Lambda}_{BB}\end{bmatrix} \begin{bmatrix}\boldsymbol{x}_A\\ \boldsymbol{x}_B\end{bmatrix}\\
&= \boldsymbol{x}_A^\top \boldsymbol{\Lambda}_{AA}\boldsymbol{x}_A + \boldsymbol{x}_A^\top \boldsymbol{\Lambda}_{AB}\boldsymbol{x}_B + \boldsymbol{x}_B^\top \boldsymbol{\Lambda}_{BA}\boldsymbol{x}_A + \boldsymbol{x}_B^\top \boldsymbol{\Lambda}_{BB}\boldsymbol{x}_B\\
&= \boldsymbol{x}_A^\top (\boldsymbol{\Lambda}_{AA} - \boldsymbol{\Lambda}_{AB}\boldsymbol{\Lambda}_{BB}^{-1}\boldsymbol{\Lambda}_{BA})\boldsymbol{x}_A +\\
&\qquad (\boldsymbol{x}_B + \boldsymbol{\Lambda}_{BB}^{-1}\boldsymbol{\Lambda}_{BA}\boldsymbol{x}_A)^\top \boldsymbol{\Lambda}_{BB}(\boldsymbol{x}_B + \boldsymbol{\Lambda}_{BB}^{-1}\boldsymbol{\Lambda}_{BA}\boldsymbol{x}_A),
\end{aligned}$$

- $\boldsymbol{\Lambda}_{BB}^{-1} = \boldsymbol{\Sigma}_{BB} - \boldsymbol{\Sigma}_{BA}\boldsymbol{\Sigma}_{AA}^{-1}\boldsymbol{\Sigma}_{AB}$,
- $\boldsymbol{\Lambda}_{BB}^{-1}\boldsymbol{\Lambda}_{BA} = \boldsymbol{\Sigma}_{BA}\boldsymbol{\Sigma}_{AA}^{-1}$.

The final two equations follow from the general characterization of the inverse of a block matrix (Petersen et al., 2008, section 9.1.3).

▷ *Solution*

The following theorem states that the independence of Gaussians is characterized by their correlation.

Theorem 1.68 (Independence of Gaussians)**.** *Two jointly Gaussian random vectors, $\mathbf{X}$ and $\mathbf{Y}$, are independent if and only if $\mathbf{X}$ and $\mathbf{Y}$ are uncorrelated.*

Proof. Recall that independence of any random vectors $\mathbf{X}$ and $\mathbf{Y}$ implies that they are uncorrelated,[31] the converse implication is a special property of Gaussian random vectors.

[31] This is because the expectation of their product factorizes, see eq. (1.26).

Consider the joint Gaussian random vector $\mathbf{Z} \doteq [\mathbf{X}\ \mathbf{Y}] \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and assume that $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{X}, \boldsymbol{\Sigma}_\mathbf{X})$ and $\mathbf{Y} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{Y}, \boldsymbol{\Sigma}_\mathbf{Y})$ are uncorrelated.

Then, $\boldsymbol{\Sigma}$ can be expressed as

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{X}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{\mathbf{Y}} \end{bmatrix},$$

implying that the PDF of $\mathbf{Z}$ factorizes,

$$\begin{aligned} &\mathcal{N}([\boldsymbol{x}\ \boldsymbol{y}]; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ &= \frac{1}{\sqrt{\det(2\pi\boldsymbol{\Sigma}_{\mathbf{X}}) \cdot \det(2\pi\boldsymbol{\Sigma}_{\mathbf{Y}})}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_{\mathbf{X}})^\top \boldsymbol{\Sigma}_{\mathbf{X}}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_{\mathbf{X}}) \right. \\ &\qquad \left. -\frac{1}{2}(\boldsymbol{y} - \boldsymbol{\mu}_{\mathbf{Y}})^\top \boldsymbol{\Sigma}_{\mathbf{Y}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_{\mathbf{Y}}) \right) \\ &= \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{\mathbf{X}}, \boldsymbol{\Sigma}_{\mathbf{X}}) \cdot \mathcal{N}(\boldsymbol{y}; \boldsymbol{\mu}_{\mathbf{Y}}, \boldsymbol{\Sigma}_{\mathbf{Y}}). \end{aligned}$$ □

In particular, this implies that random variables that are jointly Gaussian are mutually independent iff the covariance matrix of the corresponding multivariate normal distribution is diagonal.

Exercise 1.69: Closedness properties of Gaussians

Next, we derive two important closedness properties of Gaussians.

First, we recall the notion of a *moment-generating function* (MGF) of a random vector $\mathbf{X}$ in $\mathbb{R}^n$ which is defined as

$$\varphi_{\mathbf{X}}(\boldsymbol{t}) \doteq \mathbb{E} \exp\left(\boldsymbol{t}^\top \mathbf{X}\right), \quad \text{for all } \boldsymbol{t} \in \mathbb{R}^n. \tag{1.121}$$

An MGF uniquely characterizes a distribution. The MGF of the multivariate Gaussian $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is

$$\varphi_{\mathbf{X}}(\boldsymbol{t}) = \exp\left(\boldsymbol{t}^\top \boldsymbol{\mu} + \frac{1}{2}\boldsymbol{t}^\top \boldsymbol{\Sigma} \boldsymbol{t} \right). \tag{1.122}$$

This generalizes the MGF of the univariate Gaussian from eq. (1.87).

Prove the following facts.

1. **Fact 1.70** (Closedness under affine transformations)**.** *Given an n-dimensional Gaussian $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$,*

$$\boldsymbol{A}\mathbf{X} + \boldsymbol{b} \sim \mathcal{N}(\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b}, \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top). \tag{1.123}$$

2. **Fact 1.71** (Additivity)**.** *Given two independent Gaussian random vectors $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\mathbf{X}' \sim \mathcal{N}(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$ in $\mathbb{R}^n$,*

$$\mathbf{X} + \mathbf{X}' \sim \mathcal{N}(\boldsymbol{\mu} + \boldsymbol{\mu}', \boldsymbol{\Sigma} + \boldsymbol{\Sigma}'). \tag{1.124}$$

These properties are unique to Gaussians and a reason for why they are widely used for learning and inference.

▷ *Solution*

Observe that eq. (1.123) implies that the random vector $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\boldsymbol{\Sigma}$ is positive definite is equivalently characterized as

$$\mathbf{X} = \boldsymbol{\Sigma}^{1/2}\mathbf{Y} + \boldsymbol{\mu}. \tag{1.125}$$

where $\mathbf{Y} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $\boldsymbol{\Sigma}^{1/2}$ is the square root of $\boldsymbol{\Sigma}$.[32] Importantly, this implies together with theorem 1.66 and fact 1.71 that:

[32] More details on the square root of a symmetric and positive semi-definite matrix can be found in section 1.6.3.

- Any marginal of a GRV is a GRV.
- Any affine transformation of a GRV is a GRV.

Exercise 1.72: Expectation and Variance of Gaussians

Derive that $\mathbb{E}[\mathbf{X}] = \boldsymbol{\mu}$ and $\mathrm{Var}[\mathbf{X}] = \boldsymbol{\Sigma}$ when $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.
Hint: First derive the expectation and variance of a univariate standard normal random variable.

▷ *Solution*

1.6.2 *Conditional Linear Gaussians*

It is often useful to consider a Gaussian random variable X in terms of another Gaussian random variable Y. Suppose $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$ and $Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$ are jointly Gaussian. Then, using the closed-form expression of Gaussian conditionals (1.118), their conditional distribution is given as

$$X \mid Y = y \sim \mathcal{N}(\mu_{X|Y}, \sigma_{X|Y}^2) \quad \text{where} \tag{1.126}$$

$$\mu_{X|Y} \doteq \mu_X + \sigma_{XY}\sigma_Y^{-2}(y - \mu_Y), \tag{1.127}$$

$$\sigma_{X|Y}^2 \doteq \sigma_X^2 - \sigma_{XY}\sigma_Y^{-2}\sigma_{YX}. \tag{1.128}$$

Here, σ_{XY} refers to the covariance of X and Y. Again, note that $\mu_{X|Y}$ depends affinely on μ_X.

Observe that this admits an equivalent characterization of X as an affine function of Y with added independent Gaussian noise. Formally, we define

$$X \doteq aY + b + \epsilon \quad \text{where} \tag{1.129}$$

$$a \doteq \sigma_{XY}\sigma_Y^{-2}, \tag{1.130}$$

$$b \doteq \mu_X - \sigma_{XY}\sigma_Y^{-2}\mu_Y, \tag{1.131}$$

$$\epsilon \sim \mathcal{N}(0, \sigma_{X|Y}^2). \tag{1.132}$$

It directly follows from the closedness of Gaussians under linear transformations (1.123) that the characterization of X via (1.129) is equivalent to $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$, and hence, *any* Gaussian X can be modeled as

a *conditional linear Gaussian*, i.e., a linear function of another Gaussian Y with additional independent Gaussian noise.

1.6.3 Quadratic Forms

Definition 1.73 (Quadratic form)**.** Given a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, the *quadratic form* induced by $\boldsymbol{A}$ is defined as

$$f : \mathbb{R}^n \to \mathbb{R}, \quad \boldsymbol{x} \mapsto \boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} = \sum_{i=1}^{n} \sum_{j=1}^{n} \boldsymbol{A}(i,j) \cdot \boldsymbol{x}(i) \cdot \boldsymbol{x}(j). \tag{1.133}$$

We call $\boldsymbol{A}$ a *positive definite* matrix if all eigenvalues of $\boldsymbol{A}$ are positive. Equivalently, we have $f(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ and $f(\mathbf{0}) = 0$. Similarly, $\boldsymbol{A}$ is called *positive semi-definite* if all eigenvalues of $\boldsymbol{A}$ are non-negative, or equivalently, if $f(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$. In particular, if $\boldsymbol{A}$ is positive definite then $\sqrt{f(\boldsymbol{x})}$ is a norm (called the *Mahalanobis norm* induced by $\boldsymbol{A}$), and is often denoted by $\|\boldsymbol{x}\|_{\boldsymbol{A}}$.

If $\boldsymbol{A}$ is positive definite, then the sublevel sets of its induced quadratic form f are convex ellipsoids. Not coincidentally, the same is true for the sublevel sets of the PDF of a normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, which we have seen an example of in fig. 1.12. Hereby, the axes of the ellipsoid and their corresponding squared lengths are the eigenvectors and eigenvalues of $\boldsymbol{\Sigma}$, respectively.

Remark 1.74: Correspondence of quadratic forms and Gaussians

Observe that the PDF of a zero-mean multivariate Gaussian $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ is an exponentiated and appropriately scaled quadratic form induced by the positive definite precision matrix $\boldsymbol{\Sigma}^{-1}$. The constant factor is chosen such that the resulting function is a valid probability density function, that is, sums to one.

One important property of positive definiteness of $\boldsymbol{\Sigma}$ is that $\boldsymbol{\Sigma}$ can be decomposed into the product of a lower-triangular matrix with its transpose. This is known as the *Cholesky decomposition*.

Fact 1.75 (Cholesky decomposition, symmetric matrix-form)**.** *For any symmetric and positive (semi-)definite matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, there is a decomposition of the form*

$$\boldsymbol{A} = \mathcal{L}\mathcal{L}^\top \tag{1.134}$$

where $\mathcal{L} \in \mathbb{R}^{n \times n}$ is lower triangular and positive (semi-)definite.

We will not prove this fact, but it is not hard to see that a decomposition exists (it takes more work to show that $\mathcal{L}$ is lower-triangular).

Let $\boldsymbol{A}$ be a symmetric and positive (semi-)definite matrix. By the spectral theorem of symmetric matrices, $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top$, where $\boldsymbol{\Lambda}$ is a diagonal matrix of eigenvalues and $\boldsymbol{V}$ is an orthonormal matrix of corresponding eigenvectors. Consider the matrix

$$\boldsymbol{A}^{1/2} \doteq \boldsymbol{V}\boldsymbol{\Lambda}^{1/2}\boldsymbol{V}^\top \tag{1.135}$$

where $\boldsymbol{\Lambda}^{1/2} \doteq \operatorname{diag}\{\sqrt{\boldsymbol{\Lambda}(i,i)}\}$, also called the *square root* of $\boldsymbol{A}$. Then,

$$\begin{aligned}\boldsymbol{A}^{1/2}\boldsymbol{A}^{1/2} &= \boldsymbol{V}\boldsymbol{\Lambda}^{1/2}\boldsymbol{V}^\top\boldsymbol{V}\boldsymbol{\Lambda}^{1/2}\boldsymbol{V}^\top \\ &= \boldsymbol{V}\boldsymbol{\Lambda}^{1/2}\boldsymbol{\Lambda}^{1/2}\boldsymbol{V}^\top \\ &= \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top = \boldsymbol{A}. \end{aligned} \tag{1.136}$$

It is immediately clear from the definition that $\boldsymbol{A}^{1/2}$ is also symmetric and positive (semi-)definite.

Quadratic forms of positive semi-definite matrices are a generalization of the Euclidean norm, as

$$\|\boldsymbol{x}\|_{\boldsymbol{A}}^2 = \boldsymbol{x}^\top\boldsymbol{A}\boldsymbol{x} = \boldsymbol{x}^\top\boldsymbol{A}^{1/2}\boldsymbol{A}^{1/2}\boldsymbol{x} = (\boldsymbol{A}^{1/2}\boldsymbol{x})^\top\boldsymbol{A}^{1/2}\boldsymbol{x} = \|\boldsymbol{A}^{1/2}\boldsymbol{x}\|_2^2, \tag{1.137}$$

and in particular,

$$\log\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = -\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{\mu}\|_{\boldsymbol{\Sigma}^{-1}}^2 + \text{const} \tag{1.138}$$

$$= -\frac{1}{2}\left[\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x} - 2\boldsymbol{\mu}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x}\right] + \text{const}, \tag{1.139}$$

$$\log\mathcal{N}(\boldsymbol{x};\boldsymbol{0},\boldsymbol{I}) = -\frac{1}{2}\|\boldsymbol{x}\|_2^2 + \text{const}. \tag{1.140}$$

PART I

Probabilistic Machine Learning

2

Bayesian Linear Regression

2.1 *Linear Regression*

We consider the supervised learning setting that we have introduced in section 1.2. Given a set of $(\boldsymbol{x}, y)$ pairs, linear regression aims to find a linear model that fits the data optimally.[1] Given the linear model

$$y \approx \boldsymbol{w}^\top \boldsymbol{x} \doteq f(\boldsymbol{x}; \boldsymbol{w})$$

for $\boldsymbol{x} \in \mathbb{R}^d$ and labeled training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, we want to find optimal weights $\boldsymbol{w}$.

[1] As we have discussed in section 1.2, regression models can also be used for classification. One example of a linear regression model for classification is Bayesian logistic regression, which we will see in example 5.2.

Remark 2.1: Implicit use of intercepts

In general, a linear function

$$\boldsymbol{w}^\top \boldsymbol{x} + w_0$$

also has an intercept term w_0. Observe, however, that if we define $\boldsymbol{x}' \doteq (\boldsymbol{x}, 1)$ and $\boldsymbol{w}' \doteq (\boldsymbol{w}, w_0)$, then we have $\boldsymbol{w}'^\top \boldsymbol{x}' = \boldsymbol{w}^\top \boldsymbol{x} + w_0$, implying that w.l.o.g. it suffices to study linear functions without the intercept term w_0.

We define the *design matrix*,

$$\boldsymbol{X} \doteq \begin{bmatrix} \boldsymbol{x}_1^\top \\ \vdots \\ \boldsymbol{x}_n^\top \end{bmatrix} \in \mathbb{R}^{n \times d}, \tag{2.1}$$

as the collection of inputs and the vector $\boldsymbol{y} \doteq [y_1 \cdots y_n]^\top \in \mathbb{R}^n$ as the collection of labels. For each noisy observation $(\boldsymbol{x}_i, y_i)$, we define the value of the approximation of our model, $f_i \doteq \boldsymbol{w}^\top \boldsymbol{x}_i$. Our model at the inputs $\boldsymbol{X}$ is described by the vector $\boldsymbol{f} \doteq [f_1 \ \cdots \ f_n]^\top$. It requires little calculation to verify that $\boldsymbol{f} = \boldsymbol{X}\boldsymbol{w}$.

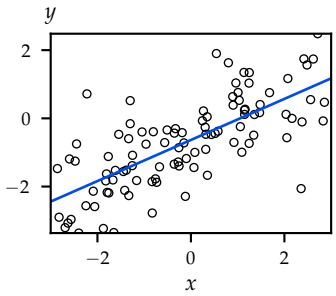


Figure 2.1: Example of linear regression with the least squares estimator (shown in blue).

There are many ways of estimating $\boldsymbol{w}$ from data, the most common being the *least squares estimator*,

$$\hat{\boldsymbol{w}}_{\text{ls}} \doteq \arg\min_{\boldsymbol{w}\in\mathbb{R}^d} \sum_{i=1}^{n} (y_i - \boldsymbol{w}^\top \boldsymbol{x}_i)^2 = \arg\min_{\boldsymbol{w}\in\mathbb{R}^d} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 , \tag{2.2}$$

minimizing the squared difference between the labels and predictions of the model. A slightly different estimator is used for *ridge regression*,

$$\hat{\boldsymbol{w}}_{\text{ridge}} \doteq \arg\min_{\boldsymbol{w}\in\mathbb{R}^d} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda \|\boldsymbol{w}\|_2^2 \tag{2.3}$$

where $\lambda > 0$. The squared ℓ_2 regularization term $\lambda \|\boldsymbol{w}\|_2^2$ penalizes large $\boldsymbol{w}$ and thus reduces the complexity of the resulting model. This reduces the potential of overfitting to the training data.[2]

[2] Ridge regression is more robust than standard linear regression in the presence of multicollinearity. *Multicollinearity* occurs when multiple independent inputs are highly correlated. In this case, their individual effects on the predicted variable cannot be estimated well. Classical linear regression is highly volatile to small input changes. The regularization of ridge regression reduces this volatility by introducing a bias on the weights towards 0.

Exercise 2.2: Closed-form linear regression

1. Show that the unique solutions to least squares and ridge regression are given by

$$\hat{\boldsymbol{w}}_{\text{ls}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y} \qquad \text{and} \tag{2.4}$$

$$\hat{\boldsymbol{w}}_{\text{ridge}} = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y}, \tag{2.5}$$

   respectively if the Hessian of the loss is positive definite (i.e., the losses are strictly convex functions).
2. For a $n \times m$ matrix $\boldsymbol{A}$ and vector $\boldsymbol{x} \in \mathbb{R}^m$, we call $\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{x}$ the orthogonal projection of $\boldsymbol{x}$ onto $\text{span}\{\boldsymbol{A}\} = \{\boldsymbol{A}\boldsymbol{x}' \mid \boldsymbol{x}' \in \mathbb{R}^m\}$. In particular, an orthogonal projection satisfies $\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{x} \perp \boldsymbol{A}\boldsymbol{x}'$ for all $\boldsymbol{x}' \in \mathbb{R}^m$.
   Show that $\hat{\boldsymbol{w}}_{\text{ls}}$ from eq. (2.4) is such that $\boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ls}}$ is the unique closest point to $\boldsymbol{y}$ on $\text{span}\{\boldsymbol{X}\}$, i.e., it satisfies $\boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ls}} = \boldsymbol{\Pi}_{\boldsymbol{X}}\boldsymbol{y}$.

▷ *Solution*

Readings

For a more thorough introduction to linear regression, read

- chapter 4 of "Introduction to Machine Learning" (Krause and Yang, 2022) or
- chapter 9 of "Mathematics for machine learning" (Deisenroth et al., 2020).

2.1.1 *Maximum Likelihood Estimation*

Let us assume that the observations

$$y_i = \boldsymbol{w}^\top \boldsymbol{x}_i + \epsilon_i \tag{2.6}$$

are made under the influence of i.i.d. homoscedastic Gaussian noise $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$.[3] Recall from section 1.6.2 that this is equivalently characterized by the Gaussian likelihood,

$$y_i \mid \boldsymbol{x}_i, \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_i, \sigma_n^2). \tag{2.7}$$

[3] ϵ_i is often called *additive white Gaussian noise*. In practice, the noise variance σ_n^2 is typically unknown and also has to be estimated. It is a straightforward exercise to check that the maximum likelihood estimate of σ_n^2 given fixed weights $\boldsymbol{w}$ is given by $\hat{\sigma}_n^2 = \frac{1}{n}\sum_{i=1}^n (y_i - \boldsymbol{w}^\top \boldsymbol{x}_i)^2$.

Computing the maximum likelihood estimate for the weights,

$$\begin{aligned}
\hat{\boldsymbol{w}}_{\text{MLE}} &= \operatorname*{arg\,max}_{\boldsymbol{w}\in\mathbb{R}^d} \log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{w}) \\
&= \operatorname*{arg\,max}_{\boldsymbol{w}\in\mathbb{R}^d} \sum_{i=1}^n \log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}) && \text{using independence of the training data} \\
&= \operatorname*{arg\,min}_{\boldsymbol{w}\in\mathbb{R}^d} \sum_{i=1}^n (y_i - \boldsymbol{w}^\top \boldsymbol{x}_i)^2, && \text{plugging in the Gaussian likelihood}
\end{aligned}$$

we observe that $\hat{\boldsymbol{w}}_{\text{MLE}} = \hat{\boldsymbol{w}}_{\text{ls}}$.

Example 2.3: Variance of least squares

It is useful to understand the variance of the weights $\hat{\boldsymbol{w}}_{\text{ls}}$ in terms of the data $(\boldsymbol{X}, \boldsymbol{y})$ because this allows us to determine the variance of a prediction,

$$\begin{aligned}
\operatorname{Var}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{X}] &= \operatorname{Var}\left[\hat{\boldsymbol{w}}_{\text{ls}}^\top \boldsymbol{x}^\star \;\middle|\; \boldsymbol{x}^\star, \boldsymbol{X}\right] && \text{using that } y^\star \text{ is a linear function in } \hat{\boldsymbol{w}}_{\text{ls}} \text{ evaluated at } \boldsymbol{x}^\star \\
&= \boldsymbol{x}^{\star\top} \operatorname{Var}[\hat{\boldsymbol{w}}_{\text{ls}} \mid \boldsymbol{X}]\, \boldsymbol{x}^\star. && \text{using eq. (1.47)} \qquad (2.8)
\end{aligned}$$

The variance of the weights of the least squares estimator is

$$\begin{aligned}
\operatorname{Var}[\hat{\boldsymbol{w}}_{\text{ls}} \mid \boldsymbol{X}] &= \operatorname{Var}\left[(\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y} \;\middle|\; \boldsymbol{X}\right] && \text{using eq. (2.4)} \\
&= (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \operatorname{Var}[\boldsymbol{y} \mid \boldsymbol{X}]\, ((\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top)^\top && \text{using eq. (1.47)} \\
&= (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \operatorname{Var}[\boldsymbol{y} \mid \boldsymbol{X}]\, \boldsymbol{X} (\boldsymbol{X}^\top \boldsymbol{X})^{-1}. && \text{using } (\boldsymbol{A}^\top)^{-1} = (\boldsymbol{A}^{-1})^\top
\end{aligned}$$

Due to the Gaussian likelihood (2.7), $\operatorname{Var}[\boldsymbol{y} \mid \boldsymbol{X}] = \sigma_n^2 \boldsymbol{I}$, so

$$= \sigma_n^2 (\boldsymbol{X}^\top \boldsymbol{X})^{-1}. \tag{2.9}$$

Exercise 2.4: Variance of least squares around training data

Show that the variance of a prediction at the point $[1\ x^\star]^\top$ is smallest when $x^\star$ is the mean of the training data. More formally, show that if inputs are of the form $\boldsymbol{x}_i = [1\ x_i]^\top$ where $x_i \in \mathbb{R}$ and $\hat{\boldsymbol{w}}_{\text{ls}}$ is the least squares estimate, then $\operatorname{Var}\left[y^\star \mid [1\ x^\star]^\top, \hat{\boldsymbol{w}}_{\text{ls}}\right]$ is minimized for $x^\star = \frac{1}{n}\sum_{i=1}^n x_i$.

▷ *Solution*

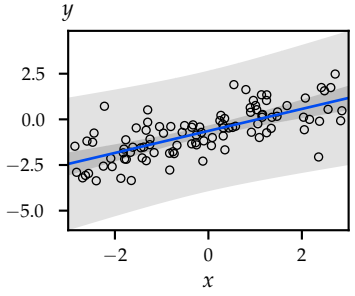


Figure 2.2: The same linear regression model as in fig. 2.1 assuming $\sigma_n = 1$. The estimated variance of the weights (i.e., epistemic uncertainty) is shown in dark gray, the aleatoric uncertainty in light gray.

2.2 *Weight-space View*

There are two views on Bayesian linear regression, the weight-space view and the function-space view. We begin by discussing the common view in terms of a weight parameterization. In section 2.6, we introduce the interpretation in terms of functions, which will then naturally lead us to Gaussian processes.

The most immediate and natural Bayesian interpretation of linear regression is to simply impose a prior on the weights $\boldsymbol{w}$. We will use a Gaussian prior,

$$\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \sigma_p^2 \boldsymbol{I}), \tag{2.10}$$

and the Gaussian likelihood from eq. (2.7).[4]

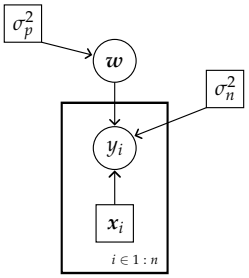


Figure 2.3: Directed graphical model of Bayesian linear regression in plate notation.

[4] The choice of hyperparameters such as the prior variance σ_p^2 is discussed in greater detail in section 4.4.

Remark 2.5: Maximum entropy principle

The choice of using a Gaussian prior may seem somewhat arbitrary at first sight (except for the nice analytical properties of Gaussians that we have seen in section 1.6). The following intuitive principle gives a more fundamental argument in favor of Gaussian priors.

The *entropy* of a distribution p is defined as

$$\mathrm{H}[p] \doteq \mathbb{E}_{\boldsymbol{x} \sim p}[-\log p(\boldsymbol{x})]. \tag{2.11}$$

Intuitively, if the entropy of p is high, then the distribution p exhibits high "uncertainty". We give a thorough introduction to entropy in section 5.4.

When choosing a prior, the *maximum entropy principle* states that one should choose the distribution maximizing entropy subject to all the constraints that are to be imposed on the prior. Intuitively, this encodes the least information necessary in the prior. In philosophy, this principle is also known as *Occam's razor*. It turns out that $\mathcal{N}$ has the maximum entropy among all distributions on $\mathbb{R}^d$ with known mean and covariance. You will prove this in the univariate setting in exercise 5.20.

Thus, the maximum entropy principle can serve as an argument for why it might be reasonable to assume a Gaussian prior.

Next, let us derive the posterior distribution over the weights.

$$\begin{aligned} &\log p(\boldsymbol{w} \mid \boldsymbol{x}_{1:n}, y_{1:n}) \\ &= \log p(\boldsymbol{w}) + \log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{w}) + \text{const} \end{aligned}$$

by Bayes' rule (1.59)

$$= \log p(\boldsymbol{w}) + \sum_{i=1}^{n} \log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}) + \text{const} \qquad \text{using independence of the samples}$$

$$= -\frac{1}{2}\left[\sigma_p^{-2} \|\boldsymbol{w}\|_2^2 + \sigma_n^{-2} \sum_{i=1}^{n} (y_i - \boldsymbol{w}^\top \boldsymbol{x}_i)^2\right] + \text{const} \qquad \text{using the Gaussian prior and likelihood}$$

$$= -\frac{1}{2}\left[\sigma_p^{-2} \|\boldsymbol{w}\|_2^2 + \sigma_n^{-2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2\right] + \text{const} \qquad \text{using } \textstyle\sum_{i=1}^{n} (y_i - \boldsymbol{w}^\top \boldsymbol{x}_i)^2 = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

$$= -\frac{1}{2}\left[\sigma_p^{-2}\boldsymbol{w}^\top\boldsymbol{w} + \sigma_n^{-2}\left(\boldsymbol{w}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{y}^\top\boldsymbol{X}\boldsymbol{w} + \boldsymbol{y}^\top\boldsymbol{y}\right)\right] + \text{const}$$

$$= -\frac{1}{2}\left[\boldsymbol{w}^\top(\sigma_n^{-2}\boldsymbol{X}^\top\boldsymbol{X} + \sigma_p^{-2}\boldsymbol{I})\boldsymbol{w} - 2\sigma_n^{-2}\boldsymbol{y}^\top\boldsymbol{X}\boldsymbol{w}\right] + \text{const}. \tag{2.12}$$

Observe that the log-posterior is a quadratic form in $\boldsymbol{w}$, so the posterior distribution is Gaussian (cf. eq. (1.139)):

$$\boldsymbol{w} \mid \boldsymbol{x}_{1:n}, y_{1:n} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{2.13}$$

where we can read off the mean and covariance to be

$$\boldsymbol{\Sigma} \doteq \left(\sigma_n^{-2}\boldsymbol{X}^\top\boldsymbol{X} + \sigma_p^{-2}\boldsymbol{I}\right)^{-1}, \tag{2.14a}$$

$$\boldsymbol{\mu} \doteq \sigma_n^{-2}\boldsymbol{\Sigma}\boldsymbol{X}^\top\boldsymbol{y}. \tag{2.14b}$$

This also shows that Gaussians with known variance and linear likelihood are self-conjugate, a property that we had hinted at in section 1.3.2. It can be shown more generally that Gaussians with known variance are self-conjugate to any Gaussian likelihood (Murphy, 2007).

For general distributions the posterior can typically not be expressed in closed-form — this is a very special property of Gaussians!

2.2.1 *Maximum A Posteriori Estimation*

Computing the MAP estimate for the weights,

$$\begin{aligned}\hat{\boldsymbol{w}}_{\text{MAP}} &= \arg\max_{\boldsymbol{w}} \log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{w}) + \log p(\boldsymbol{w}) \\ &= \arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \frac{\sigma_n^2}{\sigma_p^2}\|\boldsymbol{w}\|_2^2, \qquad \text{using that the likelihood and prior are Gaussian}\end{aligned} \tag{2.15}$$

we observe that this is identical to ridge regression with weight decay $\lambda \doteq \sigma_n^2/\sigma_p^2$, and hence, $\hat{\boldsymbol{w}}_{\text{MAP}} = \hat{\boldsymbol{w}}_{\text{ridge}}$. Note that this is simply the MLE loss with an additional ℓ_2 regularization term (originating from the prior) that encourages keeping weights small.

Also, recall that the MAP estimate corresponds to the mode of the posterior distribution, which in the case of a Gaussian is simply its mean $\boldsymbol{\mu}$. Ass to be expected, $\boldsymbol{\mu}$ coincides with the analytical solution to ridge regression from eq. (2.5).

Example 2.6: Lasso as the MAP estimate with a Laplace prior

One problem with ridge regression is that the contribution of nearly-zero weights to the ℓ_2 regularization term is negligible. Thus, ℓ_2 regularization is typically not sufficient to perform variable selection, that is, set some weights to zero entirely.

A commonly used alternative to ridge regression is the *least absolute shrinkage and selection operator* (or *lasso*), which uses ℓ_1 regularization,

$$\hat{\boldsymbol{w}}_{\text{lasso}} \doteq \arg\min_{\boldsymbol{w} \in \mathbb{R}^d} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda \|\boldsymbol{w}\|_1 . \tag{2.16}$$

It turns out that lasso can also be viewed as Bayesian learning, using a Laplace prior $\boldsymbol{w} \sim \text{Laplace}(\boldsymbol{0}, \ell)$ with length scale ℓ instead of a Gaussian prior. Recall that for the PDF of the Laplace distribution we have,

$$\text{Laplace}(\boldsymbol{x}; \boldsymbol{\mu}, \ell) \propto \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{\mu}\|_1}{\ell}\right). \tag{2.17}$$

Computing the MAP estimate for the weights yields,

$$\begin{aligned} \hat{\boldsymbol{w}}_{\text{MAP}} &= \arg\max_{\boldsymbol{w}} \log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{w}) + \log p(\boldsymbol{w}) \\ &= \arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} (y_i - \boldsymbol{w}^\top \boldsymbol{x}_i)^2 + \frac{\sigma_n^2}{\ell} \|\boldsymbol{w}\|_1 . \end{aligned} \tag{2.18}$$

using that the likelihood is Gaussian and the prior is Laplacian

This coincides with the optimization objective of lasso with weight decay $\lambda \doteq \sigma_n^2/\ell$.

To make predictions at a test point $\boldsymbol{x}^\star$, we define the (model-)predicted point $f^\star \doteq \hat{\boldsymbol{w}}_{\text{MAP}}^\top \boldsymbol{x}^\star$ and obtain the label prediction

$$y^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, y_{1:n} \sim \mathcal{N}(f^\star, \sigma_n^2). \tag{2.19}$$

Note that using point estimates like the MAP estimate does not quantify uncertainty in the weights. The MAP estimate simply collapses all mass of the posterior around its mode. This is especially harmful when we are highly unsure about the best model (e.g., because we have observed insufficient data).

2.2.2 *Bayesian Inference*

Rather than selecting a single weight vector $\hat{\boldsymbol{w}}$ to make predictions, we can use the full posterior distribution. Doing so is also called "Bayesian

inference", and in the setting of linear regression known as *Bayesian linear regression*.

For a test point $\boldsymbol{x}^\star$, we let $f^\star \doteq \boldsymbol{w}^\top \boldsymbol{x}^\star$. Using the closedness of Gaussians under linear transformations (1.123),

$$f^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, y_{1:n} \sim \mathcal{N}(\boldsymbol{\mu}^\top \boldsymbol{x}^\star, \boldsymbol{x}^{\star\top} \boldsymbol{\Sigma} \boldsymbol{x}^\star). \tag{2.20}$$

Note that this does not take into account the noise in the labels σ_n^2. For the label prediction $y^\star$, we obtain

$$y^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, y_{1:n} \sim \mathcal{N}(\boldsymbol{\mu}^\top \boldsymbol{x}^\star, \boldsymbol{x}^{\star\top} \boldsymbol{\Sigma} \boldsymbol{x}^\star + \sigma_n^2). \tag{2.21}$$

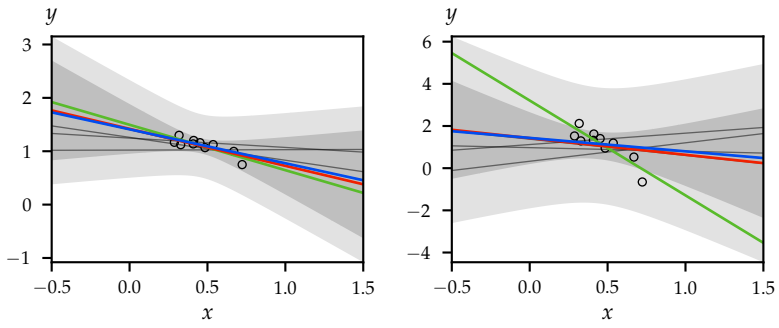


Figure 2.4: Comparison of linear regression (MLE), ridge regression (MAP estimate), and **Bayesian linear regression** when the data is generated according to

$$y \mid \boldsymbol{w}, \boldsymbol{x} \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}, \sigma_n^2).$$

The true mean is shown in blue, the MLE in green, and the MAP estimate in red. The dark gray area denotes the epistemic uncertainty of Bayesian linear regression and the light gray area the additional homoscedastic noise. Plausible realizations are shown in black. On the left, $\sigma_n = 0.15$. On the right, $\sigma_n = 0.7$.

Exercise 2.7: Bayesian linear regression

Suppose you are given the following observations

$$\boldsymbol{X} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 2 & 1 \\ 2 & 2 \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} 2.4 \\ 4.3 \\ 3.1 \\ 4.9 \end{bmatrix}$$

and assume the data follows a linear model with homoscedastic noise $\mathcal{N}(0, \sigma_n^2)$ where $\sigma_n^2 = 0.1$.

1. Find the maximum likelihood estimate $\hat{\boldsymbol{w}}_{\text{MLE}}$ given the data.
2. Now assume that we have a prior $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}; \boldsymbol{0}, \sigma_p^2 \boldsymbol{I})$ with $\sigma_p^2 = 0.05$. Find the MAP estimate $\hat{\boldsymbol{w}}_{\text{MAP}}$ given the data and the prior.
3. Use the posterior $p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y})$ to get a posterior prediction for the label $y^\star$ at $\boldsymbol{x}^\star = [3 \quad 3]^\top$. Report the mean and the variance of this prediction.
4. How would you have to change the prior $p(\boldsymbol{w})$ such that

$$\hat{\boldsymbol{w}}_{\text{MAP}} \to \hat{\boldsymbol{w}}_{\text{MLE}}?$$

▷ *Solution*

2.3 *Aleatoric and Epistemic Uncertainty*

The predictive posterior distribution from eq. (2.21) highlights a decomposition of uncertainty wherein $\boldsymbol{x}^{\star\top}\boldsymbol{\Sigma}\boldsymbol{x}^\star$ corresponds to the uncertainty about our model due to the lack of data (commonly referred to as the *epistemic uncertainty*) and σ_n^2 corresponds to the uncertainty about the labels that cannot be explained by the inputs and any model from the model class (commonly referred to as the *aleatoric uncertainty*, "irreducible noise", or simply "(label) noise").

A natural Bayesian approach is to represent epistemic uncertainty with a probability distribution over models. Intuitively, the variance of this distribution measures our uncertainty about the model and its mode corresponds to our current best (point) estimate. The distribution over weights of a linear model is one example, and we will continue to explore this approach for other models in the following chapters.

It is a practical modeling choice how much inaccuracy to attribute to epistemic or aleatoric uncertainty. Generally, when a poor model is used to explain a process, more inaccuracy has to be attributed to irreducible noise. For example, when a linear model is used to "explain" a non-linear process, most uncertainty is aleatoric as the model cannot explain the data well. As we use more expressive models, a larger portion of the uncertainty can be explained by the inputs.

Epistemic and aleatoric uncertainty can be formally defined in terms of the law of total variance (1.50),

$$\mathrm{Var}[y^\star \mid \boldsymbol{x}^\star] = \mathbb{E}_{\boldsymbol{\theta}}\left[\mathrm{Var}_{y^\star}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}]\right] + \mathrm{Var}_{\boldsymbol{\theta}}\left[\mathbb{E}_{y^\star}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}]\right]. \tag{2.22}$$

Here, the mean variability of predictions $y^\star$ averaged across all models $\boldsymbol{\theta}$ is the aleatoric uncertainty. In contrast, the variability of the mean prediction $y^\star$ under each model $\boldsymbol{\theta}$ is the epistemic uncertainty.

Exercise 2.8: Aleatoric and epistemic uncertainty of BLR

Prove for Bayesian linear regression that $\boldsymbol{x}^{\star\top}\boldsymbol{\Sigma}\boldsymbol{x}^\star$ is the epistemic uncertainty and σ_n^2 the aleatoric uncertainty in $y^\star$ under the decomposition of eq. (2.22).

▷ *Solution*

2.4 *Recursive Bayesian Updates*

A more general feature of supervised Bayesian learning, which often leads to efficient algorithms, is its recursive structure. Suppose we are

given a prior $p(\boldsymbol{\theta})$ and observations $y_{1:n}$. We define

$$p^{(t)}(\boldsymbol{\theta}) \doteq p(\boldsymbol{\theta} \mid y_{1:t}) \tag{2.23}$$

to be the posterior after the first t observations. We therefore have $p^{(0)}(\boldsymbol{\theta}) = p(\boldsymbol{\theta})$. Now, suppose that we have already computed $p^{(t)}(\boldsymbol{\theta})$ and observe y_{t+1}. We can then recursively update the posterior as follows,

$$\begin{aligned} p^{(t+1)}(\boldsymbol{\theta}) &= p(\boldsymbol{\theta} \mid y_{1:t+1}) \\ &\propto p(\boldsymbol{\theta} \mid y_{1:t}) \cdot p(y_{t+1} \mid \boldsymbol{\theta}, y_{1:t}) && \text{using Bayes' rule (1.59)} \\ &= p^{(t)}(\boldsymbol{\theta}) \cdot p(y_{t+1} \mid \boldsymbol{\theta}). && \text{using } y_{t+1} \perp y_{1:t} \mid \boldsymbol{\theta}\text{, see fig. 2.3} \end{aligned} \tag{2.24}$$

Intuitively, the posterior distribution "absorbs", or "summarizes" all seen data.

Thus, as data arrives online (i.e., in "real-time"), we can obtain the new posterior and use it to replace our prior. In general, the posterior distribution can be complicated. For Bayesian linear regression with a Gaussian prior and likelihood, however, the posterior distribution is also a Gaussian,

$$p^{(t)}(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}; \boldsymbol{\mu}^{(t)}, \boldsymbol{\Sigma}^{(t)}), \tag{2.25}$$

which can be stored efficiently using only $\mathcal{O}(d^2)$ parameters.

Exercise 2.9: Online Bayesian linear regression

1. Can you design an algorithm that updates the posterior (as opposed to recalculating it from scratch using eq. (2.13)) in a smarter way? The requirement is that the memory should not grow as $\mathcal{O}(t)$.
2. If d is large, computing the inverse every round is very expensive. Can you use the recursive structure you found in the previous question to bring down the computational complexity of every round to $\mathcal{O}(d^2)$?

The resulting efficient online algorithm is known as *online Bayesian linear regression*.

▷ *Solution*

This interpretation of Bayesian linear regression as an online algorithm also highlights similarities to other sequential models such as Kalman filters, which we discuss in chapter 3. In example 3.2, we show that online Bayesian linear regression is, in fact, an example of a Kalman filter.

2.5 *Non-linear Regression*

We can use linear regression not only to learn linear functions. The trick is to apply a non-linear transformation $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^e$ to the features $\boldsymbol{x}_i$, where d is the dimension of the input space and e is the dimension of the designed *feature space*. We denote the design matrix comprised of transformed features by $\boldsymbol{\Phi} \in \mathbb{R}^{n \times e}$. Note that if the feature transformation $\boldsymbol{\phi}$ is the identity function then $\boldsymbol{\Phi} = \boldsymbol{X}$.

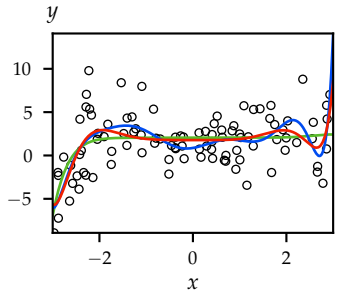


Figure 2.5: Applying linear regression with a feature space of polynomials of degree 10. The **least squares estimate** is shown in blue, **ridge regression** in red, and **lasso** in green.

Example 2.10: Polynomial regression

Let $\boldsymbol{\phi}(x) \doteq [x^2, x, 1]$ and $\boldsymbol{w} \doteq [a, b, c]$. Then the function that our model learns is given as

$$f = ax^2 + bx + c.$$

Thus, our model can exactly represent all polynomials up to degree 2.

However, to learn polynomials of degree m in d input dimensions, we need to apply the nonlinear transformation

$$\begin{aligned}\boldsymbol{\phi}(\boldsymbol{x}) = [1, x_1, \ldots, x_d, x_1^2, \ldots, x_d^2, x_1 \cdot x_2, \ldots, x_{d-1} \cdot x_d,\\ \ldots,\\ x_{d-m+1} \cdot \cdots \cdot x_d].\end{aligned}$$

Note that the feature dimension e is $\sum_{i=0}^{m} \binom{d+i-1}{i} = \Theta(d^m)$.[5] Thus, the dimension of the feature space grows exponentially in the degree of polynomials and input dimensions. Even for relatively small m and d, this becomes completely unmanageable.

The example of polynomials highlights that it may be inefficient to keep track of the weights $\boldsymbol{w} \in \mathbb{R}^e$ when e is large, and that it may be useful to instead consider a reparameterization which is of dimension n rather than of the feature dimension.

[5] Observe that the vector contains $\binom{d+i-1}{i}$ monomials of degree i as this is the number of ways to choose i times from d items with replacement and without consideration of order. To see this, consider the following encoding: We take a sequence of $d + i - 1$ spots. Selecting any subset of i spots, we interpret the remaining $d - 1$ spots as "barriers" separating each of the d items. The selected spots correspond to the number of times each item has been selected. For example, if 2 items are to be selected out of a total of 4 items with replacement, one possible configuration is "$\circ$ || $\circ$ |" where $\circ$ denotes a selected spot and | denotes a barrier. This configuration encodes that the first and third item have each been chosen once. The number of possible configurations — each encoding a unique outcome — is therefore $\binom{d+i-1}{i}$.

2.6 *Function-space View*

Let us look at Bayesian linear regression through a different lens. Previously, we have been interpreting it as a distribution over the weights $\boldsymbol{w}$ of a linear function $\boldsymbol{f} = \boldsymbol{\Phi}\boldsymbol{w}$. The key idea is that for a finite set of inputs (ensuring that the design matrix is well-defined), we can equivalently consider a distribution directly over the estimated function values $\boldsymbol{f}$.

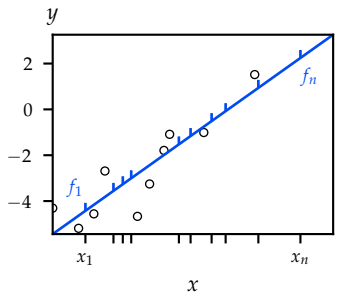


Figure 2.6: An illustration of the function-space view. The model is described by the points (x_i, f_i).

You may say, that nothing has changed — and you would be right: that is precisely the point. However, instead of considering a prior over the weights $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \sigma_p^2 \boldsymbol{I})$ as we have done previously, we now impose a prior directly on the values of our model at the observations. Using that Gaussians are closed under linear maps (1.123), we obtain the equivalent prior

$$\boldsymbol{f} \mid \boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\Phi}\mathbb{E}[\boldsymbol{w}], \boldsymbol{\Phi}\mathrm{Var}[\boldsymbol{w}]\boldsymbol{\Phi}^\top) = \mathcal{N}(\boldsymbol{0}, \underbrace{\sigma_p^2 \boldsymbol{\Phi}\boldsymbol{\Phi}^\top}_{\boldsymbol{K}}) \tag{2.26}$$

where $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ is the so-called *kernel matrix*. Observe that the entries of the kernel matrix can be expressed as $\boldsymbol{K}(i,j) = \sigma_p^2 \cdot \boldsymbol{\phi}(\boldsymbol{x}_i)^\top \boldsymbol{\phi}(\boldsymbol{x}_j)$.

The kernel matrix $\boldsymbol{K}$ has entries only for the finite set of observed inputs. However, in principle, we could have observed any input, and this motivates the definition of the *kernel function*

$$k(\boldsymbol{x}, \boldsymbol{x}') \doteq \sigma_p^2 \cdot \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}') \tag{2.27}$$

for arbitrary inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$. A kernel matrix is simply a finite "view" of the kernel function,

$$\boldsymbol{K} = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix} \tag{2.28}$$

Observe that by definition of the kernel matrix in eq. (2.26), the kernel matrix is a covariance matrix and the kernel function measures the covariance of the function values $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ given inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$. That is,

$$k(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{Cov}\big[f(\boldsymbol{x}), f(\boldsymbol{x}')\big]. \tag{2.29}$$

Moreover, note that we have reformulated[6] the learning algorithm such that the feature space is now *implicit* in the choice of kernel, and the kernel is defined by inner products of (nonlinearly transformed) inputs. In other words, the choice of kernel implicitly determines the class of functions that $\boldsymbol{f}$ is sampled from (without expressing the functions explicitly in closed-form), and encodes our prior beliefs. This is known as the *kernel trick*.

[6] we often say "kernelized"

2.6.1 Learning and Inference

We have already kernelized the Bayesian linear regression prior. The posterior distribution $\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{y}$ is again Gaussian due to the closedness properties of Gaussians, analogously to our derivation of the prior kernel matrix in eq. (2.26).

It remains to show that we can also rely on the kernel trick during inference. Given the test point $\boldsymbol{x}^\star$, we define

$$\tilde{\boldsymbol{\Phi}} \doteq \begin{bmatrix} \boldsymbol{\Phi} \\ \boldsymbol{\phi}(\boldsymbol{x}^\star)^\top \end{bmatrix}, \quad \tilde{\boldsymbol{y}} \doteq \begin{bmatrix} \boldsymbol{y} \\ y^\star \end{bmatrix}, \quad \tilde{\boldsymbol{f}} \doteq \begin{bmatrix} \boldsymbol{f} \\ f^\star \end{bmatrix}.$$

We immediately obtain $\tilde{\boldsymbol{f}} = \tilde{\boldsymbol{\Phi}}\boldsymbol{w}$. Analogously to our analysis of inference from the weight-space view, we add the label noise to obtain the estimate $\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{f}} + \tilde{\boldsymbol{\epsilon}}$ where $\tilde{\boldsymbol{\epsilon}} \doteq [\epsilon_1 \ \cdots \ \epsilon_n \ \epsilon^\star]^\top \sim \mathcal{N}(\boldsymbol{0}, \sigma_n^2 \boldsymbol{I})$ is the independent label noise. Applying the same reasoning as we did for the prior, we obtain

$$\tilde{\boldsymbol{f}} \mid \boldsymbol{X}, \boldsymbol{x}^\star \sim \mathcal{N}(\boldsymbol{0}, \tilde{\boldsymbol{K}}) \tag{2.30}$$

where $\tilde{\boldsymbol{K}} \doteq \sigma_p^2 \tilde{\boldsymbol{\Phi}} \tilde{\boldsymbol{\Phi}}^\top$. Adding the label noise yields

$$\tilde{\boldsymbol{y}} \mid \boldsymbol{X}, \boldsymbol{x}^\star \sim \mathcal{N}(\boldsymbol{0}, \tilde{\boldsymbol{K}} + \sigma_n^2 \boldsymbol{I}). \tag{2.31}$$

Finally, we can conclude from the closedness of Gaussian random vectors under conditional distributions (1.118) that the predictive posterior $y^\star \mid \boldsymbol{x}^\star, \boldsymbol{X}, \boldsymbol{y}$ follows again a normal distribution. We will do a full derivation of the posterior and predictive posterior in section 4.1.

2.6.2 *Efficient Polynomial Regression*

But how does the kernel trick address our concerns about efficiency raised in section 2.5? After all, computing the kernel for a feature space of dimension e still requires computing sums of length e which is prohibitive when e is large. The kernel trick opens up a couple of new doors for us:

1. For certain feature transformations $\boldsymbol{\phi}$, we may be able to find an easier to compute expression equivalent to $\boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}')$.
2. If this is not possible, we could approximate the inner product by an easier to compute expression.
3. Or, alternatively, we may decide not to care very much about the exact feature transformation and simply experiment with kernels that induce *some* feature space (which may even be infinitely dimensional).

We will explore the third approach when we revisit kernels in section 4.3. A polynomial feature transformation can be computed efficiently in closed-form.

Fact 2.11. *For the polynomial feature transformation $\boldsymbol{\phi}$ up to degree m from example 2.10, it can be shown that*

$$\boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}') = (1 + \boldsymbol{x}^\top \boldsymbol{x}')^m \tag{2.32}$$

up to constant factors.

For example, for input dimension 2, the kernel $(1 + \boldsymbol{x}^\top \boldsymbol{x}')^2$ corresponds to the feature vector $\boldsymbol{\phi}(\boldsymbol{x}) = [1 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2 \;\; \sqrt{2}x_1x_2 \;\; x_1^2 \;\; x_2^2]^\top$.

3

Kalman Filters

Before we continue in chapter 4 with the function-space view of regression, we want to look at a seemingly different but very related problem. Kalman filters are an example of Bayesian learning and inference in the *state space model*, where we want to keep track of the state of an agent over time based on noisy observations. In this model, we have a sequence of (hidden) states $(\mathbf{X}_t)_{t\in\mathbb{N}_0}$ where $\mathbf{X}_t$ is in $\mathbb{R}^d$ and a sequence of observations $(\mathbf{Y}_t)_{t\in\mathbb{N}_0}$ where $\mathbf{Y}_t$ is in $\mathbb{R}^m$.

The process of keeping track of the state using noisy observations is also known as *Bayesian filtering* or *recursive Bayesian estimation*. We will discuss this process more broadly in the next section. A Kalman filter is simply a Bayes filter using a Gaussian distribution over the states and conditional linear Gaussians to describe the evolution of states and observations.

Definition 3.1 (Kalman filter)**.** A *Kalman filter* is specified by a Gaussian prior over the states,

$$\mathbf{X}_0 \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{3.1}$$

and a conditional linear Gaussian *motion model* and *sensor model*,

$$\mathbf{X}_{t+1} \doteq \boldsymbol{F}\mathbf{X}_t + \boldsymbol{\epsilon}_t \qquad \boldsymbol{F} \in \mathbb{R}^{d\times d}, \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_x), \tag{3.2}$$

$$\mathbf{Y}_t \doteq \boldsymbol{H}\mathbf{X}_t + \boldsymbol{\eta}_t \qquad \boldsymbol{H} \in \mathbb{R}^{m\times d}, \boldsymbol{\eta}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_y), \tag{3.3}$$

respectively. The motion model is sometimes also called *transition model* or *dynamics model*.

Crucially, Kalman filters assume that $\boldsymbol{F}$ and $\boldsymbol{H}$ are known. In general, $\boldsymbol{F}$ and $\boldsymbol{H}$ may depend on t. Also, $\boldsymbol{\epsilon}$ and $\boldsymbol{\eta}$ may have a non-zero mean, commonly called a "drift".

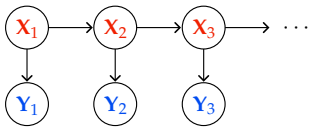


Figure 3.1: Directed graphical model of a Kalman filter with hidden states $\mathbf{X}_t$ and observables $\mathbf{Y}_t$.

Because Kalman filters use conditional linear Gaussians, which we have already seen in section 1.6.2, their joint distribution (over all variables) is also Gaussian. This means that predicting the future states

of a Kalman filter is simply inference with multivariate Gaussians. In Bayesian filtering, however, we do not only want to make predictions occasionally. In Bayesian filtering, we want to *keep track* of states, that is, predict the current state of an agent online.[1] To do this efficiently, we need to update our *belief* about the state of the agent recursively, similarly to our recursive Bayesian updates in Bayesian linear regression (see section 2.4).

[1] Here, *online* is common terminology to say that we want to perform inference at time t without exposure to times $t+1, t+2, \ldots$, so in "real-time".

From the directed graphical model of a Kalman filter shown in fig. 3.1, we can immediately gather the following conditional independence relations,[2]

[2] Alternatively, they follow from the definition of the motion and sensor models as linear updates.

$$\mathbf{X}_{t+1} \perp \mathbf{X}_{1:t-1}, \mathbf{Y}_{1:t-1} \mid \mathbf{X}_t, \tag{3.4}$$

$$\mathbf{Y}_t \perp \mathbf{X}_{1:t-1} \mid \mathbf{X}_t \tag{3.5}$$

$$\mathbf{Y}_t \perp \mathbf{Y}_{1:t-1} \mid \mathbf{X}_{t-1}. \tag{3.6}$$

The first conditional independence property is also known as the *Markov property*, which we will return to later in our discussion of Markov chains and Markov decision processes. This characterization of the Kalman filter, yields the following factorization of the joint distribution,

$$\begin{aligned} p(\boldsymbol{x}_{1:t}, \boldsymbol{y}_{1:t}) &= \prod_{i=1}^{t} p(\boldsymbol{x}_i \mid \boldsymbol{x}_{1:i-1}) p(\boldsymbol{y}_i \mid \boldsymbol{x}_{1:t}, \boldsymbol{y}_{1:i-1}) && \text{using the product rule (1.14)} \\ &= p(\boldsymbol{x}_1) p(\boldsymbol{y}_1 \mid \boldsymbol{x}_1) \prod_{i=2}^{t} p(\boldsymbol{x}_i \mid \boldsymbol{x}_{i-1}) p(\boldsymbol{y}_i \mid \boldsymbol{x}_i). && \text{using the conditional independence properties from (3.4), (3.5), and (3.6)} \end{aligned} \tag{3.7}$$

Example 3.2: Bayesian linear regression as a Kalman filter

Even though they arise from a rather different setting, it turns out that Kalman filters are a generalization of Bayesian linear regression! To see this, recall the online Bayesian linear regression algorithm from section 2.4. Observe that this algorithm performs Bayesian filtering. That is, it keeps track of the weights given noisy labels y_i. Moreover, we have used a Gaussian prior and likelihood. This is precisely the setting of a Kalman filter!

More concretely, our motion model is given as

- $\boldsymbol{x}_t = \boldsymbol{w}^{(t)}$;
- $\boldsymbol{F} = \boldsymbol{I}$; and
- $\boldsymbol{\epsilon} = \mathbf{0}$.

Our (time-dependent) sensor model is given as

- $\boldsymbol{H}_t = \operatorname{diag}\{\boldsymbol{x}_t\}$; and
- $\boldsymbol{\eta}_t = \boldsymbol{\epsilon}_t$.

Using the sensor model (3.3), this yields d-dimensional labels,

$$\boldsymbol{y}_t = \text{diag}\{\boldsymbol{x}_t\}\boldsymbol{w}^{(t)} + \boldsymbol{\epsilon}_t \in \mathbb{R}^d. \tag{3.8}$$

Observe that the label y_t can simply be recovered by summing over all elements of the vector,

$$y_t = \sum_{i=1}^{d} \boldsymbol{y}_t(i). \tag{3.9}$$

3.1 *Bayesian Filtering*

As we have already mentioned, *Bayesian filtering* is the process of keeping track of an agent's state using noisy observations. Bayesian filtering is described by the following recursive scheme with the two phases, *conditioning* (also called *update*) and *prediction*.

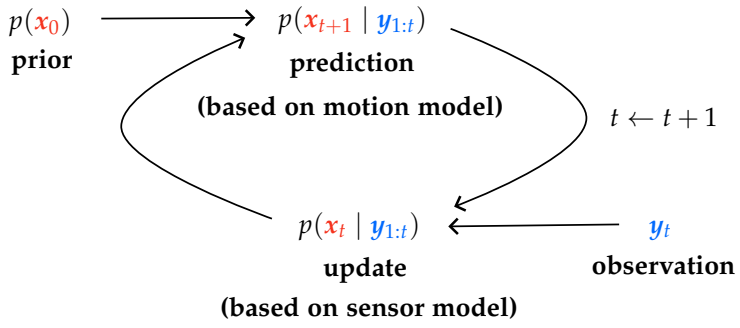


Figure 3.2: Schematic view of Bayesian filtering.

Algorithm 3.3: Bayesian filtering

start with a prior over initial states $p(\boldsymbol{x}_0)$
for $t = 1$ **to** ∞ **do**
    assume we have $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t-1})$
    compute $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t})$ using the new observation $\boldsymbol{y}_t$ (conditioning)
    compute $p(\boldsymbol{x}_{t+1} \mid \boldsymbol{y}_{1:t})$ (prediction)

Let us consider the conditioning step first:

$$\begin{aligned} p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}) &= \frac{1}{Z} p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t-1}) p(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{y}_{1:t-1}) && \text{using Bayes' rule (1.59)} \\ &= \frac{1}{Z} p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t-1}) p(\boldsymbol{y}_t \mid \boldsymbol{x}_t). && \text{using the conditional independence structure (3.6)} \end{aligned} \tag{3.10}$$

For the prediction step, we obtain,

$$\begin{aligned} p(\boldsymbol{x}_{t+1} \mid \boldsymbol{y}_{1:t}) &= \int p(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t \mid \boldsymbol{y}_{1:t})\, d\boldsymbol{x}_t && \text{using the sum rule (1.10)} \\ &= \int p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, y_{1:t}) p(\boldsymbol{x}_t \mid y_{1:t})\, d\boldsymbol{x}_t && \text{using the product rule (1.14)} \\ &= \int p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t) p(\boldsymbol{x}_t \mid y_{1:t})\, d\boldsymbol{x}_t. \qquad (3.11) && \text{using the conditional independence structure (3.4)} \end{aligned}$$

In general, these distributions can be very complicated, but for Gaussians (i.e., Kalman filters) they can be expressed in closed-form.

Remark 3.4: Bayesian smoothing

Bayesian smoothing is a closely related task to Bayesian filtering. While Bayesian filtering methods estimate the current state based only on observations obtained before and at the current time step, Bayesian smoothing computes the distribution of $\mathbf{X}_k \mid \boldsymbol{y}_{1:t}$ where $t > k$. Note that if $k = t$, then Bayesian smoothing coincides with Bayesian filtering.

Analogously to eq. (3.10),

$$p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:t}) \propto p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k}) p(\boldsymbol{y}_{k+1:t} \mid \boldsymbol{x}_k). \qquad (3.12)$$

If we assume a Gaussian prior and conditional Gaussian transition and dynamics models (this is called *Kalman smoothing*), then by the closedness properties of Gaussians, $\mathbf{X}_k \mid \boldsymbol{y}_{1:t}$ is a Gaussian. Indeed, all terms of eq. (3.12) are Gaussian PDFs and as seen in eq. (1.116), the product of two Gaussian PDFs is again proportional to a Gaussian PDF.

The first term, $\mathbf{X}_k \mid \boldsymbol{y}_{1:k}$, is the marginal posterior of the hidden states of the Kalman filter which can be obtained with Bayesian filtering.

By conditioning on $\mathbf{X}_{k+1}$, we have for the second term,

$$\begin{aligned} p(\boldsymbol{y}_{k+1:t} \mid \boldsymbol{x}_k) &= \int p(\boldsymbol{y}_{k+1:t} \mid \boldsymbol{x}_k, \boldsymbol{x}_{k+1}) p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)\, d\boldsymbol{x}_{k+1} && \text{using the sum rule (1.10) and product rule (1.14)} \\ &= \int p(\boldsymbol{y}_{k+1:t} \mid \boldsymbol{x}_{k+1}) p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)\, d\boldsymbol{x}_{k+1} && \text{using the conditional independence structure (3.5)} \\ &= \int p(\boldsymbol{y}_{k+1} \mid \boldsymbol{x}_{k+1}) p(\boldsymbol{y}_{k+2:t} \mid \boldsymbol{x}_{k+1}) p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)\, d\boldsymbol{x}_{k+1} && \text{using the conditional independence structure (3.6)} \\ & \qquad (3.13) \end{aligned}$$

Let us have a look at the terms in the product:

- $p(\boldsymbol{y}_{k+1} \mid \boldsymbol{x}_{k+1})$ is obtained from the sensor model,
- $p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)$ is obtained from the transition model, and
- $p(\boldsymbol{y}_{k+2:t} \mid \boldsymbol{x}_{k+1})$ can be computed recursively backwards in time.

This recursion results in linear equations resembling a Kalman filter running backwards in time.

Thus, in the setting of Kalman smoothing, both factors of eq. (3.12) can be computed efficiently: one using a (forward) Kalman filter; the other using a "backward" Kalman filter. More concretely, in time $\mathcal{O}(t)$, we can compute the two factors for all $k \in [t]$. This approach is known as *two-filter smoothing* or the *forward-backward algorithm*.

3.2 *Kalman Update*

Before introducing the general Kalman update, let us consider a simple example.

Example 3.5: Random walk in 1d

We use the simple motion and sensor models,[3]

$$X_{t+1} \mid x_t \sim \mathcal{N}(x_t, \sigma_x^2), \tag{3.14a}$$
$$Y_t \mid x_t \sim \mathcal{N}(x_t, \sigma_y^2). \tag{3.14b}$$

Let $X_t \mid y_{1:t} \sim \mathcal{N}(\mu_t, \sigma_t^2)$ be our belief at time t. It can be shown that Bayesian filtering yields the belief $X_{t+1} \mid y_{1:t+1} \sim \mathcal{N}(\mu_{t+1}, \sigma_{t+1}^2)$ at time $t+1$ where

$$\mu_{t+1} \doteq \frac{\sigma_y^2 \mu_t + (\sigma_t^2 + \sigma_x^2) y_{t+1}}{\sigma_t^2 + \sigma_x^2 + \sigma_y^2}, \quad \sigma_{t+1}^2 \doteq \frac{(\sigma_t^2 + \sigma_x^2)\sigma_y^2}{\sigma_t^2 + \sigma_x^2 + \sigma_y^2}. \tag{3.15}$$

Although looking intimidating at first, this update has a very natural interpretation. Let us define the following quantity,

$$\lambda \doteq \frac{\sigma_t^2 + \sigma_x^2}{\sigma_t^2 + \sigma_x^2 + \sigma_y^2} = 1 - \frac{\sigma_y^2}{\sigma_t^2 + \sigma_x^2 + \sigma_y^2} \in [0, 1]. \tag{3.16}$$

Using λ, we can write the updated mean as a convex combination of the previous mean and the observation,

$$\mu_{t+1} = (1 - \lambda)\mu_t + \lambda y_{t+1} \tag{3.17}$$
$$= \mu_t + \lambda(y_{t+1} - \mu_t). \tag{3.18}$$

Intuitively, λ is a form of "gain" that influences how much of the new information should be incorporated into the updated mean. For this reason, λ is also called *Kalman gain*.

The updated variance can similarly be rewritten,

$$\sigma_{t+1}^2 = \lambda \sigma_y^2 = (1 - \lambda)(\sigma_t^2 + \sigma_x^2). \tag{3.19}$$

[3] This corresponds to $\boldsymbol{F} = \boldsymbol{H} = \boldsymbol{I}$ and a drift of 0.

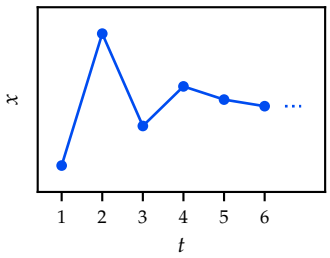


Figure 3.3: Hidden states during a random walk in one dimension.

In particular, observe that if $\mu_t = y_{t+1}$ (i.e., we observe our prediction), we have $\mu_{t+1} = \mu_t$ as there is no new information. Similarly, for $\sigma_y^2 \to \infty$ (i.e., we do not trust our observations), we have

$$\lambda \to 0, \quad \mu_{t+1} = \mu_t, \quad \sigma_{t+1}^2 = \sigma_t^2 + \sigma_x^2.$$

In contrast, for $\sigma_y^2 \to 0$, we have

$$\lambda \to 1, \quad \mu_{t+1} = y_{t+1}, \quad \sigma_{t+1}^2 = 0.$$

Exercise 3.6: Kalman update

Derive the predictive distribution $X_{t+1} \mid y_{1:t+1}$ (3.15) of the Kalman filter described in the above example using your knowledge about multivariate Gaussians from section 1.6.
Hint: First compute the predictive distribution $X_{t+1} \mid y_{1:t}$.

▷ *Solution*

The general formulas for the *Kalman update* follow the same logic. Given the prior belief $\mathbf{X}_t \mid \boldsymbol{y}_{1:t} \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$, we have

$$\mathbf{X}_{t+1} \mid \boldsymbol{y}_{1:t+1} \sim \mathcal{N}(\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1}) \quad \text{where} \tag{3.20}$$

$$\boldsymbol{\mu}_{t+1} \doteq \boldsymbol{F}\boldsymbol{\mu}_t + \boldsymbol{K}_{t+1}(\boldsymbol{y}_{t+1} - \boldsymbol{H}\boldsymbol{F}\boldsymbol{\mu}_t), \tag{3.21}$$

$$\boldsymbol{\Sigma}_{t+1} \doteq (\boldsymbol{I} - \boldsymbol{K}_{t+1}\boldsymbol{H})(\boldsymbol{F}\boldsymbol{\Sigma}_t\boldsymbol{F}^\top + \boldsymbol{\Sigma}_x). \tag{3.22}$$

Hereby, $\boldsymbol{K}_{t+1}$ is the *Kalman gain*,

$$\boldsymbol{K}_{t+1} \doteq (\boldsymbol{F}\boldsymbol{\Sigma}_t\boldsymbol{F}^\top + \boldsymbol{\Sigma}_x)\boldsymbol{H}^\top(\boldsymbol{H}(\boldsymbol{F}\boldsymbol{\Sigma}_t\boldsymbol{F}^\top + \boldsymbol{\Sigma}_x)\boldsymbol{H}^\top + \boldsymbol{\Sigma}_y)^{-1}. \tag{3.23}$$

Note that $\boldsymbol{\Sigma}_t$ and $\boldsymbol{K}_t$ can be computed offline as they are independent of the observation $\boldsymbol{y}_{t+1}$. $\boldsymbol{F}\boldsymbol{\mu}_t$ represents the expected state at time $t+1$, and hence, $\boldsymbol{H}\boldsymbol{F}\boldsymbol{\mu}_t$ corresponds to the expected observation. Therefore, the term $\boldsymbol{y}_{t+1} - \boldsymbol{H}\boldsymbol{F}\boldsymbol{\mu}_t$ measures the error in the predicted observation and the Kalman gain $\boldsymbol{K}_{t+1}$ appears as a measure of relevance of the new observation compared to the prediction.

3.3 *Predicting*

Using now that the marginal posterior of $\mathbf{X}_t$ is a Gaussian due to the closedness properties of Gaussians, we have

$$\mathbf{X}_{t+1} \mid \boldsymbol{y}_{1:t} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_{t+1}, \hat{\boldsymbol{\Sigma}}_{t+1}), \tag{3.24}$$

and it suffices to compute the prediction mean $\hat{\boldsymbol{\mu}}_{t+1}$ and covariance matrix $\hat{\boldsymbol{\Sigma}}_{t+1}$.

For the mean,

$$\begin{aligned}
\hat{\boldsymbol{\mu}}_{t+1} &= \mathbb{E}[\boldsymbol{x}_{t+1} \mid \boldsymbol{y}_{1:t}] && \\
&= \mathbb{E}[\boldsymbol{F}\boldsymbol{x}_t + \boldsymbol{\epsilon}_t \mid \boldsymbol{y}_{1:t}] && \text{using the motion model (3.2)} \\
&= \boldsymbol{F}\mathbb{E}[\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}] && \text{using linearity of expectation (1.24) and } \mathbb{E}[\boldsymbol{\epsilon}_t] = \mathbf{0} \\
&= \boldsymbol{F}\boldsymbol{\mu}_t. && (3.25) \quad \text{using the mean of the Kalman update}
\end{aligned}$$

For the covariance matrix,

$$\begin{aligned}
\hat{\boldsymbol{\Sigma}}_{t+1} &= \mathbb{E}\left[(\boldsymbol{x}_{t+1} - \hat{\boldsymbol{\mu}}_{t+1})(\boldsymbol{x}_{t+1} - \hat{\boldsymbol{\mu}}_{t+1})^\top \;\middle|\; \boldsymbol{y}_{1:t}\right] && \text{using the definition of the covariance matrix (1.46)} \\
&= \boldsymbol{F}\mathbb{E}\left[(\boldsymbol{x}_t - \boldsymbol{\mu}_t)(\boldsymbol{x}_t - \boldsymbol{\mu}_t)^\top \;\middle|\; \boldsymbol{y}_{1:t}\right]\boldsymbol{F}^\top + \mathbb{E}\left[\boldsymbol{\epsilon}_t\boldsymbol{\epsilon}_t^\top\right] && \text{using (3.25), the motion model (3.2) and that } \boldsymbol{\epsilon}_t \text{ is independent of the observations} \\
&= \boldsymbol{F}\boldsymbol{\Sigma}_t\boldsymbol{F}^\top + \boldsymbol{\Sigma}. && (3.26)
\end{aligned}$$

Exercise 3.7: Parameter estimation using Kalman filters

Suppose that we want to estimate the value of an unknown constant π using uncorrelated measurements

$$y_t = \pi + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \sigma_y^2).$$

1. How can this problem be formulated as a Kalman filter? Compute closed form expressions for the Kalman gain and the variance of the estimation error σ_t^2 in terms of t, σ_y^2, and σ_0^2.
2. What is the Kalman filter when $t \to \infty$?
3. Suppose that one has no prior assumptions on π, meaning that $\mu_0 = 0$ and $\sigma_0^2 \to \infty$. Which well-known estimator does the Kalman filter reduce to in this case?

▷ *Solution*

Readings

Kalman filters and related models are often called *temporal models*. For a broader look at such models, read chapter 15 of "Artificial intelligence: a modern approach" (Russell and Norvig, 2002).

4

Gaussian Processes

We obtain Gaussian processes by extending the function-space view of Bayesian linear regression which we introduced in section 2.6 to infinite domains.[1] We are still concerned with the problem of estimating the value of a function $f : \mathcal{X} \to \mathbb{R}$ at arbitrary points $\boldsymbol{x}^\star \in \mathcal{X}$ given training data $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$, where the labels are assumed to be corrupted by homoscedastic Gaussian noise,

$$y_i = f(\boldsymbol{x}_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma_n^2).$$

As in chapter 2 on Bayesian linear regression, we denote by $\boldsymbol{X}$ the design matrix (collection of training inputs) and by $\boldsymbol{y}$ the vector of training labels.

Definition 4.1 (Gaussian process, GP)**.** A *Gaussian process* is an infinite set of random variables such that any finite number of them are jointly Gaussian.

We use a set $\mathcal{X}$ to index the collection of random variables. A Gaussian process is characterized by a *mean function* $\mu : \mathcal{X} \to \mathbb{R}$ and a *covariance function* (or *kernel function*) $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that for any $A \doteq \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\} \subseteq \mathcal{X}$, we have

$$\boldsymbol{f}_A \doteq [f_{\boldsymbol{x}_1} \ \cdots \ f_{\boldsymbol{x}_m}]^\top \sim \mathcal{N}(\boldsymbol{\mu}_A, \boldsymbol{K}_{AA}) \tag{4.1}$$

where

$$\boldsymbol{\mu}_A \doteq \begin{bmatrix} \mu(\boldsymbol{x}_1) \\ \vdots \\ \mu(\boldsymbol{x}_m) \end{bmatrix}, \quad \boldsymbol{K}_{AA} \doteq \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_m) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_m, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_m, \boldsymbol{x}_m) \end{bmatrix}. \tag{4.2}$$

We write $f \sim \mathcal{GP}(\mu, k)$. In particular, given a mean function, covariance function, and using the homoscedastic noise assumption,

$$y^\star \mid \boldsymbol{x}^\star, \mu, k \sim \mathcal{N}(\mu(\boldsymbol{x}^\star), k(\boldsymbol{x}^\star, \boldsymbol{x}^\star) + \sigma_n^2). \tag{4.3}$$

[1] In section 2.6, our domain was given by the set of weights (weight-space view) or the set of (noisy) observations (function-space view).

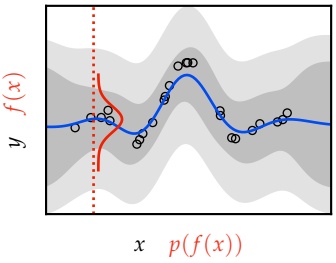


Figure 4.1: A Gaussian process can be interpreted as an infinite-dimensional Gaussian over functions. At any location x in the domain, this yields a distribution over values $f(x)$ shown in red. The blue line corresponds to the MAP estimate (i.e., mean function of the Gaussian process), the dark gray region corresponds to the epistemic uncertainty and the light gray region denotes the additional aleatoric uncertainty.

The random variable $f_{\boldsymbol{x}}$ represents the value of the function $f(\boldsymbol{x})$ at location $\boldsymbol{x} \in \mathcal{X}$, we thus write $f(\boldsymbol{x}) \doteq f_{\boldsymbol{x}}$. Intuitively, a Gaussian process can be interpreted as a normal distribution over functions — and is therefore often called an infinite-dimensional Gaussian.

Commonly, for notational simplicity, the mean function is taken to be zero. Note that for a fixed mean this is not a restriction, as we can simply apply the zero-mean Gaussian process to the difference between the mean and the observations.[2]

[2] For alternative ways of representing a mean function, refer to section 2.7 of "Gaussian processes for machine learning" (Williams and Rasmussen, 2006).

Readings

Chapter 2 of "Gaussian processes for machine learning" (Williams and Rasmussen, 2006) provides additional background.

4.1 Learning and Inference

First, let us look at learning and inference in the context of Gaussian processes. Given a prior $f \sim \mathcal{GP}(\mu, k)$ and (noisy) observations $y_i = f(\boldsymbol{x}_i) + \epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$ at locations $A \doteq \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$, we can write the joint distribution of the observations $y_{1:n}$ and the noise-free prediction $f^\star$ at a test point $\boldsymbol{x}^\star$ as

$$\begin{bmatrix} \boldsymbol{y} \\ f^\star \end{bmatrix} \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{K}}), \quad \text{where} \tag{4.4}$$

$$\tilde{\boldsymbol{\mu}} \doteq \begin{bmatrix} \boldsymbol{\mu}_A \\ \mu(\boldsymbol{x}^\star) \end{bmatrix}, \quad \tilde{\boldsymbol{K}} \doteq \begin{bmatrix} \boldsymbol{K}_{AA} + \sigma_n^2 \boldsymbol{I} & \boldsymbol{k}_{\boldsymbol{x}^\star, A} \\ \boldsymbol{k}_{\boldsymbol{x}^\star, A}^\top & k(\boldsymbol{x}^\star, \boldsymbol{x}^\star) \end{bmatrix}, \quad \boldsymbol{k}_{\boldsymbol{x}, A} \doteq \begin{bmatrix} k(\boldsymbol{x}, \boldsymbol{x}_1) \\ \vdots \\ k(\boldsymbol{x}, \boldsymbol{x}_n) \end{bmatrix}. \tag{4.5}$$

Deriving the conditional distribution using (1.118), we obtain that the Gaussian process posterior is given by

$$f \mid \boldsymbol{x}_{1:n}, y_{1:n} \sim \mathcal{GP}(\mu', k'), \quad \text{where} \tag{4.6}$$

$$\mu'(\boldsymbol{x}) \doteq \mu(\boldsymbol{x}) + \boldsymbol{k}_{\boldsymbol{x}, A}^\top (\boldsymbol{K}_{AA} + \sigma_n^2 \boldsymbol{I})^{-1} (\boldsymbol{y}_A - \boldsymbol{\mu}_A), \tag{4.7}$$

$$k'(\boldsymbol{x}, \boldsymbol{x}') \doteq k(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{k}_{\boldsymbol{x}, A}^\top (\boldsymbol{K}_{AA} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{k}_{\boldsymbol{x}', A}. \tag{4.8}$$

Observe that analogously to Bayesian linear regression, the posterior covariance can only decrease when conditioning on additional data, and is independent of the observations y_i.

We already studied inference in the function-space in section 2.6.1 in the context of Bayesian linear regression, but did not make the predictive posterior explicit. Using eq. (4.6) and our assumption of homoscedastic noise, the predictive posterior at the test point $\boldsymbol{x}^\star$ is

$$y^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, y_{1:n} \sim \mathcal{N}(\mu^\star, k^\star + \sigma_n^2). \tag{4.9}$$

4.2 Sampling

Often, we are not interested in the full predictive posterior distribution, but merely want to obtain samples of our Gaussian process model. We will briefly examine two approaches.

1. For the first approach, consider a discretized subset of points

   $$\boldsymbol{f} \doteq [f_1, \dots, f_n]$$

   that we want to sample.[3] Note that $\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K})$. We have already seen in eq. (1.125) that

   $$\boldsymbol{f} = \boldsymbol{K}^{1/2}\boldsymbol{\epsilon} \tag{4.10}$$

   where $\boldsymbol{K}^{1/2}$ is the square root of $\boldsymbol{K}$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is standard Gaussian noise.
2. For the second approach, recall the product rule (1.14),

   $$p(f_1, \dots, f_n) = \prod_{i=1}^{n} p(f_i \mid f_{1:i-1}).$$

   That is the joint distribution factorizes neatly into a product where each factor only depends on the "outcomes" of preceding factors. We can therefore obtain samples one-by-one, each time conditioning on one more observation:

   $$\begin{aligned} f_1 &\sim p(f_1) \\ f_2 &\sim p(f_2 \mid f_1) \\ f_3 &\sim p(f_3 \mid f_1, f_2) \\ &\vdots \end{aligned} \tag{4.11}$$

   This general approach is known as *forward sampling*.

[3] For example, if we want to render the function, the length of this vector could be guided by the screen resolution.

We will discuss approximate sampling methods in section 4.5.

4.3 Kernel Functions

We have previously introduced kernels in section 2.6 in the context of a function-space view of Bayesian linear regression. There we have already seen that kernel functions define the "shape" of the functions that are realized from a Gaussian distribution over functions, namely a Gaussian process. In this section, we generalize the notion of kernels to encompass function classes beyond linear and polynomial functions.

Definition 4.2 (Kernel function)**.** A *kernel function* k is defined as

$$k(\boldsymbol{x}, \boldsymbol{x}') \doteq \operatorname{Cov}\big[f(\boldsymbol{x}), f(\boldsymbol{x}')\big] \tag{4.12}$$

for locations $\boldsymbol{x}$ and $\boldsymbol{x}'$ such that

- $k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}', \boldsymbol{x})$ for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$ (symmetry), and
- $\boldsymbol{K}_{AA}$ is positive semi-definite for any $A \subseteq \mathcal{X}$.

We say that a kernel function is *positive definite* if $\boldsymbol{K}_{AA}$ is positive definite for any $A \subseteq \mathcal{X}$.

The two defining conditions ensure that for any $A \subseteq \mathcal{X}$, $\boldsymbol{K}_{AA}$ is a valid covariance matrix. Indeed, it can be shown that a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ satisfying the above two conditions corresponds to a (not necessarily finite dimensional) feature representation in some feature space.[4] The corresponding feature space is a reproducing kernel Hilbert space (RKHS) which we will study in greater detail in section 4.3.4.

Intuitively, the kernel function evaluated at locations $\boldsymbol{x}$ and $\boldsymbol{x}'$ describes how $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ are related. If $\boldsymbol{x}$ and $\boldsymbol{x}'$ are "close", then $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ are usually taken to be positively correlated, encoding a "smooth" function.

4.3.1 Common Kernels

We will now define some commonly used kernels. Often an additional factor σ^2 (*output scale*) is added, which we assume to be 1 for simplicity.

1. The *linear kernel* is defined as

$$k(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\phi}) \doteq \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}') \tag{4.13}$$

   where $\boldsymbol{\phi}$ is a nonlinear transformation as introduced in section 2.5 or the identity.

   Remark 4.3: GPs with linear kernel and BLR

   A Gaussian process with a linear kernel is equivalent to Bayesian linear regression. This follows directly from the function-space view of Bayesian linear regression (see section 2.6) and comparing the derived kernel function (2.27) with the definition of the linear kernel (4.13).

2. The *Gaussian kernel* (also known as *squared exponential kernel* or *radial basis function (RBF) kernel*) is defined as

$$k(\boldsymbol{x}, \boldsymbol{x}'; \ell) \doteq \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{2\ell^2}\right) \tag{4.14}$$

   where ℓ is its *length scale*. The larger the length scale ℓ, the smoother the resulting functions.[5]

[4] That is, there exists a feature transformation $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^e$ where $e \in \mathbb{R} \cup \{\infty\}$ such that $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}')$. This is known as *Mercer's theorem*, see theorem 4.2 of "Gaussian processes for machine learning" (Williams and Rasmussen, 2006).

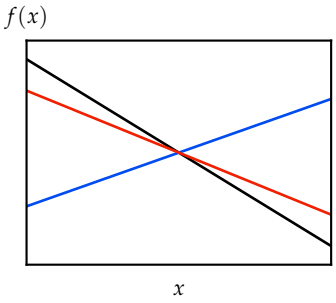


Figure 4.2: Functions sampled according to a Gaussian process with a linear kernel and $\phi = \mathrm{id}$.

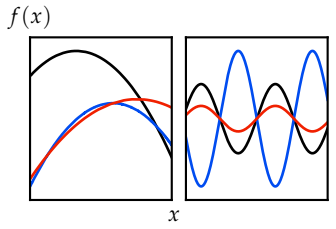


Figure 4.3: Functions sampled according to a Gaussian process with a linear kernel and $\boldsymbol{\phi}(x) = [1, x, x^2]$ (left) and $\phi(x) = \sin(x)$ (right).

[5] As the length scale is increased, the exponent of the exponential increases, resulting in a higher dependency between locations.

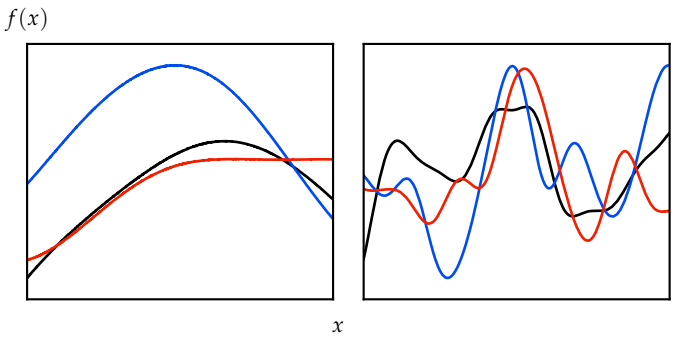


Figure 4.4: Functions sampled according to a Gaussian process with a Gaussian kernel and length scales $\ell = 5$ (left) and $\ell = 1$ (right).

Exercise 4.4: Feature space of Gaussian kernel

1. Show that the univariate Gaussian kernel with length scale $\ell = 1$ implicitly defines a feature space with basis vectors

$$\boldsymbol{\phi}(x) = \begin{bmatrix} \phi_0(x) \\ \phi_1(x) \\ \vdots \end{bmatrix} \quad \text{with} \quad \phi_j(x) = \frac{1}{\sqrt{j!}} e^{-\frac{x^2}{2}} x^j.$$

   Hint: Use the Taylor series expansion of the exponential function, $e^x = \sum_{j=0}^{\infty} \frac{x^j}{j!}$.
2. Note that the vector $\boldsymbol{\phi}(x)$ is ∞-dimensional. Thus, taking the function-space view allows us to perform regression in an infinite-dimensional feature space. What is the effective dimension when regressing n univariate data points with a Gaussian kernel?

▷ *Solution*

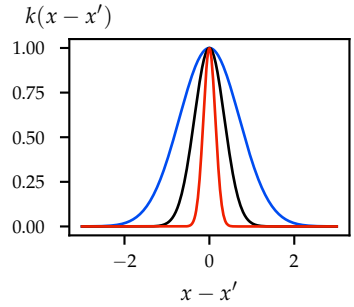


Figure 4.5: Gaussian kernel with length scales $\ell = 1$, $\ell = 0.5$, and $\ell = 0.2$.

3. The *Laplace kernel* (also known as *exponential kernel*) is defined as

$$k(\boldsymbol{x}, \boldsymbol{x}'; \ell) \doteq \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_1}{\ell}\right). \tag{4.15}$$

   As can be seen in fig. 4.7, samples from a GP with Laplace kernel are non-smooth as opposed to the samples from a GP with Gaussian kernel.
4. The *Matérn kernel* trades the smoothness of the Gaussian and the Laplace kernels. It is defined as

$$k(\boldsymbol{x}, \boldsymbol{x}'; \nu, \ell) \doteq \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\,\|\boldsymbol{x} - \boldsymbol{x}'\|_2}{\ell}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}\,\|\boldsymbol{x} - \boldsymbol{x}'\|_2}{\ell}\right) \tag{4.16}$$

   where Γ is the Gamma function, K_ν the modified Bessel function of the second kind, and ℓ a length scale parameter. The result-

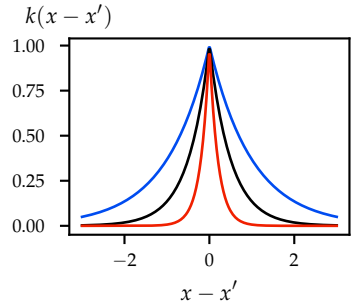


Figure 4.6: Laplace kernel with length scales $\ell = 1$, $\ell = 0.5$, and $\ell = 0.2$.

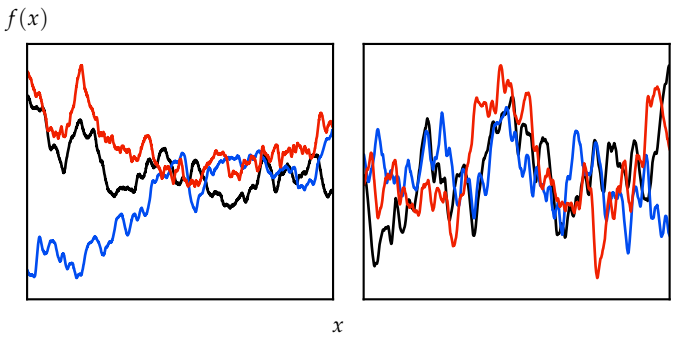


Figure 4.7: Functions sampled according to a Gaussian process with a Laplace kernel and length scales $\ell = 10\,000$ (left) and $\ell = 10$ (right).

ing functions are $\lceil \nu \rceil - 1$ times mean square differentiable.[6] For $\nu = 1/2$, the Matérn kernel is equivalent to the Laplace kernel. For $\nu \to \infty$, the Matérn kernel is equivalent to the Gaussian kernel. In particular, GPs with a Gaussian kernel are infinitely many times mean square differentiable whereas GPs with a Laplace kernel are mean square continuous but not mean square differentiable.

Exercise 4.5: A Kalman filter as a Gaussian process

Next we will show that the Kalman filter from example 3.5 can be seen as a Gaussian process. To this end, we define

$$f : \mathbb{N}_0 \to \mathbb{R}, \quad t \mapsto X_t. \tag{4.21}$$

Assuming that $X_0 \sim \mathcal{N}(0, \sigma_0^2)$ and $X_{t+1} \doteq X_t + \epsilon_t$ with independent noise $\epsilon_t \sim \mathcal{N}(0, \sigma_x^2)$, show that

$$f \sim \mathcal{GP}(0, k_{\text{KF}}) \quad \text{where} \tag{4.22}$$
$$k_{\text{KF}}(t, t') \doteq \sigma_0^2 + \sigma_x^2 \min\{t, t'\}. \tag{4.23}$$

This particular kernel $k(t, t') \doteq \min\{t, t'\}$ but over the continuous-time domain defines the *Wiener process* (also known as Brownian motion).

▷ *Solution*

4.3.2 Kernel Composition

Given kernels $k_1 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $k_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, they can be composed to obtain a new kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ in the following ways:

- $k(\boldsymbol{x}, \boldsymbol{x}') \doteq k_1(\boldsymbol{x}, \boldsymbol{x}') + k_2(\boldsymbol{x}, \boldsymbol{x}')$,
- $k(\boldsymbol{x}, \boldsymbol{x}') \doteq k_1(\boldsymbol{x}, \boldsymbol{x}') \cdot k_2(\boldsymbol{x}, \boldsymbol{x}')$,
- $k(\boldsymbol{x}, \boldsymbol{x}') \doteq c \cdot k_1(\boldsymbol{x}, \boldsymbol{x}')$ for any $c > 0$,
- $k(\boldsymbol{x}, \boldsymbol{x}') \doteq f(k_1(\boldsymbol{x}, \boldsymbol{x}'))$ for any polynomial f with positive coeffi-

[6] Mean square continuity and mean square differentiability are a generalization of continuity and differentiation to random processes, where the limit is generalized to a limit in the "mean square sense".

A sequence of random variables $\{X_n\}_{n \in \mathbb{N}}$ is said to converge to the random variable X in *mean square* if

$$\lim_{n \to \infty} \mathbb{E}\left[(X_n - X)^2\right] = 0. \tag{4.17}$$

Using Markov's inequality (1.82) it can be seen that convergence in mean square implies convergence in probability. Moreover, convergence in mean square implies

$$\lim_{n \to \infty} \mathbb{E} X_n = \mathbb{E} X \quad \text{and} \tag{4.18}$$
$$\lim_{n \to \infty} \mathbb{E} X_n^2 = \mathbb{E} X^2. \tag{4.19}$$

Whereas a deterministic function $f(\boldsymbol{x})$ is said to be continuous at a point $\boldsymbol{x}^\star$ if $\lim_{x \to x^\star} f(\boldsymbol{x}) = f(\boldsymbol{x}^\star)$, a random process $f(\boldsymbol{x})$ is *mean square continuous* at $\boldsymbol{x}^\star$ if

$$\lim_{x \to x^\star} \mathbb{E}\left[(f(\boldsymbol{x}) - f(\boldsymbol{x}^\star))^2\right] = 0. \tag{4.20}$$

It can be shown that a random process is mean square continuous at $\boldsymbol{x}^\star$ iff its kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ is continuous at $\boldsymbol{x} = \boldsymbol{x}' = \boldsymbol{x}^\star$.

Similarly, a random process $f(\boldsymbol{x})$ is *mean square differentiable* at a point $\boldsymbol{x}$ in direction i if $(f(\boldsymbol{x} + \delta \boldsymbol{e}_i) - f(\boldsymbol{x}))/\delta$ converges in mean square as $\delta \to 0$ where $\boldsymbol{e}_i$ is the unit vector in direction i. This notion can be extended to higher-order derivatives.

For the precise notions of mean square continuity and mean square differentiability in the context of Gaussian processes, refer to section 4.1.1 of "Gaussian processes for machine learning" (Williams and Rasmussen, 2006).

cients or $f = \exp$.

For example, the additive structure of a function $f(\boldsymbol{x}) \doteq f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})$ can be easily encoded in GP models. Suppose that $f_1 \sim \mathcal{GP}(\mu_1, k_1)$ and $f_2 \sim \mathcal{GP}(\mu_2, k_2)$, then the distribution of the sum of those two functions $f = f_1 + f_2 \sim \mathcal{GP}(\mu_1 + \mu_2, k_1 + k_2)$ is another GP.[7]

[7] We use $f \doteq f_1 + f_2$ to denote the function $f(\cdot) = f_1(\cdot) + f_2(\cdot)$.

Whereas the addition of two kernels k_1 and k_2 can be thought of as an *OR* operation (i.e., the kernel has high value if either k_1 or k_2 have high value), the multiplication of k_1 and k_2 can be thought of as an *AND* operation (i.e., the kernel has high value if both k_1 and k_2 have high value). For example, the product of two linear kernels results in functions which are quadratic.

As mentioned previously, the constant c of a scaled kernel function $k'(\boldsymbol{x}, \boldsymbol{x}') \doteq c \cdot k(\boldsymbol{x}, \boldsymbol{x}')$ is generally called the *output scale* of a kernel, and it scales the variance $\mathrm{Var}[f(\boldsymbol{x})] = c \cdot k(\boldsymbol{x}, \boldsymbol{x})$ of the predictions $f(\boldsymbol{x})$ from $\mathcal{GP}(\mu, k')$.

Readings

For a broader introduction to how kernels can be used and combined to model certain classes of functions, read

- chapter 2 of "Automatic model construction with Gaussian processes" (Duvenaud, 2014) also known as the "kernel cookbook" and
- chapter 4 of "Gaussian processes for machine learning" (Williams and Rasmussen, 2006).

4.3.3 *Stationarity and Isotropy*

Kernel functions are commonly classified according to two properties.

Definition 4.6 (Stationarity and isotropy)**.** A kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is called

- *stationary* (or *shift-invariant*) if there exists a function $\tilde{k}$ such that $\tilde{k}(\boldsymbol{x} - \boldsymbol{x}') = k(\boldsymbol{x}, \boldsymbol{x}')$, and
- *isotropic* if there exists a function $\tilde{k}$ such that $\tilde{k}(\|\boldsymbol{x} - \boldsymbol{x}'\|_2) = k(\boldsymbol{x}, \boldsymbol{x}')$.

Note that stationarity is a necessary condition for isotropy. In other words, isotropy implies stationarity.

Example 4.7: Stationarity and isotropy of kernels

	stationary	isotropic
linear kernel	no	no
Gaussian kernel	yes	yes
$k(\boldsymbol{x}, \boldsymbol{x}') \doteq \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|_{\boldsymbol{M}}^2)$ where $\boldsymbol{M}$ is positive semi-definite	yes	no

$\|\cdot\|_{\boldsymbol{M}}$ denotes the Mahalanobis norm induced by matrix $\boldsymbol{M}$

For $\boldsymbol{x}' = \boldsymbol{x}$, stationarity implies that the kernel must only depend on $\mathbf{0}$. In other words, a stationary kernel must depend on relative locations only. This is clearly not the case for the linear kernel, which depends on the absolute locations of $\boldsymbol{x}$ and $\boldsymbol{x}'$. Therefore, the linear kernel cannot be isotropic either.

For the Gaussian kernel, isotropy follows immediately from its definition.

The last kernel is clearly stationary by definition, but not isotropic for general matrices $\boldsymbol{M}$. Note that for $\boldsymbol{M} = \boldsymbol{I}$ it is indeed isotropic.

4.3.4 *Reproducing Kernel Hilbert Spaces*

Recall that Gaussian processes keep track of a posterior distribution $f \mid \boldsymbol{x}_{1:n}, y_{1:n}$ over functions. It can be shown that the corresponding MAP estimate $\hat{f}$ corresponds to a regularized optimization problem in a suitable space of functions known as the *reproducing kernel Hilbert space*. This duality is similar to the duality between the MAP estimate of Bayesian linear regression and ridge regression we observed in chapter 2. So what is the reproducing kernel Hilbert space of a kernel function k?

Definition 4.8 (Reproducing kernel Hilbert space, RKHS)**.** Given a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, its corresponding *reproducing kernel Hilbert space* is the space of functions f defined as

$$\mathcal{H}_k(\mathcal{X}) \doteq \left\{ f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) : n \in \mathbb{N}, \boldsymbol{x}_i \in \mathcal{X}, \alpha_i \in \mathbb{R} \right\} \tag{4.24}$$

with inner product

$$\langle f, g \rangle_k \doteq \sum_{i=1}^{n} \sum_{j=1}^{n'} \alpha_i \alpha'_j k(\boldsymbol{x}_i, \boldsymbol{x}'_j), \tag{4.25}$$

where $g(\boldsymbol{x}) = \sum_{j=1}^{n'} \alpha'_j k(\boldsymbol{x}, \boldsymbol{x}'_j)$ and norm $\|f\|_k = \sqrt{\langle f, f \rangle_k}$.

It is straightforward to check that for any $\boldsymbol{x} \in \mathcal{X}$, $k(\boldsymbol{x}, \cdot) \in \mathcal{H}_k(\mathcal{X})$. Moreover, the RKHS inner product $\langle \cdot, \cdot \rangle_k$ satisfies for all $\boldsymbol{x} \in \mathcal{X}$ and

$f \in \mathcal{H}_k(\mathcal{X})$ that $f(\boldsymbol{x}) = \langle f(\cdot), k(\boldsymbol{x}, \cdot)\rangle_k$ which is also known as the *reproducing property*. That is, evaluations of RKHS functions f are inner products in $\mathcal{H}_k(\mathcal{X})$ parameterized by the "feature map" $k(\boldsymbol{x}, \cdot)$.

Exercise 4.9: Reproducing property and RKHS norm

1. Derive the reproducing property.
   Hint: Use $k(\boldsymbol{x}, \boldsymbol{x}') = \langle k(\boldsymbol{x}, \cdot), k(\boldsymbol{x}', \cdot)\rangle_k$.
2. Show that the RKHS norm $\|\cdot\|_k$ is a measure of smoothness by proving that for any $f \in \mathcal{H}_k(\mathcal{X})$ and $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ it holds that $|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq \|f\|_k \, \|k(\boldsymbol{x}, \cdot) - k(\boldsymbol{y}, \cdot)\|_k$.

▷ *Solution*

Foundational in characterizing the solution to regularized optimization problems within RKHSs is the *representer theorem* (Schölkopf et al., 2001).

Fact 4.10 (Representer theorem)**.** *Let k be a kernel and let $\lambda > 0$. For $f \in \mathcal{H}_k(\mathcal{X})$ and training data $\mathcal{D}$ comprising n observations, let $\ell(f, \mathcal{D}) \in \mathbb{R} \cup \{\infty\}$ denote any loss function. Then, any minimizer of*

$$\hat{f} \doteq \operatorname*{arg\,min}_{f \in \mathcal{H}_k(\mathcal{X})} \ell(f, \mathcal{D}) + \lambda \, \|f\|_k^2 \tag{4.26}$$

admits a representation of the form

$$\hat{f}(\boldsymbol{x}) = \hat{\boldsymbol{\alpha}}^\top \boldsymbol{k}_{\boldsymbol{x}, A} \quad \text{for some } \hat{\boldsymbol{\alpha}} \in \mathbb{R}^n. \tag{4.27}$$

Exercise 4.11: MAP estimate of Gaussian processes

We will now show that the MAP estimate of GP regression corresponds to the solution of the following regularized linear regression problem in the RKHS,

$$\hat{f} \doteq \operatorname*{arg\,min}_{f \in \mathcal{H}_k(\mathcal{X})} - \log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, f) + \frac{1}{2} \|f\|_k^2 . \tag{4.28}$$

Here, the first term corresponds to the likelihood, measuring the "quality of fit". The regularization term encodes smoothness assumptions on f as specified by the reproducing kernel Hilbert space of the used kernel function.

In the following, we abbreviate $\boldsymbol{K} = \boldsymbol{K}_{AA}$. We will also assume that the GP has a zero mean function.

1. Show that eq. (4.28) is equivalent to

$$\hat{\boldsymbol{\alpha}} \doteq \operatorname*{arg\,min}_{\boldsymbol{\alpha} \in \mathbb{R}^n} \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha}\|_2^2 + \lambda \, \|\boldsymbol{\alpha}\|_{\boldsymbol{K}}^2 \tag{4.29}$$

for some $\lambda > 0$ which is also known as *kernel ridge regression*. Determine λ.

2. Show that eq. (4.29) with the λ determined in (1) is equivalent to the MAP estimate of GP regression.
   Hint: Recall from eq. (4.6) *that the MAP estimate at a point* $\boldsymbol{x}^\star$ *is* $\mathbb{E}[f^\star \mid \boldsymbol{x}^\star, \boldsymbol{X}, \boldsymbol{y}] = \boldsymbol{k}_{\boldsymbol{x}^\star, A}^\top (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{y}$.

▷ *Solution*

4.4 Model Selection

We have not yet discussed how to pick the hyperparameters $\boldsymbol{\theta}$ (e.g., parameters of kernels). A common technique in supervised learning is to select hyperparameters $\boldsymbol{\theta}$, such that the resulting function estimate $\hat{f}_{\boldsymbol{\theta}}$ leads to the most accurate predictions on hold-out validation data. After reviewing this approach, we contrast it with a Bayesian approach to model selection, which avoids using point estimates of $\hat{f}_{\boldsymbol{\theta}}$ and rather utilizes the full posterior.

4.4.1 Optimizing Validation Set Performance

A common approach to model selection is to split our data $\mathcal{D}$ into separate training set $\mathcal{D}^{\text{train}} \doteq \{(\boldsymbol{x}_i^{\text{train}}, y_i^{\text{train}})\}_{i=1}^n$ and validation sets $\mathcal{D}^{\text{val}} \doteq \{(\boldsymbol{x}_i^{\text{val}}, y_i^{\text{val}})\}_{i=1}^m$. We then optimize the model for a parameter candidate $\boldsymbol{\theta}_j$ using the training set. This is usually done by picking a point estimate (like the MAP estimate),

$$\hat{f}_j \doteq \arg\max_f p(f \mid \boldsymbol{x}_{1:n}^{\text{train}}, y_{1:n}^{\text{train}}). \tag{4.30}$$

Then, we score $\boldsymbol{\theta}_j$ according to the performance of $\hat{f}_j$ on the validation set,

$$\hat{\boldsymbol{\theta}} \doteq \arg\max_{\boldsymbol{\theta}_j} p(y_{1:m}^{\text{val}} \mid \boldsymbol{x}_{1:m}^{\text{val}}, \hat{f}_j). \tag{4.31}$$

This ensures that $\hat{f}_j$ does not depend on $\mathcal{D}^{\text{val}}$.

Remark 4.12: Approximating population risk

Why is it useful to separate the data into a training and a validation set? Recall from section 1.2.1 that minimizing the empirical risk without separating training and validation data may lead to overfitting as both the loss and $\hat{f}_j$ depend on the same data $\mathcal{D}$. In contrast, using independent training and validation sets, $\hat{f}_j$ does

not depend on $\mathcal{D}^{\text{val}}$, and we have that

$$\frac{1}{m}\sum_{i=1}^{m} \ell(y_i^{\text{val}} \mid \boldsymbol{x}_i^{\text{val}}, \hat{f}_j) \approx \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{P}}\Big[\ell(y \mid \boldsymbol{x}, \hat{f}_j)\Big], \tag{4.32}$$

using Monte Carlo sampling.[8] In words, for reasonably large m, minimizing the empirical risk as we do in eq. (4.31) approximates minimizing the population risk.

[8] We generally assume $\mathcal{D} \overset{\text{iid}}{\sim} \mathcal{P}$, in particular, we assume that the individual samples of the data are i.i.d.. Recall that in this setting, Hoeffding's inequality (1.88) can be used to gauge how large m should be.

While this approach often is quite effective at preventing overfitting as compared to using the same data for training and picking $\hat{\boldsymbol{\theta}}$, it still collapses the uncertainty in f into a point estimate. Can we do better?

4.4.2 *Maximizing the Marginal Likelihood*

We have already seen for Bayesian linear regression, that picking a point estimate loses a lot of information. Instead of optimizing the effects of $\boldsymbol{\theta}$ for a specific point estimate $\hat{f}$ of the model f, *maximizing the marginal likelihood* optimizes the effects of $\boldsymbol{\theta}$ across all realizations of f. In this approach, we obtain our hyperparameter estimate via

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{\text{MLE}} &\doteq \arg\max_{\boldsymbol{\theta}} p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta}) && \text{(4.33)} \\
&= \arg\max_{\boldsymbol{\theta}} \int p(y_{1:n}, f \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta})\, df && \text{by conditioning on } f \text{ using the sum rule (1.10)} \\
&= \arg\max_{\boldsymbol{\theta}} \int p(y_{1:n} \mid \boldsymbol{x}_{1:n}, f, \boldsymbol{\theta}) p(f \mid \boldsymbol{\theta})\, df. && \text{(4.34)}
\end{aligned}$$

using the definition of marginal likelihood in Bayes' rule (1.59)

using the product rule (1.14)

Remarkably, this approach typically avoids overfitting even though we do not use a separate training and validation set. The following table provides an intuitive argument for why maximizing the marginal likelihood is a good strategy.

	likelihood	prior
"underfit" model (too simple $\boldsymbol{\theta}$)	small for "almost all" f	large
"overfit" model (too complex $\boldsymbol{\theta}$)	large for "few" f small for "most" f	small
"just right"	moderate for "many" f	moderate

Table 4.1: The table gives an intuitive explanation of effects of parameter choices $\boldsymbol{\theta}$ on the marginal likelihood. Note that words in quotation marks refer to intuitive quantities, as we have infinitely many realizations of f.

For an "underfit" model, the likelihood is mostly small as the data cannot be well described, while the prior is large as there are "fewer" functions to choose from. For an "overfit" model, the likelihood is large for "some" functions (which would be picked if we were only minimizing the training error and not doing cross validation) but small

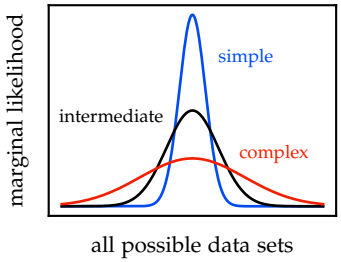


Figure 4.8: A schematic illustration of the marginal likelihood of a simple, intermediate, and complex model across all possible data sets.

for "most" functions. The prior is small, as the probability mass has to be distributed among "more" functions. Thus, in both cases, one term in the product will be small. Hence, maximizing the marginal likelihood naturally encourages trading between a large likelihood and a large prior.

In the context of Gaussian process regression, recall from eq. (4.3) that

$$y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{K}_{f,\boldsymbol{\theta}} + \sigma_n^2 \boldsymbol{I}) \tag{4.35}$$

where $\boldsymbol{K}_{f,\boldsymbol{\theta}}$ denotes the kernel matrix at the inputs $\boldsymbol{x}_{1:n}$ depending on the kernel function parameterized by $\boldsymbol{\theta}$. We write $\boldsymbol{K}_{y,\boldsymbol{\theta}} \doteq \boldsymbol{K}_{f,\boldsymbol{\theta}} + \sigma_n^2 \boldsymbol{I}$. Continuing from eq. (4.33), we obtain

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{\text{MLE}} &= \arg\max_{\boldsymbol{\theta}} \mathcal{N}(\boldsymbol{y}; \mathbf{0}, \boldsymbol{K}_{y,\boldsymbol{\theta}}) \\
&= \arg\min_{\boldsymbol{\theta}} \frac{1}{2} \boldsymbol{y}^\top \boldsymbol{K}_{y,\boldsymbol{\theta}}^{-1} \boldsymbol{y} + \frac{1}{2} \log\det(\boldsymbol{K}_{y,\boldsymbol{\theta}}) + \frac{n}{2} \log 2\pi && (4.36) && \text{taking the negative logarithm} \\
&= \arg\min_{\boldsymbol{\theta}} \frac{1}{2} \boldsymbol{y}^\top \boldsymbol{K}_{y,\boldsymbol{\theta}}^{-1} \boldsymbol{y} + \frac{1}{2} \log\det(\boldsymbol{K}_{y,\boldsymbol{\theta}}) && (4.37) && \text{the last term is independent of } \boldsymbol{\theta}
\end{aligned}$$

The first term of the optimization objective describes the "goodness of fit" (i.e., the "alignment" of $\boldsymbol{y}$ with $\boldsymbol{K}_{y,\boldsymbol{\theta}}$). The second term characterizes the "volume" of the model class. Thus, this optimization naturally trades the aforementioned objectives.

Marginal likelihood maximization is an empirical Bayes method. Often it is simply referred to as *empirical Bayes*. It also has the nice property that the gradient of its objective (the MLL loss) can be expressed in closed-from,

$$\frac{\partial}{\partial \theta_j} \log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta}) = \frac{1}{2} \operatorname{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \boldsymbol{K}_{y,\boldsymbol{\theta}}^{-1}) \frac{\partial \boldsymbol{K}_{y,\boldsymbol{\theta}}}{\partial \theta_j} \right) \tag{4.38}$$

where $\boldsymbol{\alpha} \doteq \boldsymbol{K}_{y,\boldsymbol{\theta}}^{-1} \boldsymbol{y}$ and $\operatorname{tr}(\boldsymbol{M})$ is the trace of a matrix $\boldsymbol{M}$. This optimization problem is, in general, non-convex. Figure 4.9 gives an example of two local optima according to empirical Bayes.

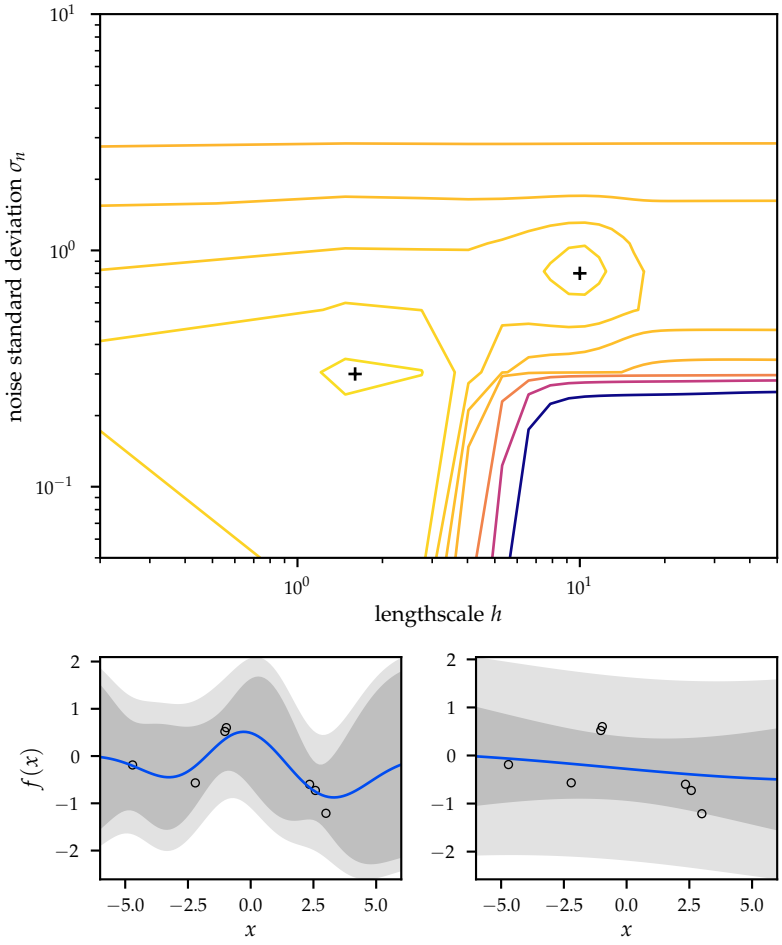


Figure 4.9: The top plot shows contour lines of an empirical Bayes with two local optima. The bottom two plots show the Gaussian processes corresponding to the two optimal models. The left model with smaller lengthscale is chosen within a more flexible class of models, while the right model explains more observations through noise. Adapted from figure 5.5 of "Gaussian processes for machine learning" (Williams and Rasmussen, 2006).

Exercise 4.13: Gradient of the marginal likelihood

In this exercise, we derive eq. (4.38).

Recall that we were considering a dataset $(\boldsymbol{X}, \boldsymbol{y})$ of noise-perturbed evaluations $y_i = f(\boldsymbol{x}_i) + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$ and f is an unknown function. We make the hypothesis $f \sim \mathcal{GP}(0, k_{\boldsymbol{\theta}})$ with a zero mean function and the covariance function $k_{\boldsymbol{\theta}}$. We are interested in finding the hyperparameters $\boldsymbol{\theta}$ that maximize the marginal likelihood $p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$.

1. Derive eq. (4.38).
   Hint: You can use the following identities:
   (a) for any invertible matrix $\boldsymbol{M}$,

$$\frac{\partial}{\partial \theta_j} \boldsymbol{M}^{-1} = -\boldsymbol{M}^{-1} \frac{\partial \boldsymbol{M}}{\partial \theta_j} \boldsymbol{M}^{-1} \quad \textit{and} \tag{4.39}$$

   (b) for any symmetric positive definite matrix $\boldsymbol{M}$,

$$\frac{\partial}{\partial \theta_j} \log \det(\boldsymbol{M}) = \operatorname{tr}\left(\boldsymbol{M}^{-1} \frac{\partial \boldsymbol{M}}{\partial \theta_j} \right). \tag{4.40}$$

2. Assume now that the covariance function for the noisy targets (i.e., including the noise contribution) can be expressed as $k_{\boldsymbol{y},\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}') = \theta_0 \tilde{k}(\boldsymbol{x}, \boldsymbol{x}')$ where $\tilde{k}$ is a valid kernel independent of θ_0.[9] Show that $\frac{\partial}{\partial \theta_0} \log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) = 0$ admits a closed-form solution for θ_0 which we denote by $\theta_0^\star$.
3. How should the optimal parameter $\theta_0^\star$ be scaled if we scale the labels $\boldsymbol{y}$ by a scalar s?

▷ *Solution*

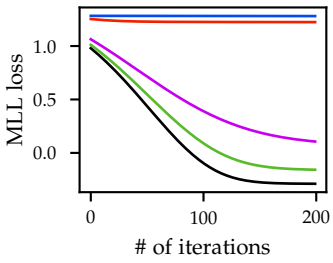


Figure 4.10: An example of model selection by maximizing the log likelihood (without hyperpriors) using a **linear**, **quadratic**, **Laplace**, **Matérn** ($\nu = 3/2$), and **Gaussian** kernel, respectively. They are used to learn the function

$$x \mapsto \frac{\sin(x)}{x} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, 0.01)$$

using SGD with learning rate 0.1.

[9] That is, $\boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}(i, j) = k_{\boldsymbol{y},\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Taking a step back, observe that taking a Bayesian perspective on model selection naturally led us to consider all realizations of our model f instead of using point estimates. However, we are still using point estimates for our model parameters $\boldsymbol{\theta}$. Continuing on our Bayesian adventure, we could place a prior $p(\boldsymbol{\theta})$ on them too.[10] We could use it to obtain the MAP estimate (still a point estimate!) which adds an additional regularization term

[10] Such a prior is called *hyperprior*.

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} \doteq \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}, y_{1:n}) \tag{4.41}$$

$$= \arg\min_{\boldsymbol{\theta}} -\log p(\boldsymbol{\theta}) - \log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta}). \tag{4.42}$$

using Bayes' rule (1.59) and then taking the negative logarithm

An alternative approach is to consider the full posterior distribution over parameters $\boldsymbol{\theta}$. The resulting predictive distribution is, however,

intractable,

$$p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, y_{1:n}) = \int \int p(y^\star \mid \boldsymbol{x}^\star, f) \cdot p(f \mid \boldsymbol{x}_{1:n}, y_{1:n}, \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta})\, df\, d\boldsymbol{\theta}. \tag{4.43}$$

Recall that as the mode of Gaussians coincides with their mean, the MAP estimate corresponds to the mean of the predictive posterior.

As a final note, observe that in principle, there is nothing stopping us from descending deeper in the Bayesian hierarchy. The prior on the model parameters $\boldsymbol{\theta}$ is likely to have parameters too. Ultimately, we need to break out of this hierarchy of dependencies and choose a prior.

Readings

Chapter 5 of "Gaussian processes for machine learning" (Williams and Rasmussen, 2006).

4.5 *Approximations*

To learn a Gaussian process, we need to invert matrices, hence the computational cost is $\mathcal{O}(n^3)$. Compare this to Bayesian linear regression which allows us to learn a regression model in $\mathcal{O}(nd^2)$ time (even online) where d is the feature dimension. It is therefore natural to look for ways of approximating a Gaussian process.

4.5.1 *Local Methods*

Recall that during forward sampling, we had to condition on a larger and larger number of previous samples. When sampling at a location $\boldsymbol{x}$, a very simple approximation is to only condition on those samples $\boldsymbol{x}'$ that are "close" (where $|k(\boldsymbol{x}, \boldsymbol{x}')| \geq \tau$ for some $\tau > 0$). Essentially, this method "cuts off the tails" of the kernel function k. However, τ has to be chosen carefully as if τ is chosen too large, samples become essentially independent.

This is one example of a *sparse approximation* of a Gaussian process. We will discuss more advanced sparse approximations known as "inducing point methods" in section 4.5.3.

4.5.2 *Kernel Function Approximation*

Another method is to approximate the kernel function directly. The idea is to construct a "low-dimensional" feature map $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^m$ that approximates the kernel,

$$k(\boldsymbol{x}, \boldsymbol{x}') \approx \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}'). \tag{4.44}$$

Then, we can apply Bayesian linear regression, resulting in a time complexity of $\mathcal{O}(nm^2 + m^3)$.

One example of this approach are *random Fourier features*, which we will discuss in the following.

Remark 4.14: Fourier transform

First, let us remind ourselves of Fourier transformations. The Fourier transform is a method of decomposing frequencies into their individual components.

Recall *Euler's formula* which states that for any $x \in \mathbb{R}$,

$$e^{ix} = \cos x + i \sin x \tag{4.45}$$

where i is the imaginary unit of complex numbers. The formula is illustrated in fig. 4.11. Note that $e^{-i2\pi x}$ corresponds to rotating clockwise around the unit circle in $\mathbb{R}^2$ — completing a rotation whenever $x \in \mathbb{R}$ reaches the next natural number.

We can scale x by a frequency ξ: $e^{-i2\pi\xi x}$. If $\boldsymbol{x} \in \mathbb{R}^d$, we can also scale each component j of $\boldsymbol{x}$ by a different frequency $\boldsymbol{\xi}(j)$. Multiplying a function $f : \mathbb{R}^d \to \mathbb{R}$ with the rotation around the unit circle with given frequencies $\boldsymbol{\xi}$, yields a quantity that describes the amplitude of the frequencies $\boldsymbol{\xi}$,

$$\hat{f}(\boldsymbol{\xi}) \doteq \int_{\mathbb{R}^d} f(\boldsymbol{x}) e^{-i2\pi\boldsymbol{\xi}^\top \boldsymbol{x}} \, d\boldsymbol{x}. \tag{4.46}$$

$\hat{f}$ is called the *Fourier transform* of f. f is called the *inverse Fourier transform* of $\hat{f}$, and can be computed using

$$f(\boldsymbol{x}) = \int_{\mathbb{R}^d} \hat{f}(\boldsymbol{\xi}) e^{i2\pi\boldsymbol{\xi}^\top \boldsymbol{x}} \, d\boldsymbol{\xi}. \tag{4.47}$$

It is common to write $\boldsymbol{\omega} \doteq 2\pi\boldsymbol{\xi}$. See fig. 4.12 for an example.

Refer to "But what is the Fourier Transform? A visual introduction" (Sanderson, 2018) for a visual introduction.

Because a stationary kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ can be interpreted as a function in one variable, it has an associated Fourier transform which we denote by $p(\boldsymbol{\omega})$. That is,

$$k(\boldsymbol{x} - \boldsymbol{x}') = \int_{\mathbb{R}^d} p(\boldsymbol{\omega}) e^{i\boldsymbol{\omega}^\top (\boldsymbol{x} - \boldsymbol{x}')} \, d\boldsymbol{\omega}. \tag{4.48}$$

Fact 4.15 (Bochner's theorem). *A continuous stationary kernel on $\mathbb{R}^d$ is positive definite if and only if its Fourier transform $p(\boldsymbol{\omega})$ is non-negative.*

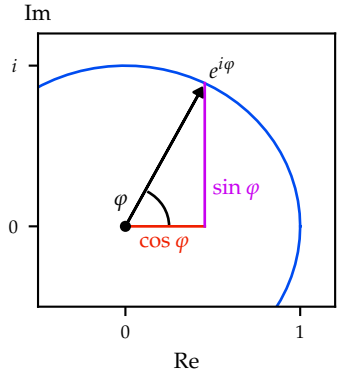


Figure 4.11: Illustration of Euler's formula. It can be seen that $e^{i\varphi}$ corresponds to a (counter-clockwise) rotation on the unit circle as φ varies from 0 to 2π.

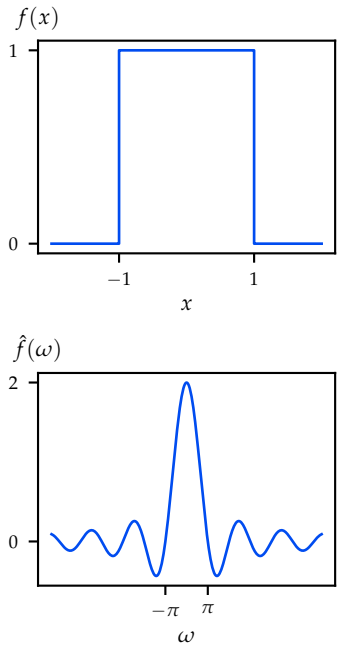


Figure 4.12: The Fourier transform of a rectangular pulse,

$$f(x) \doteq \begin{cases} 1 & x \in [-1, 1] \\ 0 & \text{otherwise,} \end{cases}$$

is given by

$$\hat{f}(\omega) = \int_{-1}^{1} e^{-i\omega x} \, dx = \frac{1}{i\omega}\left(e^{i\omega} - e^{-i\omega}\right) = \frac{2\sin(\omega)}{\omega}.$$

Bochner's theorem implies that when a continuous and stationary kernel is positive definite and scaled appropriately, its Fourier transform $p(\boldsymbol{\omega})$ is a proper probability distribution. In this case, $p(\boldsymbol{\omega})$ is called the *spectral density* of the kernel k.

Remark 4.16: Eigenvalue spectrum of stationary kernels

It turns out that $e^{i\boldsymbol{\omega}^\top(\boldsymbol{x}-\boldsymbol{x}')}$ are the eigenfunctions of a stationary kernel with respect to Lebesgue measure. Intuitively, $p(\boldsymbol{\omega})$ corresponds therefore to the average magnitude of the eigenfunction with frequency $\boldsymbol{\omega}$. The decay of eigenvalues provides information about the smoothness of a kernel: a "rougher" process has more power at high frequencies, and therefore their eigenvalues decay more slowly.

For an in-depth introduction to the eigenfunction analysis of kernels, refer to section 4.3 of "Gaussian processes for machine learning" (Williams and Rasmussen, 2006).

Example 4.17: Spectral density of the Gaussian kernel

The Gaussian kernel with length scale ℓ has the spectral density

$$\begin{aligned} p(\boldsymbol{\omega}) &= \int_{\mathbb{R}^d} k(\boldsymbol{x}-\boldsymbol{x}';\ell) e^{-i\boldsymbol{\omega}^\top(\boldsymbol{x}-\boldsymbol{x}')}\, d(\boldsymbol{x}-\boldsymbol{x}') && \text{using the definition of the Fourier transform (4.46)} \\ &= \int_{\mathbb{R}^d} \exp\left(-\frac{\|\boldsymbol{x}\|_2^2}{2\ell^2} - i\boldsymbol{\omega}^\top\boldsymbol{x}\right) d\boldsymbol{x} && \text{using the definition of the Gaussian kernel (4.14)} \\ &= (2\ell^2\pi)^{d/2} \exp\left(-\ell^2\frac{\|\boldsymbol{\omega}\|_2^2}{2}\right). && (4.49) \end{aligned}$$

The key idea is now to interpret the kernel as an expectation,

$$\begin{aligned} k(\boldsymbol{x}-\boldsymbol{x}') &= \int_{\mathbb{R}^d} p(\boldsymbol{\omega}) e^{i\boldsymbol{\omega}^\top(\boldsymbol{x}-\boldsymbol{x}')}\, d\boldsymbol{\omega} && \text{from eq. (4.48)} \\ &= \mathbb{E}_{\boldsymbol{\omega}\sim p}\left[e^{i\boldsymbol{\omega}^\top(\boldsymbol{x}-\boldsymbol{x}')}\right] && \text{by the definition of expectation (1.23b)} \\ &= \mathbb{E}_{\boldsymbol{\omega}\sim p}\left[\cos(\boldsymbol{\omega}^\top\boldsymbol{x}-\boldsymbol{\omega}^\top\boldsymbol{x}') + i\sin(\boldsymbol{\omega}^\top\boldsymbol{x}-\boldsymbol{\omega}^\top\boldsymbol{x}')\right]. && \text{using Euler's formula (4.45)} \end{aligned}$$

Observe that as both k and p are real, convergence of the integral implies $\mathbb{E}_{\boldsymbol{\omega}\sim p}\left[\sin(\boldsymbol{\omega}^\top\boldsymbol{x}-\boldsymbol{\omega}^\top\boldsymbol{x}')\right] = 0$. Hence,

$$\begin{aligned} &= \mathbb{E}_{\boldsymbol{\omega}\sim p}\left[\cos(\boldsymbol{\omega}^\top\boldsymbol{x}-\boldsymbol{\omega}^\top\boldsymbol{x}')\right] \\ &= \mathbb{E}_{\boldsymbol{\omega}\sim p}\mathbb{E}_{b\sim\mathrm{Unif}([0,2\pi])}\left[\cos((\boldsymbol{\omega}^\top\boldsymbol{x}+b)-(\boldsymbol{\omega}^\top\boldsymbol{x}'+b))\right] && \text{epanding with } b-b \\ &= \mathbb{E}_{\boldsymbol{\omega}\sim p}\mathbb{E}_{b\sim\mathrm{Unif}([0,2\pi])}\left[\cos(\boldsymbol{\omega}^\top\boldsymbol{x}+b)\cos(\boldsymbol{\omega}^\top\boldsymbol{x}'+b)\right. \\ &\qquad\qquad \left. +\sin(\boldsymbol{\omega}^\top\boldsymbol{x}+b)\sin(\boldsymbol{\omega}^\top\boldsymbol{x}'+b)\right] && \text{using the angle subtraction identity, } \cos(\alpha-\beta)=\cos\alpha\cos\beta+\sin\alpha\sin\beta \end{aligned}$$

$$= \mathbb{E}_{\omega\sim p}\mathbb{E}_{b\sim\text{Unif}([0,2\pi])}\left[2\cos(\omega^\top x + b)\cos(\omega^\top x' + b)\right]$$
$$= \mathbb{E}_{\omega\sim p, b\sim\text{Unif}([0,2\pi])}\left[z_{\omega,b}(x)\cdot z_{\omega,b}(x')\right] \tag{4.50}$$

using

$$\mathbb{E}_b[\cos(\alpha+b)\cos(\beta+b)] = \mathbb{E}_b[\sin(\alpha+b)\sin(\beta+b)]$$

for $b \sim \text{Unif}([0,2\pi])$

where $z_{\omega,b}(x) \doteq \sqrt{2}\cos(\omega^\top x + b)$,

$$\approx \frac{1}{m}\sum_{i=1}^{m} z_{\omega^{(i)},b^{(i)}}(x)\cdot z_{\omega^{(i)},b^{(i)}}(x') \tag{4.51}$$

using Monte Carlo sampling to estimate the expectation, see example 1.36

for independent samples $\omega^{(i)} \overset{\text{iid}}{\sim} p$ and $b^{(i)} \overset{\text{iid}}{\sim} \text{Unif}([0,2\pi])$,

$$= z(x)^\top z(x') \tag{4.52}$$

where

$$z(x) \doteq \frac{1}{\sqrt{m}}\left[z_{\omega^{(1)},b^{(1)}}(x), \ldots, z_{\omega^{(m)},b^{(m)}}(x)\right]^\top \tag{4.53}$$

is the (randomized) feature map of random Fourier features.

Intuitively, each component of the feature map $z(x)$ projects x onto a random direction ω drawn from the (inverse) Fourier transform $p(\omega)$ of $k(x-x')$, and wraps this line onto the unit circle in $\mathbb{R}^2$. After transforming two points x and x' in this way, their inner product is an unbiased estimator of $k(x-x')$. The mapping $z_{\omega,b}(x) = \sqrt{2}\cos(\omega^\top x + b)$ additionally rotates the circle by a random amount b and projects the points onto the interval $[0,1]$.

Rahimi et al. (2007) show that Bayesian linear regression with the feature map z approximates Gaussian processes with a stationary kernel:

Theorem 4.18 (Uniform convergence of Fourier features)**.** *Suppose $\mathcal{M}$ is a compact subset of $\mathbb{R}^d$ with diameter $\text{diam}(\mathcal{M})$. Then for a stationary kernel k, the random Fourier features z, and any $\epsilon > 0$ it holds that*

$$\mathbb{P}\left(\sup_{x,x'\in\mathcal{M}}\left|z(x)^\top z(x') - k(x-x')\right| \geq \epsilon\right) \leq 2^8\left(\frac{\sigma_p \text{diam}(\mathcal{M})}{\epsilon}\right)^2 \exp\left(-\frac{m\epsilon^2}{8(d+2)}\right) \tag{4.54}$$

where $\sigma_p^2 \doteq \mathbb{E}_{\omega\sim p}\left[\omega^\top\omega\right]$ is the second moment of p, m is the dimension of $z(x)$, and d is the dimension of x.

Note that the error probability decays exponentially fast in the dimension of the Fourier feature space.

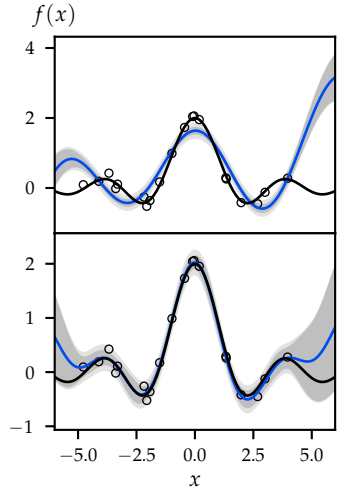


Figure 4.13: Example of random Fourier features with where the number of features m is 5 (top) and 10 (bottom), respectively. The noise-free true function is shown in black and the mean of the Gaussian process is shown in blue.

Exercise 4.19: Uniform convergence of Fourier features

In this exercise, we will prove the above theorem.

Let $s(\boldsymbol{x}, \boldsymbol{x}') \doteq \boldsymbol{z}(\boldsymbol{x})^\top \boldsymbol{z}(\boldsymbol{x}')$ and $f(\boldsymbol{x}, \boldsymbol{x}') \doteq s(\boldsymbol{x}, \boldsymbol{x}') - k(\boldsymbol{x}, \boldsymbol{x}')$. Observe that both functions are shift invariant, and we will therefore denote them as univariate functions with argument $\boldsymbol{\Delta} \equiv \boldsymbol{x} - \boldsymbol{x}' \in \mathcal{M}_\Delta$. Notice that our goal is to bound the probability of the event $\sup_{\boldsymbol{\Delta} \in \mathcal{M}_\Delta} |f(\boldsymbol{\Delta})| \geq \epsilon$.

1. Show that for all $\boldsymbol{\Delta} \in \mathcal{M}_\Delta$, $\mathbb{P}(|f(\boldsymbol{\Delta})| \geq \epsilon) \leq 2 \exp\left(-\frac{m\epsilon^2}{4}\right)$.

What we have derived in (1) is known as a *pointwise convergence* guarantee. However, we are interested in bounding the *uniform convergence* over the compact set $\mathcal{M}_\Delta$.

Our approach will be to "cover" the compact set $\mathcal{M}_\Delta$ using T balls of radius r whose centers we denote by $\{\boldsymbol{\Delta}_i\}_{i=1}^T$. It can be shown that this is possible for some $T \leq (4 \operatorname{diam}(\mathcal{M})/r)^d$. It can furthermore be shown that

$$\forall i.\ |f(\boldsymbol{\Delta}_i)| < \frac{\epsilon}{2} \text{ and } \|\boldsymbol{\nabla} f(\boldsymbol{\Delta}^\star)\|_2 < \frac{\epsilon}{2r} \implies \sup_{\boldsymbol{\Delta} \in \mathcal{M}_\Delta} |f(\boldsymbol{\Delta})| < \epsilon$$

where $\boldsymbol{\Delta}^\star = \arg\max_{\boldsymbol{\Delta} \in \mathcal{M}_\Delta} \|\boldsymbol{\nabla} f(\boldsymbol{\Delta})\|_2$.

2. Prove $\mathbb{P}\left(\|\boldsymbol{\nabla} f(\boldsymbol{\Delta}^\star)\|_2 \geq \frac{\epsilon}{2r}\right) \leq \left(\frac{2r\sigma_p}{\epsilon}\right)^2$.
   Hint: Recall that the random Fourier feature approximation is unbiased, i.e., $\mathbb{E}[s(\boldsymbol{\Delta})] = k(\boldsymbol{\Delta})$.
3. Prove $\mathbb{P}\left(\bigcup_{i=1}^T |f(\boldsymbol{\Delta}_i)| \geq \frac{\epsilon}{2}\right) \leq 2T \exp\left(-\frac{m\epsilon^2}{16}\right)$.
4. Combine the results from (2) and (3) to prove theorem 4.18.
   Hint: You may use that
   (a) $\alpha r^{-d} + \beta r^2 = 2\beta^{\frac{d}{d+2}} \alpha^{\frac{2}{d+2}}$ *for* $r = (\alpha/\beta)^{\frac{1}{d+2}}$ *and*
   (b) $\frac{\sigma_p \operatorname{diam}(\mathcal{M})}{\epsilon} \geq 1$.
5. Show that for the Gaussian kernel (4.14), $\sigma_p^2 = \frac{d}{\ell^2}$.
   Hint: First show $\sigma_p^2 = -\operatorname{tr}(\boldsymbol{H}_{\boldsymbol{\Delta}} k(\boldsymbol{0}))$.

▷ *Solution*

Remark 4.20: Solving GPs using Kalman filters and smoothers

It turns out that for kernel functions whose spectral density can be represented as the rational function

$$p(\boldsymbol{\omega}) = \frac{\text{constant}}{\text{polynomial in } \boldsymbol{\omega}^2}, \tag{4.55}$$

Kalman filters and smoothers can be used for learning and sampling. In the following, we give a brief summary of this method. For more details, read "Kalman filtering and smoothing solutions

to temporal Gaussian process regression models" (Hartikainen and Särkkä, 2010).

If the spectral density is of this form it can be shown that the random process f can be represented as the output of a linear time invariant stochastic differential equation.[11] Such Gaussian processes are also known as *Gaussian Markov Processes*. It can be shown that the Matérn kernel satisfies this property (and hence, GPs with a Matérn kernel can be solved *exactly* using this method); the spectral density of the Gaussian kernel (see eq. (4.49)) can be approximated using a second-order Taylor expansion.

The continuous linear time invariant model can be transformed into a Kalman filter where the motion and sensor models depend on the spectral density of the kernel function (Hartikainen and Särkkä, 2010).

Then, computing the marginal posterior $f \mid \boldsymbol{x}_{1:n}, y_{1:n}$ is simply a Kalman smoothing problem as seen in remark 3.4, and hence, can be solved in linear time in n.

This approach also lends itself to marginal likelihood maximization, as the terms of the optimization objective are computed as by-products of the Kalman filter algorithm.

[11] This can also be thought of as a first-order continuous (vector) Markov process

4.5.3 *Data Sampling*

Another natural approach is to only consider a (random) subset of the training data during learning. The naïve approach is to subsample uniformly at random. Not very surprisingly, we can do much better.

One subsampling method is the so-called *inducing points method* (Quinonero-Candela and Rasmussen, 2005). The idea is to summarize the data around so-called inducing points.[12] For now, let us consider an arbitrary set of inducing points,

$$U \doteq \{\overline{\boldsymbol{x}}_1, \dots, \overline{\boldsymbol{x}}_k\}.$$

Then, the original Gaussian process can be recovered using marginalization,

$$p(f^\star, \boldsymbol{f}) = \int_{\mathbb{R}^k} p(f^\star, \boldsymbol{f}, \boldsymbol{u})\, d\boldsymbol{u} = \int_{\mathbb{R}^k} p(f^\star, \boldsymbol{f} \mid \boldsymbol{u}) p(\boldsymbol{u})\, d\boldsymbol{u}, \tag{4.56}$$

using the sum rule (1.10) and product rule (1.14)

where $\boldsymbol{f} \doteq [f(\boldsymbol{x}_1) \ \cdots \ f(\boldsymbol{x}_n)]^\top$ and $f^\star \doteq f(\boldsymbol{x}^\star)$ at some evaluation point $\boldsymbol{x}^\star \in \mathcal{X}$. We use $\boldsymbol{u} \doteq [f(\overline{\boldsymbol{x}}_1) \ \cdots \ f(\overline{\boldsymbol{x}}_k)]^\top \in \mathbb{R}^k$ to denote the predictions of the model at the inducing points U. Due to the marginalization property of Gaussian processes (4.1), we have that $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_{UU})$.

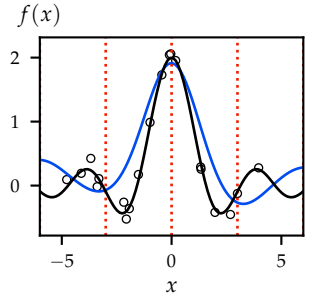


Figure 4.14: Inducing points $\boldsymbol{u}$ are shown as vertical dotted red lines. The noise-free true function is shown in black and the mean of the Gaussian process is shown in blue. Observe that the true function is approximated "well" around the inducing points.

[12] The inducing points can be treated as hyperparameters.

The key idea is to approximate the joint prior, assuming that $f^\star$ and $\boldsymbol{f}$ are conditionally independent given $\boldsymbol{u}$,

$$p(f^\star, \boldsymbol{f}) \approx \int_{\mathbb{R}^k} p(f^\star \mid \boldsymbol{u}) p(\boldsymbol{f} \mid \boldsymbol{u}) p(\boldsymbol{u}) \, d\boldsymbol{u}. \tag{4.57}$$

Here, $p(\boldsymbol{f} \mid \boldsymbol{u})$ and $p(f^\star \mid \boldsymbol{u})$ are commonly called the *training conditional* and the *testing conditional*, respectively. Still denoting the observations by $A = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and defining $\star \doteq \{\boldsymbol{x}^\star\}$, we know, using the closed-form expression for conditional Gaussians (1.118),

$$p(\boldsymbol{f} \mid \boldsymbol{u}) \sim \mathcal{N}(\boldsymbol{f}; \boldsymbol{K}_{AU}\boldsymbol{K}_{UU}^{-1}\boldsymbol{u}, \boldsymbol{K}_{AA} - \boldsymbol{Q}_{AA}), \tag{4.58a}$$
$$p(f^\star \mid \boldsymbol{u}) \sim \mathcal{N}(f^\star; \boldsymbol{K}_{\star U}\boldsymbol{K}_{UU}^{-1}\boldsymbol{u}, \boldsymbol{K}_{\star\star} - \boldsymbol{Q}_{\star\star}) \tag{4.58b}$$

where $\boldsymbol{Q}_{ab} \doteq \boldsymbol{K}_{aU}\boldsymbol{K}_{UU}^{-1}\boldsymbol{K}_{Ub}$. Intuitively, $\boldsymbol{K}_{AA}$ represents the prior covariance and $\boldsymbol{Q}_{AA}$ represents the covariance "explained" by the inducing points.[13]

[13] For more details, refer to section 2 of "A unifying view of sparse approximate Gaussian process regression" (Quinonero-Candela and Rasmussen, 2005).

Computing the full covariance matrix is expensive. In the following, we mention two approximations to the covariance of the training conditional (and testing conditional).

Example 4.21: Subset of regressors

The *subset of regressors* (SoR) approximation is defined as

$$q_{\text{SoR}}(\boldsymbol{f} \mid \boldsymbol{u}) \doteq \mathcal{N}(\boldsymbol{f}; \boldsymbol{K}_{AU}\boldsymbol{K}_{UU}^{-1}\boldsymbol{u}, \boldsymbol{0}), \tag{4.59a}$$
$$q_{\text{SoR}}(f^\star \mid \boldsymbol{u}) \doteq \mathcal{N}(f^\star; \boldsymbol{K}_{\star U}\boldsymbol{K}_{UU}^{-1}\boldsymbol{u}, \boldsymbol{0}). \tag{4.59b}$$

Comparing to eq. (4.58), SoR simply forgets about all variance and covariance.

Exercise 4.22: Subset of regressors

1. Using an SoR approximation, prove the following:

$$q_{\text{SoR}}(\boldsymbol{f}, f^\star) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{f} \\ f^\star\end{bmatrix}; \boldsymbol{0}, \begin{bmatrix}\boldsymbol{Q}_{AA} & \boldsymbol{Q}_{A\star} \\ \boldsymbol{Q}_{\star A} & \boldsymbol{Q}_{\star\star}\end{bmatrix}\right) \tag{4.60}$$
$$q_{\text{SoR}}(f^\star \mid \boldsymbol{y}) = \mathcal{N}(f^\star; \boldsymbol{Q}_{\star A}\tilde{\boldsymbol{Q}}_{AA}^{-1}\boldsymbol{y}, \boldsymbol{Q}_{\star\star} - \boldsymbol{Q}_{\star A}\tilde{\boldsymbol{Q}}_{AA}^{-1}\boldsymbol{Q}_{A\star}) \tag{4.61}$$

   where $\tilde{\boldsymbol{Q}}_{ab} \doteq \boldsymbol{Q}_{ab} + \sigma_n^2$.
2. Derive that the resulting model is a degenerate Gaussian process with covariance function

$$k_{\text{SoR}}(\boldsymbol{x}, \boldsymbol{x}') \doteq \boldsymbol{k}_{\boldsymbol{x},U}^\top \boldsymbol{K}_{UU}^{-1} \boldsymbol{k}_{\boldsymbol{x}',U}. \tag{4.62}$$

▷ *Solution*

Example 4.23: Fully independent training conditional

The *fully independent training conditional* (FITC) approximation is defined as

$$q_{\text{FITC}}(\boldsymbol{f} \mid \boldsymbol{u}) \doteq \mathcal{N}(\boldsymbol{f}; \boldsymbol{K}_{AU}\boldsymbol{K}_{UU}^{-1}\boldsymbol{u}, \operatorname{diag}\{\boldsymbol{K}_{AA} - \boldsymbol{Q}_{AA}\}), \tag{4.63a}$$

$$q_{\text{FITC}}(f^\star \mid \boldsymbol{u}) \doteq \mathcal{N}(f^\star; \boldsymbol{K}_{\star U}\boldsymbol{K}_{UU}^{-1}\boldsymbol{u}, \operatorname{diag}\{\boldsymbol{K}_{\star\star} - \boldsymbol{Q}_{\star\star}\}). \tag{4.63b}$$

In contrast to SoR, FITC keeps track of the variances but forgets about the covariance.

The computational cost for inducing point methods SoR and FITC is dominated by the cost of inverting $\boldsymbol{K}_{UU}$. Thus, the time complexity is cubic in the number of inducing points, but only linear in the number of data points.

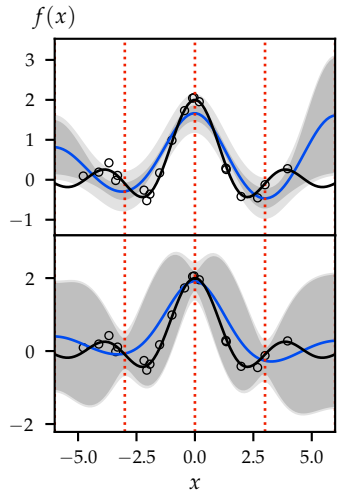


Figure 4.15: Comparison of SoR (top) and FITC (bottom). The inducing points $\boldsymbol{u}$ are shown as vertical dotted red lines. The noise-free true function is shown in black and the mean of the Gaussian process is shown in blue.

5

Variational Inference

We have seen how to do learning and inference with Gaussians, exploiting their closed-form formulas for marginal and conditional distributions. But what if we work with other distributions?

In this and the following chapter, we will discuss two methods of approximate inference. We begin by discussing variational inference, which aims to find a good approximation of the posterior distribution from which it is easy to sample. In chapter 6, we discuss Markov chain Monte Carlo methods, which approximate the sampling from the posterior distribution directly.

The fundamental idea behind variational inference is to approximate the true posterior distribution using a "simpler" posterior that is as close as possible to the true posterior:

$$p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}, y_{1:n}) = \frac{1}{Z} p(\boldsymbol{\theta}, y_{1:n} \mid \boldsymbol{x}_{1:n}) \approx q(\boldsymbol{\theta} \mid \boldsymbol{\lambda}) \doteq q_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) \tag{5.1}$$

where $\boldsymbol{\lambda}$ represents the parameters of the *variational posterior* $q_{\boldsymbol{\lambda}}$, also called *variational parameters*. In doing so, variational inference reduces learning and inference (where the fundamental difficulty lies in solving high-dimensional integrals) to an optimization problem. We have already seen in section 1.5 that optimizing (stochastic) objectives is a well-understood problem with efficient algorithms that perform well in practice.

5.1 Laplace Approximation

Before introducing a general framework of variational inference, we discuss a simpler method of approximate inference known as *Laplace's method*. This method was proposed as a method of approximating integrals as early as 1774 by Pierre-Simon Laplace. The idea is to use

a Gaussian approximation (that is, a second-order Taylor approximation) of the posterior distribution around its mode. Let

$$\psi(\boldsymbol{\theta}) \doteq \log p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}, y_{1:n}) \tag{5.2}$$

denote the log-posterior. Then, using a second-order Taylor approximation (1.104) around the mode $\hat{\boldsymbol{\theta}}$ of ψ (i.e., the MAP estimate), we obtain

$$\begin{aligned}\psi(\boldsymbol{\theta}) \approx \hat{\psi}(\boldsymbol{\theta}) &\doteq \psi(\hat{\boldsymbol{\theta}}) + (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^\top \boldsymbol{\nabla}\psi(\hat{\boldsymbol{\theta}}) + \frac{1}{2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^\top \boldsymbol{H}_\psi(\hat{\boldsymbol{\theta}})(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) \\ &= \psi(\hat{\boldsymbol{\theta}}) + \frac{1}{2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^\top \boldsymbol{H}_\psi(\hat{\boldsymbol{\theta}})(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}). \qquad (5.3) \qquad \text{using } \boldsymbol{\nabla}\psi(\hat{\boldsymbol{\theta}}) = 0\end{aligned}$$

Compare this expression to the log of the PDF of a Gaussian,

$$\log \mathcal{N}(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}, \boldsymbol{\Lambda}^{-1}) = -\frac{1}{2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^\top \boldsymbol{\Lambda}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) + \text{const.} \tag{5.4}$$

Thus,

$$\begin{aligned}\hat{\psi}(\boldsymbol{\theta}) &= \psi(\hat{\boldsymbol{\theta}}) + \frac{1}{2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^\top \boldsymbol{H}_\psi(\hat{\boldsymbol{\theta}})(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) \\ &= \log \mathcal{N}(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}, -\boldsymbol{H}_\psi(\hat{\boldsymbol{\theta}})^{-1}) + \text{const}, \qquad (5.5)\end{aligned}$$

where we note that $\psi(\hat{\boldsymbol{\theta}})$ is a constant. We therefore define

$$\boldsymbol{\Lambda} \doteq -\boldsymbol{H}_\psi(\hat{\boldsymbol{\theta}}) = -\boldsymbol{H}_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}, y_{1:n})\big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}. \tag{5.6}$$

Observe that $\boldsymbol{\Lambda}$ is symmetric. Moreover, ψ is a concave function and hence $\boldsymbol{H}_\psi(\boldsymbol{\theta})$ is negative semi-definite,[1] implying that $\boldsymbol{\Lambda}$ is positive semi-definite. Using that the inverse of a positive semi-definite matrix is also positive semi-definite, $\boldsymbol{\Lambda}^{-1}$ is a valid covariance matrix. The *Laplace approximation* q of p is then given by

$$q(\boldsymbol{\theta}) \doteq \mathcal{N}(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}, \boldsymbol{\Lambda}^{-1}) \propto \exp(\hat{\psi}(\boldsymbol{\theta})). \tag{5.7}$$

Intuitively, the Laplace approximation matches the shape of the true posterior around its mode but may not represent it accurately elsewhere — often leading to extremely overconfident predictions. An example is given in fig. 5.1. Nevertheless, the Laplace approximation has some desirable properties such as being relatively easy to apply in a post-hoc manner, that is, after having already computed the MAP estimate. It preserves the MAP point estimate as its mean and just "adds" a little uncertainty around it. However, the fact that it can be arbitrarily different from the true posterior makes it unsuitable for approximate Bayesian inference.

[1] see remark 1.57

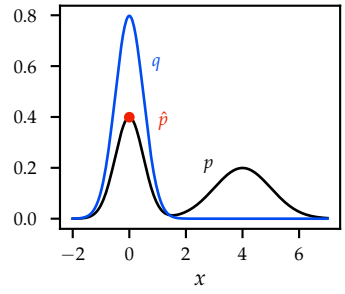


Figure 5.1: The Laplace approximation q greedily selects the mode of the true posterior distribution p and matches the curvature around the mode $\hat{p}$. As shown here, the Laplace approximation can be extremely overconfident when p is not approximately Gaussian.

Example 5.1: Laplace approximation of a Gaussian

Consider approximating the Gaussian density $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ using a Laplace approximation.

We know that the mode of p is $\boldsymbol{\mu}$, which we can verify by computing the gradient,

$$\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) = -\frac{1}{2}(2\boldsymbol{\Sigma}^{-1}\boldsymbol{\theta} - 2\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}) \stackrel{!}{=} 0 \iff \boldsymbol{\theta} = \boldsymbol{\mu}. \tag{5.8}$$

For the Hessian of $\log p(\boldsymbol{\theta})$, we get

$$\boldsymbol{H}_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) = (\boldsymbol{D}_{\boldsymbol{\theta}}(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \boldsymbol{\Sigma}^{-1}\boldsymbol{\theta}))^\top = -(\boldsymbol{\Sigma}^{-1})^\top = -\boldsymbol{\Sigma}^{-1}. \tag{5.9}$$

using $(\boldsymbol{A}^{-1})^\top = (\boldsymbol{A}^\top)^{-1}$ and symmetry of $\boldsymbol{\Sigma}$

Thus, the Laplace approximation of a Gaussian $p(\boldsymbol{\theta})$ is exact (because the second-order Taylor approximation of $\log p(\boldsymbol{\theta})$ is exact).

5.1.1 *Bayesian Logistic Regression*

As an example, we will look at Laplace approximation in the context of Bayesian logistic regression. Logistic regression learns a classifier that decides for a given input whether it belongs to one of two classes. A sigmoid function, typically the *logistic function*,

$$\sigma(z) \doteq \frac{1}{1+\exp(-z)} \in (0,1), \qquad z = \boldsymbol{w}^\top \boldsymbol{x}, \tag{5.10}$$

is used to obtain the class probabilities. *Bayesian logistic regression* corresponds to Bayesian linear regression with a Bernoulli likelihood,

$$y \mid \boldsymbol{x}, \boldsymbol{w} \sim \text{Bern}(\sigma(\boldsymbol{w}^\top \boldsymbol{x})), \tag{5.11}$$

where $y \in \{-1, 1\}$ is the binary class label.[2] Observe that given a data point $(\boldsymbol{x}, y)$, the probability of a correct classification is

$$p(y \mid \boldsymbol{x}, \boldsymbol{w}) = \begin{cases} \sigma(\boldsymbol{w}^\top \boldsymbol{x}) & \text{if } y = 1 \\ 1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}) & \text{if } y = -1 \end{cases} = \sigma(y\boldsymbol{w}^\top \boldsymbol{x}), \tag{5.12}$$

as the logistic function σ is symmetric around 0. Also, recall that Bayesian linear regression used the prior

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}; \boldsymbol{0}, \sigma_p^2 \boldsymbol{I}) \propto \exp\left(-\frac{1}{2\sigma_p^2}\|\boldsymbol{w}\|_2^2\right).$$

Let us first find the posterior mode, that is, the MAP estimate of the weights:

$$\hat{\boldsymbol{w}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w} \mid \boldsymbol{x}_{1:n}, y_{1:n})$$

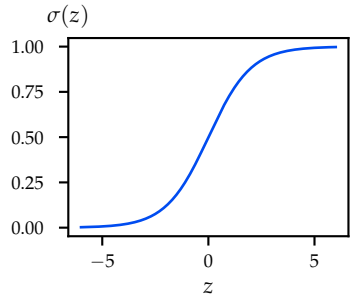


Figure 5.2: The logistic function squashes the linear function $\boldsymbol{w}^\top \boldsymbol{x}$ onto the interval $(0, 1)$.

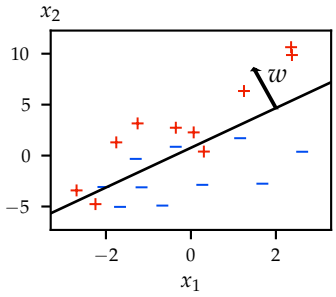


Figure 5.3: Logistic regression classifies data into two classes with a linear decision boundary.

[2] The same approach extends to Gaussian processes where it is known as *Gaussian process classification*. See exercise 5.5 and "Scalable variational Gaussian process classification" (Hensman et al., 2015).

$$= \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}) p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{w}) \qquad \text{using Bayes' rule (1.59)}$$

$$= \arg\max_{\boldsymbol{w}} \log p(\boldsymbol{w}) + \log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{w}) \qquad \text{taking the logarithm}$$

$$= \arg\max_{\boldsymbol{w}} -\frac{1}{2\sigma_p^2} \|\boldsymbol{w}\|_2^2 + \sum_{i=1}^{n} \log \sigma(y_i \boldsymbol{w}^\top \boldsymbol{x}_i) \qquad \text{using independence of the observations and eq. (5.12)}$$

$$= \arg\min_{\boldsymbol{w}} \frac{1}{2\sigma_p^2} \|\boldsymbol{w}\|_2^2 + \sum_{i=1}^{n} \log(1 + \exp(-y_i \boldsymbol{w}^\top \boldsymbol{x}_i)). \qquad (5.13) \qquad \text{using the definition of } \sigma \text{ (5.10)}$$

Note that for $\lambda = 1/2\sigma_p^2$, the above optimization is equivalent to standard (regularized) logistic regression where

$$\ell_{\log}(\boldsymbol{w}^\top \boldsymbol{x}; y) \doteq \log(1 + \exp(-y \boldsymbol{w}^\top \boldsymbol{x})) \qquad (5.14)$$

is called *logistic loss*. You show in exercise 5.3 that the gradient of the logistic loss is given by

$$\nabla_{\boldsymbol{w}} \ell_{\log}(\boldsymbol{w}^\top \boldsymbol{x}; y) = -y \boldsymbol{x} \cdot \sigma(-y \boldsymbol{w}^\top \boldsymbol{x}). \qquad (5.15)$$

Recall that due to the symmetry of σ around 0, $\sigma(-y \boldsymbol{w}^\top \boldsymbol{x})$ is the probability that $\boldsymbol{x}$ was *not* classified as y. Intuitively, if the model is "surprised" by the label, the gradient is large.

We can therefore use SGD with the (regularized) gradient step and with batch size 1,

$$\boldsymbol{w} \leftarrow \boldsymbol{w}(1 - 2\lambda\eta_t) + \eta_t y \boldsymbol{x} \sigma(-y \boldsymbol{w}^\top \boldsymbol{x}), \qquad (5.16)$$

for the data point $(\boldsymbol{x}, y)$ picked uniformly at random from the training data. Here, $2\lambda\eta_t$ is due to the gradient of the regularization term, in effect, performing weight decay.

Example 5.2: Laplace approx. of Bayesian logistic regression

We have already found the mode of the posterior distribution, $\hat{\boldsymbol{w}}$.

Let us denote by

$$\pi_i \doteq \mathbb{P}(y_i = 1 \mid \boldsymbol{x}_i, \hat{\boldsymbol{w}}) = \sigma(\hat{\boldsymbol{w}}^\top \boldsymbol{x}_i) \qquad (5.17)$$

the probability of $\boldsymbol{x}_i$ belonging to the positive class under the model given by the MAP estimate of the weights. For the precision matrix, we then have

$$\begin{aligned} \boldsymbol{\Lambda} &= -\boldsymbol{H}_{\boldsymbol{w}} \log p(\boldsymbol{w} \mid \boldsymbol{x}_{1:n}, y_{1:n})\big|_{\boldsymbol{w}=\hat{\boldsymbol{w}}} \\ &= -\boldsymbol{H}_{\boldsymbol{w}} \log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{w})\big|_{\boldsymbol{w}=\hat{\boldsymbol{w}}} - \boldsymbol{H}_{\boldsymbol{w}} \log p(\boldsymbol{w})\big|_{\boldsymbol{w}=\hat{\boldsymbol{w}}} \\ &= \sum_{i=1}^{n} \boldsymbol{H}_{\boldsymbol{w}} \ell_{\log}(\boldsymbol{w}^\top \boldsymbol{x}_i; y_i)\Big|_{\boldsymbol{w}=\hat{\boldsymbol{w}}} + \sigma_p^{-2} \boldsymbol{I} \end{aligned}$$

using the definition of the logistic loss (5.14)

$$= \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^\top \pi_i(1-\pi_i) + \sigma_p^{-2}\boldsymbol{I}$$
$$= \boldsymbol{X}^\top \operatorname{diag}_{i\in[n]}\{\pi_i(1-\pi_i)\}\boldsymbol{X} + \sigma_p^{-2}\boldsymbol{I}. \tag{5.18}$$

using the Hessian of the logistic loss (5.19)

Observe that $\pi_i(1-\pi_i) \approx 0$ if $\pi_i \approx 1$ or $\pi_i \approx 0$. That is, if a training example is "well-explained" by $\hat{\boldsymbol{w}}$, then its contribution to the precision matrix is small. In contrast, we have $\pi_i(1-\pi_i) = 0.25$ for $\pi_i = 0.5$. Importantly, $\boldsymbol{\Lambda}$ does not depend on the normalization constant of the posterior distribution which is hard to compute.

Overall, we have that $\mathcal{N}(\hat{\boldsymbol{w}}, \boldsymbol{\Lambda}^{-1})$ is the Laplace approximation of $p(\boldsymbol{w} \mid \boldsymbol{x}_{1:n}, y_{1:n})$.

Exercise 5.3: Logistic loss

1. Derive the gradient of $\ell_{\log}$ as given in eq. (5.15).
2. Show that

$$\boldsymbol{H}_{\boldsymbol{w}} \ell_{\log}(\boldsymbol{w}^\top \boldsymbol{x}; y) = \boldsymbol{x}\boldsymbol{x}^\top \cdot \sigma(\boldsymbol{w}^\top \boldsymbol{x}) \cdot (1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x})). \tag{5.19}$$

   Hint: Begin by deriving the first derivative of the logistic function, and use the chain rule of multivariate calculus,

$$\underbrace{\boldsymbol{D}_{\boldsymbol{x}}(\boldsymbol{f} \circ \boldsymbol{g})}_{\mathbb{R}^n \to \mathbb{R}^{m\times n}} = \underbrace{(\boldsymbol{D}_{\boldsymbol{g}(\boldsymbol{x})}\boldsymbol{f} \circ \boldsymbol{g}) \cdot \boldsymbol{D}_{\boldsymbol{x}}\boldsymbol{g}}_{\mathbb{R}^n \to \mathbb{R}^{m\times k}.\mathbb{R}^{k\times n}} \tag{5.20}$$

   where $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^k$ *and* $\boldsymbol{f} : \mathbb{R}^k \to \mathbb{R}^m$.
3. Is the logistic loss $\ell_{\log}$ convex in $\boldsymbol{w}$?

▷ *Solution*

5.2 *Inference with a Variational Posterior*

How can we perform inference using our variational approximation? We simply approximate the (intractable) true posterior with our variational posterior,

$$p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, y_{1:n}) = \int p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}, y_{1:n})\, d\boldsymbol{\theta}$$
$$\approx \int p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}) q_\lambda(\boldsymbol{\theta})\, d\boldsymbol{\theta} \tag{5.21}$$

using the sum rule (1.10)

At first sight, it may seem as though we did not gain much by our approximation. However, observe that interpreting the integral from a function-space view, the final prediction $y^\star$ is conditionally independent from the model parameters $\boldsymbol{\theta}$ given the "latent value" $f^\star$:[3]

$$= \int\int p(y^\star \mid f^\star) p(f^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}) q_\lambda(\boldsymbol{\theta})\, d\boldsymbol{\theta}\, df^\star$$

[3] This is discussed in greater detail in section 2.6.

once more, using the sum rule (1.10)

$$= \int p(y^\star \mid f^\star) \int p(f^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}) q_\lambda(\boldsymbol{\theta}) \, d\boldsymbol{\theta} \, df^\star \quad (5.22)$$

rearranging terms

Generally, $f^\star = g(\boldsymbol{\theta}; \boldsymbol{x}^\star)$ where g is a deterministic function representing our "model" parameterized by $\boldsymbol{\theta}$. Then, $p(f^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}) = \delta_{g(\boldsymbol{\theta}; \boldsymbol{x}^\star)}(f^\star)$ is a point density (cf. example 1.18) and

$$(g(\cdot; \boldsymbol{x}^\star)_\sharp q_\lambda)(f^\star) = \int p(f^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}) q_\lambda(\boldsymbol{\theta}) \, d\boldsymbol{\theta} \quad (5.23)$$

using sum rule (1.10) and product rule (1.14) to condition on $\boldsymbol{\theta}$

is the pushforward of $q_\lambda(\boldsymbol{\theta})$ under perturbation $g(\cdot; \boldsymbol{x}^\star)$. We discuss such a "change of variables" in greater detail in section 1.1.12. Alternatively, the integral in eq. (5.23) can be viewed as an expectation over the variational posterior q_λ and approximated using Monte Carlo sampling.

Example 5.4: Inference for Bayesian logistic regression

In the case of Bayesian logistic regression with a Laplace approximation, the posterior over weights $q_\lambda(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}; \hat{\boldsymbol{w}}, \boldsymbol{\Lambda}^{-1})$ is a Gaussian and $f^\star = g(\boldsymbol{w}; \boldsymbol{x}^\star) = \boldsymbol{w}^\top \boldsymbol{x}^\star$ given fixed weights $\boldsymbol{w}$. We therefore have due to the closedness properties of Gaussians (1.123),

$$g(\cdot; \boldsymbol{x}^\star)_\sharp q_\lambda = \mathcal{N}(\hat{\boldsymbol{w}}^\top \boldsymbol{x}^\star, \boldsymbol{x}^{\star\top} \boldsymbol{\Lambda}^{-1} \boldsymbol{x}^\star). \quad (5.24)$$

Crucially, this is a one-dimensional Gaussian!

As we have seen in eq. (5.12), for Bayesian logistic regression, the prediction $y^\star$ depends deterministically on the predicted "latent value" $f^\star$: $p(y^\star \mid f^\star) = \sigma(y^\star f^\star)$. Combining equations (5.22), (5.23), and (5.24), we obtain

$$p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, y_{1:n}) \approx \int \sigma(y^\star f^\star) \cdot \mathcal{N}(f^\star; \hat{\boldsymbol{w}}^\top \boldsymbol{x}^\star, \boldsymbol{x}^{\star\top} \boldsymbol{\Lambda}^{-1} \boldsymbol{x}^\star) \, df^\star. \quad (5.25)$$

We have replaced the (high-dimensional) integral over the model parameters $\boldsymbol{\theta}$ by the (one-dimensional) integral over the prediction of our variational posterior $f^\star$. While this integral is generally still intractable, it can be approximated efficiently using numerical quadrature methods such as the Gauss-Legendre quadrature.

Exercise 5.5: Gaussian process classification

In this exercise, we will study the use of Gaussian processes for classification tasks, commonly called *Gaussian process classification* (GPC). Linear logistic regression is extended to GPC by replacing

the Gaussian prior over weights with a GP prior on f,

$$f \sim \mathcal{GP}(0, k), \quad y \mid \boldsymbol{x}, f \sim \text{Bern}(\sigma(f(\boldsymbol{x}))) \tag{5.26}$$

where $\sigma : \mathbb{R} \to (0, 1)$ is some logistic-type function. Note that Bayesian logistic regression is the special case where k is the linear kernel and σ is the logistic function. This is analogous to the relationship of Bayesian linear regression and Gaussian process regression.

In the GP regression setting of chapter 4, y_i was assumed to be a noisy observation of $f(\boldsymbol{x}_i)$. In the classification setting, we now have that $y_i \in \{-1, +1\}$ is a binary class label and $f(\boldsymbol{x}_i) \in \mathbb{R}$ is a latent value. We study the setting where $\sigma(z) = \Phi(z; 0, \sigma_n^2)$ is the CDF of a univariate Gaussian with mean 0 and variance σ_n^2, also called a *probit likelihood*.

To make probabilistic predictions for a query point $\boldsymbol{x}^\star$, we first compute the distribution of the latent variable $f^\star$,

$$p(f^\star \mid \boldsymbol{x}_{1:n}, y_{1:n}, \boldsymbol{x}^\star) = \int p(f^\star \mid \boldsymbol{x}_{1:n}, \boldsymbol{x}^\star, \boldsymbol{f}) p(\boldsymbol{f} \mid x_{1:n}, y_{1:n}) \, d\boldsymbol{f} \tag{5.27}$$

using sum rule (1.10) and product rule (1.14), and $f^\star \perp y_{1:n} \mid \boldsymbol{f}$

where $p(\boldsymbol{f} \mid \boldsymbol{x}_{1:n}, y_{1:n})$ is the posterior over the latent variables.

1. Assuming that we can efficiently compute $p(f^\star \mid \boldsymbol{x}_{1:n}, y_{1:n}, \boldsymbol{x}^\star)$ (approximately), describe how we can find the predictive posterior $p(y^\star = +1 \mid \boldsymbol{x}_{1:n}, y_{1:n}, \boldsymbol{x}^\star)$.
2. The posterior over the latent variables is not a Gaussian as we used a non-Gaussian likelihood, and hence, the integral of the latent predictive posterior (5.27) is analytically intractable. A common technique is to approximate the latent posterior $p(\boldsymbol{f} \mid \boldsymbol{x}_{1:n}, y_{1:n})$ with a Gaussian using a Laplace approximation $q \doteq \mathcal{N}(\hat{\boldsymbol{f}}, \boldsymbol{\Lambda}^{-1})$. It is generally not possible to obtain an analytical representation of the mode of the Laplace approximation $\hat{\boldsymbol{f}}$. Instead, $\hat{\boldsymbol{f}}$ is commonly found using a second-order optimization scheme such as Newton's method.
   (a) Find the precision matrix $\boldsymbol{\Lambda}$ of the Laplace approximation. *Hint: Observe that for a label $y_i \in \{-1, +1\}$, the probability of a correct classification given the latent value f_i is $p(y_i \mid f_i) = \sigma(y_i f_i)$, where we use the symmetry of the probit likelihood around 0.*
   (b) Assume that $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}'$ is the linear kernel ($\sigma_p = 1$) and that σ is the logistic function (5.10). Show for this setting that the matrix $\boldsymbol{\Lambda}$ derived in (a) is equivalent to the precision matrix $\boldsymbol{\Lambda}'$ of the Laplace approximation of

Bayesian logistic regression (5.18).[4] You may assume that $\hat{f}_i = \hat{\boldsymbol{w}}^\top \boldsymbol{x}_i$.
Hint: First derive under which condition $\boldsymbol{\Lambda}$ and $\boldsymbol{\Lambda}'$ are "equivalent".

(c) Observe that the (approximate) latent predictive posterior

$$q(f^\star \mid \boldsymbol{x}_{1:n}, y_{1:n}, \boldsymbol{x}^\star) \doteq \int p(f^\star \mid \boldsymbol{x}_{1:n}, \boldsymbol{x}^\star, \boldsymbol{f}) q(\boldsymbol{f} \mid x_{1:n}, y_{1:n})\, d\boldsymbol{f}$$

which uses the Laplace approximation of the latent posterior is Gaussian.[5] Determine its mean and variance.
Hint: Condition on the latent variables $\boldsymbol{f}$ using the laws of total expectation and variance.

(d) Compare the prediction $p(f^\star \mid \boldsymbol{x}_{1:n}, y_{1:n}, \boldsymbol{x}^\star)$ you obtained in (1) (but now using the Laplace-approximated latent predictive posterior) to the prediction $\sigma(\mathbb{E}_{f^\star \sim q}[f^\star])$. Are they identical? If not, describe how they are different.

3. The use of the probit likelihood may seem arbitrary. Consider the following model which may be more natural,

$$f \sim \mathcal{GP}(0, k), \quad y = \mathbb{1}\{ \underbrace{f(\boldsymbol{x}) + \epsilon}_{\text{GP regression}} \geq 0\}, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2). \tag{5.28}$$

Show that the model from eq. (5.26) using a noise-free latent process with probit likelihood $\Phi(z; 0, \sigma_n^2)$ is equivalent (in expectation over ϵ) to the model from eq. (5.28).

▷ *Solution*

[4] This should not be surprising since — as already mentioned — Gaussian process classification is a generalization of Bayesian logistic regression.

[5] Using the Laplace-approximated latent posterior, $[f^\star \ \boldsymbol{f}]$ are jointly Gaussian. Thus, it directly follows from theorem 1.66 that the marginal distribution over $f^\star$ is also a Gaussian.

5.3 Blueprint of Variational Inference

General Bayesian inference poses the challenge of approximating the optimal Bayesian posterior distribution with limited memory and computation, resource constraints also present in humans and other intelligent systems. These resource constraints require information to be compressed, and as we will see, such a compression poses a fundamental trade-off between model accuracy (on the observed data) and model complexity (to avoid overfitting).

Laplace approximation approximates the true (intractable) posterior with a simpler one, by greedily matching mode and curvature around it. Can we find "less greedy" approaches? We can view variational Bayesian inference more generally as a family of approaches aiming to approximate the true posterior distribution by one that is closest (according to some criterion) among a "simpler" class of distributions. To

this end, we need to fix a class of distributions and define suitable criteria, which we can then optimize numerically. The key benefit is that we can reduce the (generally intractable) problem of high-dimensional integration to the (often much more tractable) problem of optimization.

Definition 5.6 (Variational family)**.** Let $\mathcal{P}$ be the class of all probability distributions. A *variational family* $\mathcal{Q} \subseteq \mathcal{P}$ is a class of distributions such that each distribution $q \in \mathcal{Q}$ is characterized by unique variational parameters $\boldsymbol{\lambda} \in \Lambda$.

Example 5.7: Family of independent Gaussians

A straightforward example for a variational family is the family of independent Gaussians,

$$\mathcal{Q} \doteq \left\{ q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}, \operatorname{diag}_{i \in [d]} \{\sigma_i^2\}) \right\}, \tag{5.29}$$

which is parameterized by $\boldsymbol{\lambda} \doteq [\mu_{1:d}, \sigma_{1:d}^2]$. Such a multivariate distribution where all variables are independent is called a *mean-field distribution*. Importantly, this family of distributions is characterized by only $2d$ parameters!

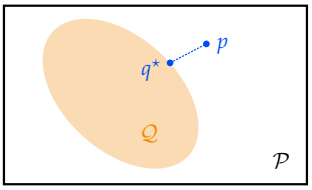


Figure 5.4: An illustration of variational inference in the space of distributions $\mathcal{P}$. The variational distribution $q^\star \in \mathcal{Q}$ is the optimal approximation of the true posterior p.

A common notion of distance between two distributions q and p is the Kullback-Leibler divergence $\mathrm{KL}(q\|p)$ which we will define in the next section. Using this notion of distance, we need to solve the following optimization problem:

$$q^\star \doteq \arg\min_{q \in \mathcal{Q}} \mathrm{KL}(q\|p) = \arg\min_{\lambda \in \Lambda} \mathrm{KL}(q_\lambda\|p). \tag{5.30}$$

In section 5.4, we introduce information theory and the Kullback-Leibler divergence. Then, in section 5.5, we discuss how the optimization problem of eq. (5.30) can be solved efficiently.

5.4 *Information Theory*

One of our main objectives throughout this course is to capture the "uncertainty" about events A in an appropriate probability space. One very natural measure of uncertainty is their probability, $\mathbb{P}(A)$. In this section, we will introduce an alternative measure of uncertainty, namely the so-called "suprise" about the event A.

5.4.1 Surprise

The *surprise* about an event with probability u is defined as

$$\mathrm{S}[u] \doteq -\log u. \tag{5.31}$$

Observe that the surprise is a function from $\mathbb{R}_{\geq 0}$ to $\mathbb{R}$, where we let $\mathrm{S}[0] \equiv \infty$. Moreover, for a discrete random variable X, we have that $p(x) \leq 1$, and hence, $\mathrm{S}[p(x)] \geq 0$. But why is it reasonable to measure surprise by $-\log u$?

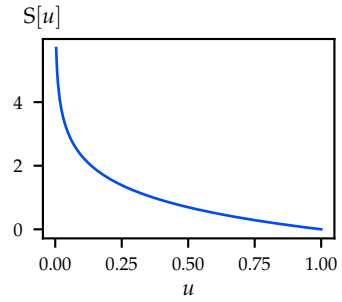


Figure 5.5: Surprise $\mathrm{S}[u]$ associated with an event of probability u.

Remarkably, it can be shown that the following natural axiomatic characterization leads to exactly this definition of surprise.

Theorem 5.8 (Axiomatic characterization of surprise)**.** *The axioms*

1. $\mathrm{S}[1] = 0$ — *certain events are not surprising*
2. $\mathrm{S}[u] > \mathrm{S}[v] \implies u < v$ *(anti-monotonicity)* — *we are more surprised by unlikely events*
3. S *is continuous,* — *no jumps in surprise for infinitesimal changes of probability*
4. $\mathrm{S}[uv] = \mathrm{S}[u] + \mathrm{S}[v]$ *for independent events,*

characterize S *up to a positive constant factor.*

Proof. Cauchy's functional equation, $f(x+y) = f(x) + f(y)$, has the unique family of solutions $\{f : x \mapsto cx : c \in \mathbb{R}\}$ if f is required to be continuous. Such a solution is called an "additive function". Consider the function $g(x) \doteq f(e^x)$. Then, g is additive if and only if

$$f(e^x e^y) = f(e^{x+y}) = g(x+y) = g(x) + g(y) = f(e^x) + f(e^y).$$

It therefore follows from the third and fourth axioms of surprise that $\mathrm{S}[u] = c \log u$ for any $c \in \mathbb{R}$. The second axiom of surprise implies that $c < 0$, and thus, $\mathrm{S}[u] = -c' \log u$ for any $c' > 0$. Finally, observe that the first axiom of surprise is satisfied for any such c'. □

Importantly, surprise offers a different perspective on uncertainty as opposed to probability: the uncertainty about an event can either be interpreted in terms of its probability or in terms of its surprise, and the two "spaces of uncertainty" are related by a log-transform. This relationship is illustrated in fig. 5.6. Information theory is the study of uncertainty in terms of surprise.

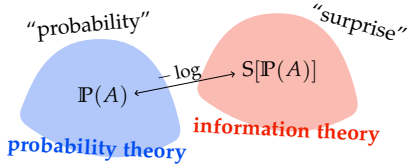


Figure 5.6: Illustration of the probability space and the corresponding "surprise space".

Throughout this course we will see many examples where modeling uncertainty in terms of surprise (i.e., the information-theoretic interpretation of uncertainty) is useful. One example where we have already encountered the "surprise space" was in the context of likelihood maximization (cf. section 1.4.3) where we used that the log-transform linearizes products of probabilities. We will see later in section 6.3.2 that in many cases the surprise $\mathrm{S}[p(x)]$ can also be interpreted as a "cost" or "energy" associated with the state x.

5.4.2 *Entropy*

The *entropy* of a distribution p is the average surprise about samples from p. In this way, entropy is a notion of uncertainty associated with the distribution p: if the entropy of p is large, we are more uncertain about $x \sim p$ than if the entropy of p were low. Formally,

$$\mathrm{H}[p] \doteq \mathbb{E}_{x\sim p}[\mathrm{S}[p(x)]] = \mathbb{E}_{x\sim p}[-\log p(x)]. \tag{5.32}$$

When $\mathbf{X} \sim p$ is a random vector distributed according to p, we write $\mathrm{H}[\mathbf{X}] \doteq \mathrm{H}[p]$. Observe that by definition, if p is discrete then $\mathrm{H}[p] \geq 0$ as $p(x) \leq 1 \ (\forall x)$.[6] For discrete distributions it is common to use the logarithm with base 2 rather than the natural logarithm.[7] Thus, entropy is given by

$$\mathrm{H}[p] = -\sum_x p(x) \log_2 p(x) \qquad \text{(if } p \text{ is discrete)}, \tag{5.33a}$$

$$\mathrm{H}[p] = -\int p(\boldsymbol{x}) \log p(\boldsymbol{x})\, d\boldsymbol{x} \qquad \text{(if } p \text{ is continuous)}. \tag{5.33b}$$

From the fourth axiom of surprise, we immediately obtain the following property of entropy.

Corollary 5.9. *For a multivariate distribution p that can be factorized into $p(x_{1:d}) = \prod_{i=1}^{d} p_i(x_i)$, we have that*

$$\mathrm{H}[p] = \sum_{i=1}^{d} \mathrm{H}[p_i]. \tag{5.34}$$

Example 5.10: Examples of entropy

- *Fair Coin* $\mathrm{H}[\mathrm{Bern}(0.5)] = -2(0.5 \log_2 0.5) = 1.$
- *Unfair Coin*

$$\mathrm{H}[\mathrm{Bern}(0.1)] = -0.1 \log_2 0.1 - 0.9 \log_2 0.9 \approx 0.469.$$

- *Uniform Distribution*

$$\mathrm{H}[\mathrm{Unif}(\{1, \ldots, n\})] = -\sum_{i=1}^{n} \frac{1}{n} \log_2 \frac{1}{n} = \log_2 n.$$

The uniform distribution has the maximum entropy among all discrete distributions supported on $\{1, \ldots, n\}$. Note that a fair coin corresponds to a uniform distribution with $n = 2$. Also observe that $\log_2 n$ corresponds to the number of bits required to encode the outcome of the experiment.

In general, the entropy $\mathrm{H}[p]$ of a discrete distribution p can be interpreted as the average number of bits required to encode a

[6] The entropy of a continuous distribution can be negative. For example,

$$\mathrm{H}[\mathrm{Unif}([a,b])] = -\int \frac{1}{b-a} \log \frac{1}{b-a}\, dx = \log(b-a)$$

which is negative if $b - a < 1$.

[7] Recall that $\log_2 x = \frac{\log x}{\log 2}$, that is, logarithms to a different base only differ by a constant factor.

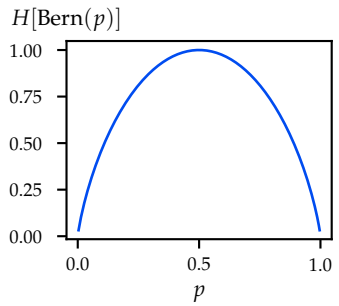


Figure 5.7: Entropy $\mathrm{H}[\mathrm{Bern}(p)]$ of a Bernoulli experiment with success probability p.

sample $x \sim p$, or in other words, the average "information" carried by a sample x.

Example 5.11: Entropy of a Gaussian

Let us derive the entropy of a univariate Gaussian. Recall the PDF,

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

where $Z = \sqrt{2\pi\sigma^2}$. Using the definition of entropy (5.33b), we obtain,

$$\begin{aligned}
\mathrm{H}\left[\mathcal{N}(\mu, \sigma^2)\right] &= -\int \frac{1}{Z} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \\
&\qquad \cdot \log\left(\frac{1}{Z} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\right) dx \\
&= \log Z \underbrace{\int \frac{1}{Z} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) dx}_{1} \\
&\qquad + \int \frac{1}{Z} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \frac{(x-\mu)^2}{2\sigma^2} dx \\
&= \log Z + \frac{1}{2\sigma^2} \mathbb{E}\left[(x-\mu)^2\right] && \text{using LOTUS (1.30)} \\
&= \log(\sigma\sqrt{2\pi}) + \frac{1}{2} && \text{using } \mathbb{E}\left[(x-\mu)^2\right] = \mathrm{Var}[x] = \sigma^2 \text{ (1.44)} \\
&= \log(\sigma\sqrt{2\pi e}). && \text{using } \log\sqrt{e} = 1/2 \qquad (5.35)
\end{aligned}$$

In general, the entropy of a Gaussian is

$$\mathrm{H}[\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})] = \frac{1}{2} \log\det(2\pi e \boldsymbol{\Sigma}) = \frac{1}{2} \log\left((2\pi e)^d \det(\boldsymbol{\Sigma})\right). \qquad (5.36)$$

Observe that the surprise $\mathrm{S}[u]$ is convex in u. Jensen's inequality is a useful tool when working with expectations of convex functions such as entropy.

Fact 5.12 (Jensen's Inequality). *Given a random variable $X : \Omega \to \mathbb{R}$ and a convex function $g : \mathbb{R} \to \mathbb{R}$, we have*

$$g(\mathbb{E}[X]) \leq \mathbb{E}[g(X)]. \qquad (5.37)$$

For a concave function h, $-h$ is convex, and hence

$$h(\mathbb{E}[X]) \geq \mathbb{E}[h(X)]. \qquad (5.38)$$

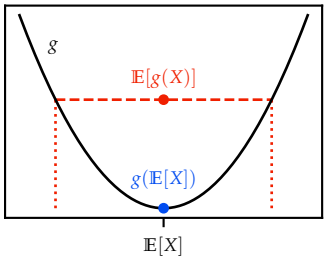


Figure 5.8: An illustration of Jensen's inequality. Due to the convexity of g, we have that g evaluated at $\mathbb{E}[X]$ will always be below the average of evaluations of g.

Exercise 5.13: Jensen's inequality

1. Prove the finite form of Jensen's inequality.
   Theorem 5.14 (Jensen's inequality, finite form)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. Suppose that $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k \in \mathbb{R}^n$ and $\theta_1, \dots, \theta_k \geq 0$ with $\theta_1 + \cdots + \theta_k = 1$. Then,*

$$f(\theta_1 \boldsymbol{x}_1 + \cdots + \theta_k \boldsymbol{x}_k) \leq \theta_1 f(\boldsymbol{x}_1) + \cdots + \theta_k f(\boldsymbol{x}_k). \tag{5.39}$$

   Observe that if X is a random variable with finite support, the above two versions of Jensen's inequality are equivalent.
2. Show that for any discrete distribution p supported on a finite domain of size n, $\mathrm{H}[p] \leq \log_2 n$.

▷ *Solution*

Readings

To read more about entropy, refer to chapter 2 of "Elements of information theory" (Cover and Thomas, 2006). In chapter 8, we introduce the related concepts of conditional entropy and mutual information.

5.4.3 Cross-Entropy

How can we use entropy to measure our average surprise when assuming the data follows some distribution q but in reality the data follows a different distribution p?

Definition 5.15 (Cross-entropy)**.** The *cross-entropy* of a distribution q relative to the distribution p is

$$\mathrm{H}[p\|q] \doteq \mathbb{E}_{x\sim p}[\mathrm{S}[q(x)]] = \mathbb{E}_{x\sim p}[-\log q(x)]. \tag{5.40}$$

Cross-entropy can also be expressed in terms of the KL-divergence (cf. section 5.4.4) $\mathrm{KL}(p\|q)$ which measures how "different" the distribution q is from a reference distribution p,

$$\mathrm{H}[p\|q] = \mathrm{H}[p] + \mathrm{KL}(p\|q) \geq \mathrm{H}[p]. \tag{5.41}$$

$\mathrm{KL}(p\|q) \geq 0$ is shown in exercise 5.19

Quite intuitively, the average surprise in samples from p with respect to the distribution q is given by the inherent uncertainty in p and the additional surprise that is due to us assuming the wrong data distribution q. The "closer" q is to the true data distribution p, the smaller is the additional average surprise.

Remark 5.16: Minimizing cross-entropy

Minimizing cross-entropy is a commonly used objective for *density estimation* problems, where we want to learn some true distribution p using a parameterized distribution $q_{\boldsymbol{\theta}}$. As $\mathrm{H}[p]$ is fixed, minimizing cross-entropy corresponds to minimizing the KL-divergence between p and $q_{\boldsymbol{\theta}}$.

For example, in a binary classification problem with the label $y \in \{0, 1\}$ and predicted class probability $\hat{y} \in [0, 1]$ for some fixed input, it is natural to use cross-entropy as a measure of dissimilarity between y and $\hat{y}$,

$$\begin{aligned} \ell_{\text{bce}}(\hat{y}; y) &\doteq \mathrm{H}[\mathrm{Bern}(y) \| \mathrm{Bern}(\hat{y})] \\ &= -\sum_{x \in \{0,1\}} \mathrm{Bern}(x; y) \log \mathrm{Bern}(x; \hat{y}) \\ &= -y \log \hat{y} - (1 - y) \log(1 - \hat{y}). \end{aligned} \tag{5.42}$$

This loss function is also known as the *binary cross-entropy loss*.

Exercise 5.17: Binary cross-entropy loss

Show that the logistic loss (5.14) is equivalent to the binary cross-entropy loss with $\hat{y} = \sigma(\hat{f})$. That is,

$$\ell_{\log}(\hat{f}; y) = \ell_{\text{bce}}(\hat{y}; y). \tag{5.43}$$

▷ *Solution*

5.4.4 Kullback-Leibler Divergence

As mentioned, the Kullback-Leibler divergence is a (non-metric) measure of distance between distributions. It is defined as follows.

Definition 5.18 (Kullback-Leibler divergence, KL-divergence)**.** Given two distributions p and q, the *Kullback-Leibler divergence* (or *relative entropy*) of q with respect to p,

$$\mathrm{KL}(p \| q) \doteq \mathrm{H}[p \| q] - \mathrm{H}[p] \tag{5.44}$$

$$= \mathbb{E}_{\boldsymbol{\theta} \sim p}[\mathrm{S}[q(\boldsymbol{\theta})] - \mathrm{S}[p(\boldsymbol{\theta})]] \tag{5.45}$$

$$= \mathbb{E}_{\boldsymbol{\theta} \sim p}\left[\log \frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}\right], \tag{5.46}$$

measures how different q is from a reference distribution p.

In words, $\mathrm{KL}(p \| q)$ measures the *additional* expected surprise when observing samples from p that is due to assuming the (wrong) distribu-

tion q and which not inherent in the distribution p already.[8] Intuitively, the KL-divergence measures the information loss when approximating p with q.

[8] The KL-divergence only captures the additional expected surprise as the surprise inherent in p (as measured by $\mathrm{H}[p]$) is subtracted.

The KL-divergence has the following properties:

- $\mathrm{KL}(p\|q) \geq 0$ for any distributions p and q,
- $\mathrm{KL}(p\|q) = 0$ if and only if $p = q$ almost surely, and
- there exist distributions p and q such that $\mathrm{KL}(p\|q) \neq \mathrm{KL}(q\|p)$.

The KL-divergence can simply be understood as a shifted version of cross-entropy, which is zero if we consider the divergence between two identical distributions.

Exercise 5.19: Gibbs' inequality

1. Prove $\mathrm{KL}(p\|q) \geq 0$ which is also known as *Gibbs' inequality*.
2. Let p and q be discrete distributions with finite identical support A. Show that $\mathrm{KL}(p\|q) = 0$ if and only if $p \equiv q$.
   Hint: Use that if a function $f : \mathbb{R}^n \to \mathbb{R}$ is strictly convex and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in \mathbb{R}^n$, $\theta_1, \ldots, \theta_k \geq 0$, $\theta_1 + \cdots + \theta_k = 1$, we have that

$$f(\theta_1 \boldsymbol{x}_1 + \cdots + \theta_k \boldsymbol{x}_k) = \theta_1 f(\boldsymbol{x}_1) + \cdots + \theta_k f(\boldsymbol{x}_k) \tag{5.47}$$

   iff $\boldsymbol{x}_1 = \cdots = \boldsymbol{x}_k$. This is a slight adaptation of Jensen's inequality in finite-form, which you proved in exercise 5.13.

▷ *Solution*

Exercise 5.20: Maximum entropy principle

In this exercise we will prove that the normal distribution is the distribution with maximal entropy among all (univariate) distributions supported on $\mathbb{R}$ with fixed mean μ and variance σ^2. Let $g(x) \doteq \mathcal{N}(x; \mu, \sigma^2)$, and $f(x)$ be any distribution on $\mathbb{R}$ with mean μ and variance σ^2.

1. Prove that $\mathrm{KL}(f\|g) = \mathrm{H}[g] - \mathrm{H}[f]$.
   Hint: Equivalently, show that $\mathrm{H}[f\|g] = \mathrm{H}[g]$. That is, the expected surprise evaluated based on the Gaussian g is invariant to the true distribution f.
2. Conclude that $\mathrm{H}[g] \geq \mathrm{H}[f]$.

▷ *Solution*

We will briefly look at another interpretation for how KL-divergence measures "distance" between distributions. Suppose we are presented with a sequence $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n$ of samples from either a distribution p or a distribution q, both of which are known. Which of p or q was used to generate the data is, however, unknown to us, and we would like to

find out. A natural approach is to choose the distribution whose data likelihood is larger. That is, we choose p if $p(\boldsymbol{\theta}_{1:n}) > q(\boldsymbol{\theta}_{1:n})$ and vice versa. Assuming that the samples are independent and rewriting the inequality slightly, we choose p if

$$\prod_{i=1}^{n} \frac{p(\boldsymbol{\theta}_i)}{q(\boldsymbol{\theta}_i)} > 1, \quad \text{or equivalently if} \quad \sum_{i=1}^{n} \log \frac{p(\boldsymbol{\theta}_i)}{q(\boldsymbol{\theta}_i)} > 0.$$

Assume that $\boldsymbol{\theta}_i \sim p$. Then, using the law of large numbers (1.84),

$$\frac{1}{n} \sum_{i=1}^{n} \log \frac{p(\boldsymbol{\theta}_i)}{q(\boldsymbol{\theta}_i)} \xrightarrow{\text{a.s.}} \mathbb{E}_{\boldsymbol{\theta} \sim p} \left[\log \frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} \right] = \mathrm{KL}(p \| q) \tag{5.48}$$

as $n \to \infty$. Recall that assuming $p \neq q$, we have that $\mathrm{KL}(p\|q) > 0$ with probability 1. In particular, this shows that as $n \to \infty$, we correctly choose p with probability 1.

Example 5.21: KL-divergence of Bernoulli random variables

Suppose we are given two Bernoulli distributions $\mathrm{Bern}(p)$ and $\mathrm{Bern}(q)$. Then, their KL-divergence is

$$\begin{aligned} \mathrm{KL}(\mathrm{Bern}(p) \| \mathrm{Bern}(q)) &= \sum_{x \in \{0,1\}} \mathrm{Bern}(x; p) \log \frac{\mathrm{Bern}(x; p)}{\mathrm{Bern}(x; q)} \\ &= p \log \frac{p}{q} + (1-p) \log \frac{(1-p)}{(1-q)}. \end{aligned} \tag{5.49}$$

Exercise 5.22: KL-divergence of Gaussians

Suppose we are given two Gaussian distributions $p \doteq \mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)$ and $q \doteq \mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$ with dimension d. Show that the KL-divergence of p and q is given by

$$\begin{aligned} \mathrm{KL}(p\|q) = \frac{1}{2} \Bigg(&\mathrm{tr}(\boldsymbol{\Sigma}_q^{-1} \boldsymbol{\Sigma}_p) + (\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)^\top \boldsymbol{\Sigma}_q^{-1} (\boldsymbol{\mu}_p - \boldsymbol{\mu}_q) \\ &- d + \log \frac{\det(\boldsymbol{\Sigma}_q)}{\det(\boldsymbol{\Sigma}_p)} \Bigg). \end{aligned} \tag{5.50}$$

Hint: If $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ in d dimensions, then we have that for any $\boldsymbol{m} \in \mathbb{R}^d$ and $\boldsymbol{A} \in \mathbb{R}^{d \times d}$,

$$\mathbb{E}\left[(\mathbf{X} - \boldsymbol{m})^\top \boldsymbol{A} (\mathbf{X} - \boldsymbol{m}) \right] = (\boldsymbol{\mu} - \boldsymbol{m})^\top \boldsymbol{A} (\boldsymbol{\mu} - \boldsymbol{m}) + \mathrm{tr}(\boldsymbol{A}\boldsymbol{\Sigma}) \tag{5.51}$$

▷ *Solution*

Example 5.23: KL-divergence of independent Gaussians

For independent Gaussians with unit variance $p \doteq \mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{I})$ and $q \doteq \mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{I})$, the expression simplifies to the squared Euclidean distance,

$$\mathrm{KL}(p\|q) = \frac{1}{2}\left\|\boldsymbol{\mu}_q - \boldsymbol{\mu}_p\right\|_2^2. \tag{5.52}$$

If we approximate independent Gaussians with variances σ_i^2,

$$p \doteq \mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}\{\sigma_1^2, \ldots, \sigma_d^2\}),$$

by a standard normal distribution, $q \doteq \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, the expression simplifies to

$$\mathrm{KL}(p\|q) = \frac{1}{2}\sum_{i=1}^{d}(\sigma_i^2 + \mu_i^2 - 1 - \log\sigma_i^2). \tag{5.53}$$

Here, the term μ_i^2 penalizes a large mean of p, the term σ_i^2 penalizes a large variance of p, and the term $-\log\sigma_i^2$ penalizes a small variance of p. As expected, $\mathrm{KL}(p\|q)$ is proportional to the amount of information we lose by approximating p with the simpler distribution q.

5.4.5 *Forward and Reverse KL-divergence*

$\mathrm{KL}(p\|q)$ is also called the *forward* (or *inclusive*) KL-divergence. In contrast, $\mathrm{KL}(q\|p)$ is called the *reverse* (or *exclusive*) KL-divergence. Figure 5.9 shows the approximations of a general Gaussian obtained when $\mathcal{Q}$ is the family of diagonal (independent) Gaussians. Thereby,

$$q_1^\star \doteq \arg\min_{q\in\mathcal{Q}} \mathrm{KL}(p\|q) \qquad q_2^\star \doteq \arg\min_{q\in\mathcal{Q}} \mathrm{KL}(q\|p)$$

$q_1^\star$ is the result when using the forward KL-divergence and $q_2^\star$ is the result when using reverse KL-divergence. It can be seen that the reverse KL-divergence tends to greedily select the mode and underestimating the variance which, in this case, leads to an overconfident prediction. The forward KL-divergence, in contrast, is more conservative and yields what one could consider the "desired" approximation.

Recall that in the blueprint of variational inference (5.30) we used the reverse KL-divergence. This is for computational reasons. Observe that to approximate the KL-divergence $\mathrm{KL}(p\|q)$ using Monte Carlo sampling, we would need to obtain samples from p yet p is the intractable posterior distribution which we were trying to approximate in the first place. Crucially, observe that if the true posterior

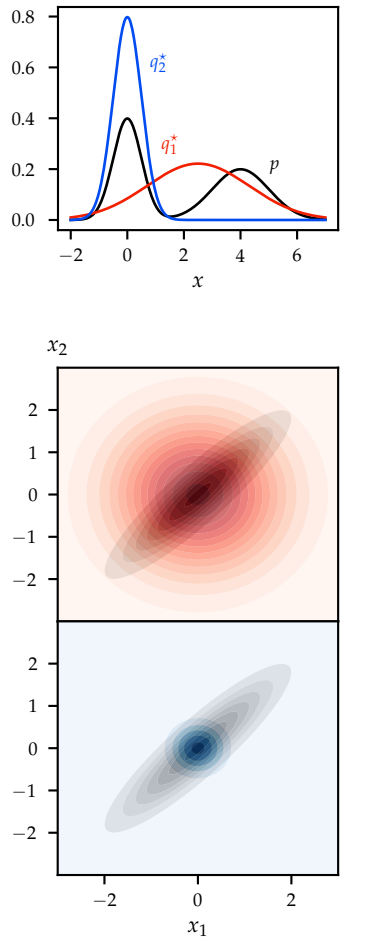


Figure 5.9: Comparison of the **forward** KL-divergence $q_1^\star$ and the **reverse** KL-divergence $q_2^\star$ when used to approximate the **true posterior** p. The first plot shows the PDFs in a one-dimensional feature space where p is a mixture of two univariate Gaussians. The second plot shows contour lines of the PDFs in a two-dimensional feature space where the non-diagonal Gaussian p is approximated by diagonal Gaussians $q_1^\star$ and $q_2^\star$. It can be seen that $q_1^\star$ selects the variance and $q_2^\star$ selects the mode of p. The approximation $q_1^\star$ is more conservative than the (overconfident) approximation $q_2^\star$.

$p(\cdot \mid \boldsymbol{x}_{1:n}, y_{1:n})$ is in the variational family $\mathcal{Q}$, then

$$\arg\min_{q \in \mathcal{Q}} \mathrm{KL}(q \| p(\cdot \mid \boldsymbol{x}_{1:n}, y_{1:n})) \overset{\text{a.s.}}{=} p(\cdot \mid \boldsymbol{x}_{1:n}, y_{1:n}) \tag{5.54}$$

as $\min_{q \in \mathcal{Q}} \mathrm{KL}(q \| p(\cdot \mid \boldsymbol{x}_{1:n}, y_{1:n})) \overset{\text{a.s.}}{=} 0$, so minimizing reverse-KL still recovers the true posterior almost surely.

Remark 5.24: Greediness of reverse-KL

As in the previous example, consider the independent Gaussians

$$p \doteq \mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}_{i \in [d]}\{\sigma_i^2\}),$$

which we seek to approximate by a standard normal distribution $q \doteq \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Using (5.50), we obtain for the reverse KL-divergence,

$$\begin{aligned} \mathrm{KL}(q \| p) &= \frac{1}{2}\Big(\mathrm{tr}\Big(\mathrm{diag}\{\sigma_i^{-2}\}\Big) + \boldsymbol{\mu}^\top \mathrm{diag}\{\sigma_i^{-2}\}\boldsymbol{\mu} - d \\ &\qquad + \log\det\Big(\mathrm{diag}\{\sigma_i^2\}\Big)\Big) \\ &= \frac{1}{2}\sum_{i=1}^{d}\left(\sigma_i^{-2} + \frac{\mu_i^2}{\sigma_i^2} - 1 + \log\sigma_i^2\right). \end{aligned} \tag{5.55}$$

Here, σ_i^{-2} penalizes small variance, μ_i^2/σ_i^2 penalizes a large mean, and $\log\sigma_i^2$ penalizes large variance. Compare this to the expression for the forward KL-divergence $\mathrm{KL}(p\|q)$ that we have seen in eq. (5.53). In particular, observe that reverse-KL penalizes large variance less strongly than forward-KL.

Note, however, that reverse-KL is not greedy in the same sense as Laplace approximation, as it does still take the variance into account and does not *purely* match the mode of p.

Exercise 5.25: Forward vs reverse KL

1. Consider a factored approximation $q(x, y) = q(x)q(y)$ to a joint distribution $p(x, y)$. Show that to minimize the forward $\mathrm{KL}(p\|q)$ we should set $q(x) = p(x)$ and $q(y) = p(y)$, i.e., the optimal approximation is a product of marginals.
2. Consider the following joint distribution p, where the rows represent y and the columns x,

	1	2	3	4
1	1/8	1/8	0	0
2	1/8	1/8	0	0
3	0	0	1/4	0
4	0	0	0	1/4

Show that the reverse $\mathrm{KL}(q\|p)$ for this p has three distinct minima. Identify those minima and evaluate $\mathrm{KL}(q\|p)$ at each of them.

3. What is the value of $\mathrm{KL}(q\|p)$ if we use the approximation $q(x, y) = p(x)p(y)$?

▷ *Solution*

5.4.6 *Minimizing Forward KL-Divergence*

Before completing the blueprint of variational inference in section 5.5 by showing how *reverse-KL* can be efficiently minimized, we relate minimizing *forward-KL* to two other well-known inference algorithms.

Minimizing Forward-KL as Maximum Likelihood Estimation First, we observe that minimizing the forward KL-divergence is equivalent to maximum likelihood estimation on an infinitely large sample size. The classical application of this result is in the setting where $p(\boldsymbol{x})$ is a generative model, and we aim to learn the parameterized model likelihood $q_{\boldsymbol{\lambda}}$.[9]

[9] Note that here, we aim to learn model parameters $\boldsymbol{\lambda}$ for estimating the probability of $\boldsymbol{x}$, whereas in the setting of variational Bayesian inference, we want to learn parameters $\boldsymbol{\lambda}$ of a distribution over $\boldsymbol{\theta}$ and $\boldsymbol{\theta}$ parameterizes a distribution over $\boldsymbol{x}$.

Lemma 5.26 (Forward KL-divergence as MLE)**.** *Given some generative model $p(\boldsymbol{x})$ and a likelihood $q_{\boldsymbol{\lambda}}(\boldsymbol{x}) = q(\boldsymbol{x} \mid \boldsymbol{\lambda})$ (that we use to approximate the true data distribution), we have*

$$\arg\min_{\boldsymbol{\lambda}\in\Lambda} \mathrm{KL}(p\|q_{\boldsymbol{\lambda}}) \overset{\text{a.s.}}{=} \arg\max_{\boldsymbol{\lambda}\in\Lambda} \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \log q(\boldsymbol{x}^{(i)} \mid \boldsymbol{\lambda}), \tag{5.56}$$

where $\boldsymbol{x}^{(i)} \overset{\text{iid}}{\sim} p$ are independent samples from the true data distribution.

Proof.

$$\begin{aligned}
\mathrm{KL}(p\|q_{\boldsymbol{\lambda}}) &= \mathrm{H}[p\|q_{\boldsymbol{\lambda}}] - \mathrm{H}[p] && \text{using the definition of KL-divergence (5.44)}\\
&= \mathbb{E}_{(\boldsymbol{x},y)\sim p}[-\log q(\boldsymbol{x} \mid \boldsymbol{\lambda})] + \text{const} && \text{dropping } \mathrm{H}[p] \text{ and using the definition of cross-entropy (5.40)}\\
&\overset{\text{a.s.}}{=} -\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \log q(\boldsymbol{x}^{(i)} \mid \boldsymbol{\lambda}) + \text{const} && \text{using Monte Carlo sampling, i.e., the law of large numbers (1.84)}
\end{aligned}$$

where $\boldsymbol{x}^{(i)} \overset{\text{iid}}{\sim} p$ are independent samples. □

This interpretation is not immediately useful for Bayesian inference (i.e., in the setting where p is a posterior distribution over model parameters $\boldsymbol{\theta}$) as a maximum likelihood estimate requires samples from p which we cannot obtain in this case.[10]

[10] It is possible to obtain "approximate" samples using Markov chain Monte Carlo (MCMC) methods which we discuss in chapter 6.

Minimizing Forward-KL as Moment Matching Now to a second interpretation of minimizing forward-KL. *Moment matching* (also known as the *method of moments*) is a technique for approximating an unknown distribution p with a parameterized distribution $q_{\boldsymbol{\lambda}}$ where $\boldsymbol{\lambda}$ is chosen such that $q_{\boldsymbol{\lambda}}$ matches the (estimated) moments of p. For example, given the estimates $\boldsymbol{a}$ and $\boldsymbol{B}$ of the first and second moment of p,[11] and if $q_{\boldsymbol{\lambda}}$ is a Gaussian with parameters $\boldsymbol{\lambda} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$, then moment matching chooses $\boldsymbol{\lambda}$ as the solution to

[11] These estimates are computed using the samples from p. For example, using a sample mean and a sample variance to compute the estimates of the first and second moment.

$$\begin{aligned}\mathbb{E}_p[\boldsymbol{\theta}] \approx \boldsymbol{a} &\overset{!}{=} \boldsymbol{\mu} = \mathbb{E}_{q_{\boldsymbol{\lambda}}}[\boldsymbol{\theta}] \\ \mathbb{E}_p\left[\boldsymbol{\theta}\boldsymbol{\theta}^\top\right] \approx \boldsymbol{B} &\overset{!}{=} \boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^\top = \mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\boldsymbol{\theta}\boldsymbol{\theta}^\top\right].\end{aligned}$$

using the definition of variance (1.45)

In general the number of moments to be matched (i.e., the number of equations) is adjusted such that it is equal to the number of parameters to be estimated. We will see now that the "matching" of moments is also ensured when $q_{\boldsymbol{\lambda}}$ is obtained by minimizing the forward KL-divergence within the family of Gaussians.

The Gaussian PDF can be expressed as

$$\mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{Z(\boldsymbol{\lambda})} \exp(\boldsymbol{\lambda}^\top \boldsymbol{s}(\boldsymbol{\theta})) \quad \text{where} \tag{5.57}$$

$$\boldsymbol{\lambda} \doteq \begin{bmatrix} \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ \operatorname{vec}[\boldsymbol{\Sigma}^{-1}] \end{bmatrix} \tag{5.58}$$

$$\boldsymbol{s}(\boldsymbol{\theta}) \doteq \begin{bmatrix} \boldsymbol{\theta} \\ \operatorname{vec}[-\frac{1}{2}\boldsymbol{\theta}\boldsymbol{\theta}^\top] \end{bmatrix} \tag{5.59}$$

and $Z(\boldsymbol{\lambda}) \doteq \int \exp(\boldsymbol{\lambda}^\top \boldsymbol{s}(\boldsymbol{\theta}))\, d\boldsymbol{\theta}$, and we will confirm this in just a moment.[12] The family of distributions with densities of the form (5.57) — with an additional scaling constant $h(\boldsymbol{\theta})$ which is often 1 — is called the *exponential family* of distributions. Here, $\boldsymbol{s}(\boldsymbol{\theta})$ are the *sufficient statistics*, $\boldsymbol{\lambda}$ are called the *natural parameters*, and $Z(\boldsymbol{\lambda})$ is the normalizing constant. In this context, $Z(\boldsymbol{\lambda})$ is often called the *partition function*.

[12] Given a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times m}$, we use

$$\operatorname{vec}[\boldsymbol{A}] \in \mathbb{R}^{n \cdot m}$$

to denote the row-by-row concatenation of $\boldsymbol{A}$ yielding a vector of length $n \cdot m$.

To see that the Gaussian is indeed part of the exponential family as promised in eqs. (5.58) and (5.59), consider

$$\begin{aligned}\mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) &\propto \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu})\right) \\ &\propto \exp\left(\operatorname{tr}\left(-\frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\theta}\right) + \boldsymbol{\theta}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right)\end{aligned}$$

expanding the inner product and using that $\operatorname{tr}(x) = x$ for all $x \in \mathbb{R}$

$$= \exp\left(\operatorname{tr}\left(-\frac{1}{2}\boldsymbol{\theta}\boldsymbol{\theta}^\top \boldsymbol{\Sigma}^{-1}\right) + \boldsymbol{\theta}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right)$$

using that the trace is invariant under cyclic permutations

$$= \exp\left(\operatorname{vec}[-\tfrac{1}{2}\boldsymbol{\theta}\boldsymbol{\theta}^\top]^\top \operatorname{vec}[\boldsymbol{\Sigma}^{-1}] + \boldsymbol{\theta}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right).$$

using $\operatorname{tr}(\boldsymbol{A}\boldsymbol{B}) = \operatorname{vec}[\boldsymbol{A}]^\top \operatorname{vec}[\boldsymbol{B}]$ for any $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$

This allows us to express the forward KL-divergence as

$$\begin{aligned}\mathrm{KL}(p\|q_\lambda) &= \int p(\boldsymbol{\theta}) \log \frac{p(\boldsymbol{\theta})}{q_\lambda(\boldsymbol{\theta})}\, d\boldsymbol{\theta} \\ &= -\int p(\boldsymbol{\theta}) \cdot \boldsymbol{\lambda}^\top \boldsymbol{s}(\boldsymbol{\theta})\, d\boldsymbol{\theta} + \log Z(\boldsymbol{\lambda}) + \text{const.}\end{aligned}$$

using that $\int p(\boldsymbol{\theta}) \log p(\boldsymbol{\theta})\, d\boldsymbol{\theta}$ is constant

Differentiating with respect to the natural parameters $\boldsymbol{\lambda}$ gives

$$\begin{aligned}\nabla_{\boldsymbol{\lambda}} \mathrm{KL}(p\|q_\lambda) &= -\int p(\boldsymbol{\theta})\boldsymbol{s}(\boldsymbol{\theta})\, d\boldsymbol{\theta} + \frac{1}{Z(\boldsymbol{\lambda})}\int \boldsymbol{s}(\boldsymbol{\theta}) \exp(\boldsymbol{\lambda}^\top \boldsymbol{s}(\boldsymbol{\theta}))\, d\boldsymbol{\theta} \\ &= -\mathbb{E}_{\boldsymbol{\theta}\sim p}[\boldsymbol{s}(\boldsymbol{\theta})] + \mathbb{E}_{\boldsymbol{\theta}\sim q_\lambda}[\boldsymbol{s}(\boldsymbol{\theta})].\end{aligned}$$

Hence, for any minimizer of $\mathrm{KL}(p\|q_\lambda)$, we have that the sufficient statistics under p and q_λ match:

$$\mathbb{E}_p[\boldsymbol{s}(\boldsymbol{\theta})] = \mathbb{E}_{q_\lambda}[\boldsymbol{s}(\boldsymbol{\theta})]. \tag{5.60}$$

Therefore, in the Gaussian case,

$$\mathbb{E}_p[\boldsymbol{\theta}] = \mathbb{E}_{q_\lambda}[\boldsymbol{\theta}] \quad \text{and} \quad \mathbb{E}_p\left[-\tfrac{1}{2}\boldsymbol{\theta}\boldsymbol{\theta}^\top\right] = \mathbb{E}_{q_\lambda}\left[-\tfrac{1}{2}\boldsymbol{\theta}\boldsymbol{\theta}^\top\right],$$

implying that

$$\mathbb{E}_p[\boldsymbol{\theta}] = \boldsymbol{\mu} \quad \text{and} \quad \operatorname{Var}_p[\boldsymbol{\theta}] = \mathbb{E}_p\left[\boldsymbol{\theta}\boldsymbol{\theta}^\top\right] - \mathbb{E}_p[\boldsymbol{\theta}] \cdot \mathbb{E}_p[\boldsymbol{\theta}]^\top = \boldsymbol{\Sigma} \tag{5.61}$$

using eq. (1.45)

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean and variance of the approximation q_λ, respectively. That is, a Gaussian q_λ minimizing $\mathrm{KL}(p\|q_\lambda)$ has the same first and second moment as p.

5.5 *Evidence Lower Bound*

To complete the blueprint of variational inference from section 5.3, it remains to show that the reverse KL-divergence can be minimized efficiently. We have

$$\begin{aligned}\mathrm{KL}(q\|p(\cdot \mid \boldsymbol{x}_{1:n}, y_{1:n})) &= \mathbb{E}_{\boldsymbol{\theta}\sim q}\left[\log \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}, y_{1:n})}\right] && \text{using the definition of the KL-divergence (5.44)} \\ &= \mathbb{E}_{\boldsymbol{\theta}\sim q}\left[\log \frac{p(y_{1:n} \mid \boldsymbol{x}_{1:n}) q(\boldsymbol{\theta})}{p(y_{1:n}, \boldsymbol{\theta} \mid \boldsymbol{x}_{1:n})}\right] && \text{using the definition of conditional probability (1.11)} \\ &= \log p(y_{1:n} \mid \boldsymbol{x}_{1:n}) && \text{using linearity of expectation (1.24)} \\ &\quad \underbrace{-\mathbb{E}_{\boldsymbol{\theta}\sim q}[\log p(y_{1:n}, \boldsymbol{\theta} \mid \boldsymbol{x}_{1:n})] - \mathrm{H}[q]}_{-L(q,p;\mathcal{D})}\end{aligned}$$

where $L(q, p; \mathcal{D})$ is called the *evidence lower bound* (ELBO) given the data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$. This gives the relationship

$$L(q, p; \mathcal{D}) = \underbrace{\log p(y_{1:n} \mid \boldsymbol{x}_{1:n})}_{\text{const}} - \mathrm{KL}(q \| p(\cdot \mid \boldsymbol{x}_{1:n}, y_{1:n})). \tag{5.62}$$

Thus, maximizing the ELBO coincides with minimizing reverse-KL.

The ELBO can also be expressed in various other forms:

$$\begin{aligned}
L(q, p; \mathcal{D}) &\doteq \mathbb{E}_{\boldsymbol{\theta} \sim q}[\log p(y_{1:n}, \boldsymbol{\theta} \mid \boldsymbol{x}_{1:n})] + \mathrm{H}[q] && \text{(5.63a)} \\
&= \mathbb{E}_{\boldsymbol{\theta} \sim q}[\log p(y_{1:n}, \boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}) - \log q(\boldsymbol{\theta})] && \text{(5.63b)} \quad \text{using the definition of entropy (5.32)} \\
&= \mathbb{E}_{\boldsymbol{\theta} \sim q}[\log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta})] && \text{(5.63c)} \quad \text{using the product rule (1.14)} \\
&= \mathbb{E}_{\boldsymbol{\theta} \sim q}[\log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta})] - \mathrm{KL}(q \| p(\cdot)). && \text{(5.63d)} \quad \text{using the definition of KL-divergence (5.44)}
\end{aligned}$$

where we denote by $p(\cdot)$ the prior distribution.

Interpreting eq. (5.63d), maximizing the ELBO selects a variational distribution q that is close to the prior distribution $p(\cdot)$ while also maximizing the likelihood of the data $p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta})$. Similarly, interpreting eq. (5.63a), maximizing the ELBO selects q that has large joint likelihood $p(y_{1:n}, \boldsymbol{\theta} \mid \boldsymbol{x}_{1:n})$ (i.e., the data is likely and q is close to the prior) and large entropy $\mathrm{H}[q]$. Why is it reasonable to maximize the entropy of q? Consider two distributions q_1 and q_2 under which the data is "equally" likely and which are "equally" close to the prior. Maximizing the entropy selects the distribution that exhibits the most uncertainty which is in accordance with the maximum entropy principle (cf. remark 2.5).[13]

Recalling that KL-divergence is non-negative, it follows from eq. (5.62) that the evidence lower bound is a (uniform[14]) lower bound to the *evidence* $\log p(y_{1:n} \mid \boldsymbol{x}_{1:n})$:

$$\log p(y_{1:n} \mid \boldsymbol{x}_{1:n}) \geq L(q, p; \mathcal{D}). \tag{5.64}$$

This indicates that maximizing the evidence lower bound is an adequate method of model selection which can be used instead of maximizing the evidence (marginal likelihood) directly (as was discussed in section 4.4.2). Note that this inequality lower bounds the logarithm of an integral by an expectation of a logarithm over some variational distribution q. Hence, the ELBO is a family of lower bounds — one for each variational distribution. Such inequalities are called variational inequalities.

Remark 5.27: Free energy and minimizing surprise

Observe that the evidence can also be interpreted as the negative

[13] The following example illustrates why maximizing entropy is reasonable. Consider a criminal trial with three suspects. The collected evidence shows that suspect 3 can not have committed the crime, however it does not yield any information about suspects 1 and 2. Clearly, any distribution respecting the data must assign zero probability of having committed the crime to suspect 3. However, any distribution interpolating between $(1, 0, 0)$ and $(0, 1, 0)$ respects the data. In this case, the maximum entropy principle suggests that the desired distribution is $(\frac{1}{2}, \frac{1}{2}, 0)$, and indeed, any alternative distribution seems unreasonable.

[14] That is, the bound holds for any variational distribution q (with full support).

surprise about the observations under the prior distribution $p(\boldsymbol{\theta})$,

$$\mathrm{S}[p(y_{1:n} \mid \boldsymbol{x}_{1:n})] \leq \underbrace{\mathbb{E}_{\boldsymbol{\theta}\sim q}[\mathrm{S}[p(y_{1:n}, \boldsymbol{\theta} \mid \boldsymbol{x}_{1:n})]]}_{\text{called } \textit{energy}} - \mathrm{H}[q] \tag{5.65}$$

by negating eq. (5.64)

$$= -L(q, p; \mathcal{D}) \tag{5.66}$$

for any variational distribution q. Here, $-L(q, p; \mathcal{D})$ is also called the *(variational) free energy* with respect to q. Free energy can also be characterized as

$$-L(q, p; \mathcal{D}) = \underbrace{\mathbb{E}_{\boldsymbol{\theta}\sim q}[\mathrm{S}[p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta})]]}_{\text{“inaccuracy”}} + \underbrace{\mathrm{KL}(q \| p(\cdot))}_{\text{“complexity”}} \tag{5.67}$$

the two terms are commonly referred to as “inaccuracy” and “complexity”, mirroring our previous interpretation of eq. (5.63d)

$$= \underbrace{\mathrm{S}[p(y_{1:n} \mid \boldsymbol{x}_{1:n})]}_{\text{“extrinsic” value}} + \underbrace{\mathrm{KL}(q \| p(\cdot \mid \boldsymbol{x}_{1:n}, y_{1:n}))}_{\text{“epistemic” value}}. \tag{5.68}$$

using the derivation of eq. (5.62)

The “extrinsic” value is independent of the variational distribution q and can be though of as a fixed “problem cost”, whereas the “epistemic” value can be interpreted as the approximation error or “solution cost”. In particular, the minimum of free energy across all variational distributions q (when q may be any distribution) coincides with the surprise about the observations almost surely.

Maximizing the ELBO can therefore be interpreted as minimizing a proxy to the surprise about the observations (averaged across all models $\boldsymbol{\theta}$ based on the prior). This is in contrast to maximum likelihood estimation, which picks a *single* model $\boldsymbol{\theta}$ that minimizes the surprise about the observations.

There exists the hypothesis that “physical systems” pursue paths of least surprise, i.e., their dynamics minimize the surprise in their observations.[15] Intuitively, to minimize surprise, there is a natural tradeoff between models $q(\boldsymbol{\theta})$ that “overfit” to observations (e.g., by using a point estimate of $\boldsymbol{\theta}$), and hence, result in a large surprise when new observations deviate from this specific belief, and models $q(\boldsymbol{\theta})$ that “underfit” to observations (e.g., by expecting outcomes with equal probability), and hence, any observation results in a non-negligible surprise.

[15] Refer to the *free energy principle* which was originally introduced by the neuroscientist Karl Friston (Friston, 2010).

As we have alluded to previously when introducing the ELBO, the two terms constituting free energy model this natural tradeoff. Therein, the *entropy* (which is maximized) encourages q to have uncertainty, i.e., to “explore”. In contrast, the *energy* (which is minimized) encourages q to fit the observed data closely, i.e., to “exploit”. This highlights a deep relationship of free energy

to decision-making under uncertainty (which is called *planning*) where the *exploration-exploitation dilemma* arises.[16] For this reason, this idea also plays a central role in approaches to reinforcement learning such as entropy regularization.[17]

[16] We discuss this in great detail in the second half of this course. See section 9.1 for an introduction to the "exploration-exploitation dilemma".

[17] We explore this connection in section 12.5.

Example 5.28: ELBO for Bayesian logistic regression

Recall that Bayesian logistic regression uses the prior distribution $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.[18]

[18] We omit the scaling factor σ_p^2 here for simplicity.

Suppose we use the variational family $\mathcal{Q}$ of all diagonal Gaussians from example 5.7. We have already seen in eq. (5.53) that for a prior $p \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and a variational distribution

$$q_\lambda \sim \mathcal{N}(\boldsymbol{\mu}, \operatorname{diag}_{i \in [d]}\{\sigma_i^2\}),$$

we have

$$\mathrm{KL}(q \| p(\cdot)) = \frac{1}{2} \sum_{i=1}^{d} (\sigma_i^2 + \mu_i^2 - 1 - \log \sigma_i^2).$$

It remains to find the expected likelihood under models from our approximate posterior,

$$\begin{aligned} \mathbb{E}_{\boldsymbol{w} \sim q_\lambda}[\log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{w})] &= \mathbb{E}_{\boldsymbol{w} \sim q_\lambda}\left[\sum_{i=1}^{n} \log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w})\right] && \text{using independence of the data} \\ &= \mathbb{E}_{\boldsymbol{w} \sim q_\lambda}\left[-\sum_{i=1}^{n} \ell_{\log}(\boldsymbol{w}; \boldsymbol{x}_i, y_i)\right] && \text{substituting the logistic loss (5.14)} \end{aligned} \tag{5.69}$$

Exercise 5.29: Gaussian VI vs Laplace approximation

In this exercise, we compare the Laplace approximation from section 5.1 to variational inference with the variational family of Gaussians,

$$\mathcal{Q} \doteq \{q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\Sigma})\}. \tag{5.70}$$

1. Let p be any distribution on $\mathbb{R}$, and let $q^\star = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}(p \| q)$. Show that $q^\star$ differs from the Laplace approximation of p.

Minimizing forward-KL is typically intractable, and we have seen that it is therefore common to minimize the reverse-KL instead:

$$\tilde{q} = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}(q \| p(\cdot \mid \mathcal{D})).$$

2. Show that $\tilde{q} = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ satisfies

$$\begin{aligned} \mathbf{0} &= \mathbb{E}_{\boldsymbol{\theta}\sim\tilde{q}}[\boldsymbol{\nabla}_{\boldsymbol{\theta}} \log p(y_{1:n}, \boldsymbol{\theta} \mid \boldsymbol{x}_{1:n})] \\ \boldsymbol{\Sigma}^{-1} &= -\mathbb{E}_{\boldsymbol{\theta}\sim\tilde{q}}[\boldsymbol{H}_{\boldsymbol{\theta}} \log p(y_{1:n}, \boldsymbol{\theta} \mid \boldsymbol{x}_{1:n})]. \end{aligned} \tag{5.71}$$

Hint 1: For any positive definite and symmetric matrix $\boldsymbol{A}$, it holds that $\boldsymbol{\nabla}_{\boldsymbol{A}} \log\det(\boldsymbol{A}) = \boldsymbol{A}^{-1}$.
Hint 2: For any function f and Gaussian $p = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$,

$$\begin{aligned} \boldsymbol{\nabla}_{\boldsymbol{\mu}} \mathbb{E}_{\boldsymbol{x}\sim p}[f(\boldsymbol{x})] &= \mathbb{E}_{\boldsymbol{x}\sim p}[\boldsymbol{\nabla}_{\boldsymbol{x}} f(\boldsymbol{x})] \\ \boldsymbol{\nabla}_{\boldsymbol{\Sigma}} \mathbb{E}_{\boldsymbol{x}\sim p}[f(\boldsymbol{x})] &= \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p}[\boldsymbol{H}_{\boldsymbol{x}} f(\boldsymbol{x})]. \end{aligned} \tag{5.72}$$

Recall the Laplace approximation of $p(\boldsymbol{\theta} \mid \mathcal{D}) \propto p(y_{1:n}, \boldsymbol{\theta} \mid \boldsymbol{x}_{1:n})$,

$$\begin{aligned} \mathbf{0} &= \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log p(y_{1:n}, \boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}) \\ \boldsymbol{\Sigma}^{-1} &= -\boldsymbol{H}_{\boldsymbol{\theta}} \log p(y_{1:n}, \boldsymbol{\theta} \mid \boldsymbol{x}_{1:n})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \end{aligned} \tag{5.73}$$

where $\hat{\boldsymbol{\theta}}$ is the MAP estimate. Crucially, the Laplace approximation is fitted *locally* at $\hat{\boldsymbol{\theta}}$. Comparing eq. (5.73) to eq. (5.71), we see that Gaussian variational inference satisfies the conditions of the Laplace approximation *on average*. For more details, refer to "The variational Gaussian approximation revisited" (Opper and Archambeau, 2009).

▷ *Solution*

5.5.1 Gradient of Evidence Lower Bound

The next problem is, of course, solving the optimization problem of maximizing the ELBO. In stochastic gradient descent, we already know a tool which we can use. However, SGD requires unbiased gradient estimates of the loss, which is given as $\ell(\boldsymbol{\lambda}; \mathcal{D}) \doteq -L(q_{\boldsymbol{\lambda}}, p; \mathcal{D})$. Thus, we need to obtain gradient estimates of

$$\boldsymbol{\nabla}_{\boldsymbol{\lambda}} L(q_{\boldsymbol{\lambda}}, p; \mathcal{D}) = \boldsymbol{\nabla}_{\boldsymbol{\lambda}} \mathbb{E}_{\boldsymbol{\theta}\sim q_{\boldsymbol{\lambda}}}[\log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta})] - \boldsymbol{\nabla}_{\boldsymbol{\lambda}} \mathrm{KL}(q_{\boldsymbol{\lambda}} \| p(\cdot)). \tag{5.74}$$

using the definition of the ELBO (5.63d)

Typically, the KL-divergence (and its gradient) can be computed exactly for commonly used variational families. For example, we have already seen a closed-form expression of the KL-divergence for Gaussians in eq. (5.50).

Exercise 5.30: Gradient of reverse-KL

Suppose $p \doteq \mathcal{N}(\mathbf{0}, \sigma_p^2 \boldsymbol{I})$ and a tractable distribution described by

$$q_{\boldsymbol{\lambda}} \doteq \mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}\{\sigma_1^2, \ldots, \sigma_d^2\})$$

where $\boldsymbol{\mu} \doteq [\mu_1 \ \cdots \ \mu_d]$ and $\boldsymbol{\lambda} \doteq [\mu_1 \ \cdots \ \mu_d \ \sigma_1 \ \cdots \ \sigma_d]$. Show that the gradient of $\mathrm{KL}(q_{\boldsymbol{\lambda}} \| p(\cdot))$ with respect to $\boldsymbol{\lambda}$ is given by

$$\nabla_{\boldsymbol{\mu}} \mathrm{KL}(q_{\boldsymbol{\lambda}} \| p(\cdot)) = \sigma_p^{-2} \boldsymbol{\mu}, \quad \text{and} \tag{5.75a}$$

$$\nabla_{[\sigma_1 \ \cdots \ \sigma_d]} \mathrm{KL}(q_{\boldsymbol{\lambda}} \| p(\cdot)) = \begin{bmatrix} \frac{\sigma_1}{\sigma_p^2} - \frac{1}{\sigma_1} & \cdots & \frac{\sigma_d}{\sigma_p^2} - \frac{1}{\sigma_d} \end{bmatrix}. \tag{5.75b}$$

▷ *Solution*

Obtaining the gradient of the evidence lower bound is more difficult. This is because the expectation integrates over the measure $q_{\boldsymbol{\lambda}}$, which depends on the variational parameters $\boldsymbol{\lambda}$. Thus, we cannot move the gradient operator inside the expectation as we have done previously in remark 1.22. There are two main techniques which are used to rewrite the gradient in such a way that Monte Carlo sampling becomes possible.

One approach is to use *score gradients* via the "score function trick":

$$\begin{aligned} &\nabla_{\boldsymbol{\lambda}} \mathbb{E}_{\boldsymbol{\theta} \sim q_{\boldsymbol{\lambda}}} [\log p(\boldsymbol{y}_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta})] \\ &\quad = \mathbb{E}_{\boldsymbol{\theta} \sim q_{\boldsymbol{\lambda}}} [\log p(\boldsymbol{y}_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta}) \underbrace{\nabla_{\boldsymbol{\lambda}} \log q_{\boldsymbol{\lambda}}(\boldsymbol{\theta})}_{\text{score function}}], \end{aligned} \tag{5.76}$$

which we introduce in section 12.3.2 in the context of reinforcement learning. More common in the context of variational inference is the so-called "reparameterization trick".

Theorem 5.31 (Reparameterization trick)**.** *Given a random variable $\boldsymbol{\epsilon} \sim \phi$ (which is independent of $\boldsymbol{\lambda}$) and given a differentiable and invertible function $\boldsymbol{g} : \mathbb{R}^d \to \mathbb{R}^d$. We let $\boldsymbol{\theta} \doteq \boldsymbol{g}(\boldsymbol{\epsilon}; \boldsymbol{\lambda})$. Then,*

$$q_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) = \phi(\boldsymbol{\epsilon}) \cdot \left| \det(\boldsymbol{D}_{\boldsymbol{\epsilon}} \boldsymbol{g}(\boldsymbol{\epsilon}; \boldsymbol{\lambda})) \right|^{-1}, \tag{5.77}$$

$$\mathbb{E}_{\boldsymbol{\theta} \sim q_{\boldsymbol{\lambda}}} [\boldsymbol{f}(\boldsymbol{\theta})] = \mathbb{E}_{\boldsymbol{\epsilon} \sim \phi} [\boldsymbol{f}(\boldsymbol{g}(\boldsymbol{\epsilon}; \boldsymbol{\lambda}))] \tag{5.78}$$

for a "nice" function $\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^e$.

Proof. By the change of variables formula (1.52) and using $\boldsymbol{\epsilon} = \boldsymbol{g}^{-1}(\boldsymbol{\theta}; \boldsymbol{\lambda})$,

$$\begin{aligned} q_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) &= \phi(\boldsymbol{\epsilon}) \cdot \left| \det\Big(\boldsymbol{D}_{\boldsymbol{\theta}} \boldsymbol{g}^{-1}(\boldsymbol{\theta}; \boldsymbol{\lambda}) \Big) \right| \\ &= \phi(\boldsymbol{\epsilon}) \cdot \left| \det\Big((\boldsymbol{D}_{\boldsymbol{\epsilon}} \boldsymbol{g}(\boldsymbol{\epsilon}; \boldsymbol{\lambda}))^{-1} \Big) \right| && \text{by the inverse function theorem, } \boldsymbol{D}\boldsymbol{g}^{-1}(\boldsymbol{y}) = \boldsymbol{D}\boldsymbol{g}(\boldsymbol{x})^{-1} \\ &= \phi(\boldsymbol{\epsilon}) \cdot \left| \det(\boldsymbol{D}_{\boldsymbol{\epsilon}} \boldsymbol{g}(\boldsymbol{\epsilon}; \boldsymbol{\lambda})) \right|^{-1}. && \text{using } \det(\boldsymbol{A}^{-1}) = \det(\boldsymbol{A})^{-1} \end{aligned}$$

Equation (5.78) is a direct consequence of the law of the unconscious statistician (1.30). □

In other words, the reparameterization trick requires finding a function $g(\cdot;\lambda)$ and density ϕ such that $q_\lambda = g(\cdot;\lambda)_\sharp \phi$ is the pushforward of ϕ under perturbation g.

Applying the reparameterization trick, we can use our analysis of eq. (1.28) to swap the order of gradient and expectation,

$$\nabla_\lambda \mathbb{E}_{\theta\sim q_\lambda}[f(\theta)] = \mathbb{E}_{\epsilon\sim\phi}[\nabla_\lambda f(g(\epsilon;\lambda))]. \tag{5.79}$$

If we can find a function g such that $\theta \sim q_\lambda$ for some sampling distribution ϕ independent of λ, we can use the reparameterization trick to simplify our gradient. We call a distribution q_λ which admits reparameterization *reparameterizable*.

Example 5.32: Reparameterization trick for Gaussians

Suppose we use a Gaussian variational approximation,

$$q_\lambda(\theta) \doteq \mathcal{N}(\theta;\mu,\Sigma),$$

where we assume Σ to have full rank (i.e., be invertible). We have seen in eq. (1.123) that a Gaussian random vector $\epsilon \sim \mathcal{N}(0, I)$ following a standard normal distribution can be transformed to follow the Gaussian distribution q_λ by using the linear transformation,

$$\theta = g(\epsilon;\lambda) \doteq \Sigma^{1/2}\epsilon + \mu. \tag{5.80}$$

In particular, we have

$$\epsilon = g^{-1}(\theta;\lambda) = \Sigma^{-1/2}(\theta-\mu) \quad \text{and} \tag{5.81}$$

by solving eq. (5.80) for ϵ

$$\phi(\epsilon) = q_\lambda(\theta)\cdot\left|\det\left(\Sigma^{1/2}\right)\right|. \tag{5.82}$$

using the reparameterization trick (i.e., the change of variables formula) (5.77)

Exercise 5.33: Reparameterizable distributions

1. Let $X \sim \text{Unif}([a,b])$ for any $a \le b$. That is,

$$p_X(x) = \begin{cases} \frac{1}{b-a} & \text{if } x \in [a,b] \\ 0 & \text{otherwise.} \end{cases} \tag{5.83}$$

   Show that X can be reparameterized in terms of $\text{Unif}([0,1])$.
   Hint: You may use that for any $Y \sim \text{Unif}([a,b])$ and $c \in \mathbb{R}$,
   - $Y + c \sim \text{Unif}([a+c, b+c])$ *and*
   - $cY \sim \text{Unif}([c\cdot a, c\cdot b])$.
2. Let $Z \sim \mathcal{N}(\mu,\sigma^2)$ and $X \doteq e^Z$. That is, X is logarithmically normal distributed with parameters μ and σ^2. Show that X

can be reparameterized in terms of $\mathcal{N}(0, 1)$.

3. Let $X \sim \text{Cauchy}(m, \tau)$ be a random variable that follows a *Cauchy distribution* with location m and scale τ. The CDF of a Cauchy distribution is defined as

$$P_X(x) \doteq \frac{1}{\pi} \arctan\left(\frac{x - m}{\tau}\right) + \frac{1}{2}. \tag{5.84}$$

Show that $\text{Cauchy}(0, 1)$ can be reparameterized in terms of $\text{Unif}([0, 1])$.

Finally, let us apply the reparameterization trick to compute the gradient of an expectation.

4. Let $\text{ReLU}(z) \doteq \max\{0, z\}$ and $w > 0$. Show that

$$\frac{d}{d\mu} \mathbb{E}_{x \sim \mathcal{N}(\mu, 1)} \text{ReLU}(wx) = w\Phi(\mu)$$

where Φ denotes the CDF of the standard normal distribution.

▷ *Solution*

In the following, we write $\boldsymbol{C} \doteq \boldsymbol{\Sigma}^{1/2}$. Let us now derive the gradient estimate for the evidence lower bound assuming the Gaussian variational approximation from example 5.32. This approach extends to any reparameterizable distribution.

$$\begin{aligned}
&\nabla_\lambda \, \mathbb{E}_{\boldsymbol{\theta} \sim q_\lambda}[\log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta})] \\
&= \nabla_{\boldsymbol{C}, \boldsymbol{\mu}} \, \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})} \Big[\log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta})|_{\boldsymbol{\theta} = \boldsymbol{C}\boldsymbol{\epsilon} + \boldsymbol{\mu}} \Big] && (5.85) && \text{using the reparameterization trick (5.78)} \\
&= n \cdot \nabla_{\boldsymbol{C}, \boldsymbol{\mu}} \, \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})} \left[\frac{1}{n} \sum_{i=1}^{n} \log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta})|_{\boldsymbol{\theta} = \boldsymbol{C}\boldsymbol{\epsilon} + \boldsymbol{\mu}} \right] && && \text{using independence of the data and extending with } {}^{n}/_{n} \\
&= n \cdot \nabla_{\boldsymbol{C}, \boldsymbol{\mu}} \, \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})} \mathbb{E}_{i \sim \text{Unif}([n])} \Big[\log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta})|_{\boldsymbol{\theta} = \boldsymbol{C}\boldsymbol{\epsilon} + \boldsymbol{\mu}} \Big] && && \text{interpreting the sum as an expectation} \\
&= n \cdot \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})} \mathbb{E}_{i \sim \text{Unif}([n])} \Big[\nabla_{\boldsymbol{C}, \boldsymbol{\mu}} \, \log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta})|_{\boldsymbol{\theta} = \boldsymbol{C}\boldsymbol{\epsilon} + \boldsymbol{\mu}} \Big] && (5.86) && \text{using eq. (1.28)} \\
&\approx n \cdot \frac{1}{m} \sum_{j=1}^{m} \nabla_{\boldsymbol{C}, \boldsymbol{\mu}} \, \log p(y_{i^{(j)}} \mid x_{i^{(j)}}, \boldsymbol{\theta})\big|_{\boldsymbol{\theta} = \boldsymbol{C}\boldsymbol{\epsilon}^{(j)} + \boldsymbol{\mu}} && (5.87) && \text{using Monte Carlo sampling}
\end{aligned}$$

where $\boldsymbol{\epsilon}^{(j)} \overset{\text{iid}}{\sim} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $i^{(j)} \overset{\text{iid}}{\sim} \text{Unif}([n])$. This yields an unbiased gradient estimate, which we can use with stochastic gradient descent to maximize the evidence lower bound. We have successfully recast the difficult problems of learning and inference as an optimization problem!

The procedure of approximating the true posterior using a variational posterior by maximizing the evidence lower bound using stochastic optimization is also called *black box stochastic variational inference*. The only requirement is that we can obtain unbiased gradient estimates

from the evidence lower bound (and the likelihood). If we use the variational family of diagonal Gaussians, we only require twice as many parameters as other inference techniques like MAP estimation. The performance can be improved by using natural gradients and variance reduction techniques for the gradient estimates such as control variates.

Optional Readings

- Ranganath, Gerrish, and Blei (2014).
  Black box variational inference.
- Duvenaud and Adams (2015).
  Black-box stochastic variational inference in five lines of python.
- Titsias and Lázaro-Gredilla (2014).
  Doubly stochastic variational Bayes for non-conjugate inference.
- Hensman, Matthews, and Ghahramani (2015).
  Scalable variational Gaussian process classification.
- Mohamed, Rosca, Figurnov, and Mnih (2020).
  Monte Carlo Gradient Estimation in Machine Learning.

6

Markov Chain Monte Carlo Methods

Variational inference approximates the entire posterior distribution. However, note that the key challenge in Bayesian inference is not learning the posterior distribution, but using the posterior distribution for predictions,

$$p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, y_{1:n}) = \int p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}, y_{1:n}) \, d\boldsymbol{\theta}. \tag{6.1}$$

This integral can be interpreted as an expectation over the posterior distribution,

$$= \mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot \mid \boldsymbol{x}_{1:n}, y_{1:n})}[p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta})]. \tag{6.2}$$

Observe that the likelihood $f(\boldsymbol{\theta}) \doteq p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta})$ is easy to evaluate. The difficulty lies in sampling from the posterior distribution. Assuming we can obtain independent samples from the posterior distribution, we can use Monte Carlo sampling to obtain an unbiased estimate of the expectation,

$$\approx \frac{1}{m} \sum_{i=1}^{m} f(\boldsymbol{\theta}^{(i)}) \tag{6.3}$$

for independent $\boldsymbol{\theta}^{(i)} \overset{\text{iid}}{\sim} p(\cdot \mid \boldsymbol{x}_{1:n}, y_{1:n})$. The law of large numbers (1.84) and Hoeffding's inequality (1.88) imply that this estimator is consistent and sharply concentrated.[1]

[1] For more details, refer to section 1.4.2.

Obtaining samples of the posterior distribution is therefore sufficient to perform approximate inference. Recall that the difficulty of computing the posterior p exactly, was in finding the normalizing constant Z,

$$p(x) = \frac{1}{Z} q(x). \tag{6.4}$$

The joint likelihood q is typically easy to obtain. Note that $q(x)$ is proportional to the probability density associated with x, but q does not

integrate to 1. Such functions are also called a *finite measure*. Without normalizing q, we cannot directly sample from it.

Even if we were able to approximate Z, this generally does not yield an efficient sampling method. For example, the inverse transform sampling discussed in section 1.1.5 requires an (approximate) quantile function. Computing the quantile function given an arbitrary PDF requires solving integrals over the domain of the PDF which is what we were trying to avoid in the first place.

The key idea of Markov chain Monte Carlo methods is to construct a Markov chain, which is efficient to simulate and has the stationary distribution p.

6.1 Markov Chains

To begin with, let us revisit the fundamental theory behind Markov chains.

Definition 6.1 (Markov chain)**.** A *(finite and discrete-time) Markov chain* over the state space

$$S \doteq \{0, \dots, n-1\} \tag{6.5}$$

is a stochastic process[2] $(X_t)_{t \in \mathbb{N}_0}$ valued in S such that the *Markov property*,

$$X_{t+1} \perp X_{0:t-1} \mid X_t, \tag{6.6}$$

is satisfied.

[2] A *stochastic process* is a sequence of random variables.

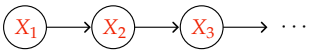


Figure 6.1: Directed graphical model of a Markov chain. The random variable X_{t+1} is conditionally independent of the random variables $X_{0:t-1}$ given X_t.

Intuitively, the Markov property states that future behavior is independent of past states given the present state.

Remark 6.2: Generalizations of Markov chains
One can also define continuous-state Markov chains (for example, where states are vectors in $\mathbb{R}^d$) and the results which we state for (finite) Markov chains will generally carry over. For a survey, refer to "General state space Markov chains and MCMC algorithms" (Roberts and Rosenthal, 2004).

Moreover, one can also consider continuous-time Markov chains. One example of such a continuous-space and continuous-time Markov chain is the *Wiener process* (cf. remark 6.29).

We restrict our attention to *time-homogeneous* Markov chains,[3] which can be characterized by a *transition function*,

$$p(x' \mid x) \doteq \mathbb{P}\big(X_{t+1} = x' \mid X_t = x\big). \tag{6.7}$$

[3] That is, the transition probabilities do not change over time.

As the state space is finite, we can describe the transition function by the *transition matrix*,

$$\boldsymbol{P} \doteq \begin{bmatrix} p(x_1 \mid x_1) & \cdots & p(x_n \mid x_1) \\ \vdots & \ddots & \vdots \\ p(x_1 \mid x_n) & \cdots & p(x_n \mid x_n) \end{bmatrix} \in \mathbb{R}^{n \times n}. \tag{6.8}$$

Note that each row of $\boldsymbol{P}$ must always sum to 1. Such matrices are also called *stochastic*.

The *transition graph* of a Markov chain is a directed graph consisting of vertices S and weighted edges represented by the adjacency matrix $\boldsymbol{P}$.

The current state of the Markov chain at time t is denoted by the probability distribution q_t over states S, that is, $X_t \sim q_t$. In the finite setting, q_t is a PMF, which is often written explicitly as the row vector $\boldsymbol{q}_t \in \mathbb{R}^{1 \times |S|}$. The initial state (or prior) of the Markov chain is given as $X_0 \sim q_0$.

Exercise 6.3: Markov chain update

Prove that one iteration of the Markov chain can be expressed as

$$\boldsymbol{q}_{t+1} = \boldsymbol{q}_t \boldsymbol{P}. \tag{6.9}$$

▷ *Solution*

It is implied directly that we can write the state of the Markov chain at time $t + k$ as

$$\boldsymbol{q}_{t+k} = \boldsymbol{q}_t \boldsymbol{P}^k. \tag{6.10}$$

Exercise 6.4: k-step transitions

Prove that the entry $\boldsymbol{P}^k(x, x')$ corresponds to the probability of transitioning from state $x \in S$ to state $x' \in S$ in exactly k steps. We denote this entry by $p^{(k)}(x' \mid x)$.

▷ *Solution*

In the analysis of Markov chains, there are two main concepts of interest: stationarity and convergence. We begin by introducing stationarity.

6.1.1 Stationarity

Definition 6.5 (Stationary distribution)**.** A distribution π is *stationary* with respect to the transition function p iff

$$\pi(x) = \sum_{x' \in S} p(x \mid x')\pi(x') \tag{6.11}$$

holds for all $x \in S$. It follows from eq. (6.9) that equivalently, π is stationary w.r.t. a transition matrix $\boldsymbol{P}$ iff

$$\boldsymbol{\pi} = \boldsymbol{\pi}\boldsymbol{P}. \tag{6.12}$$

After entering a stationary distribution π, a Markov chain will always remain in the stationary distribution. In particular, suppose that X_t is distributed according to π, then for all $k \geq 0$, $X_{t+k} \sim \pi$.

Remark 6.6: When does a stationary distribution exist?

In general, there are Markov chains with infinitely many stationary distributions or no stationary distribution at all. You can find some examples in fig. 6.2.

It can be shown that there exists a unique stationary distribution π if the Markov chain is *irreducible*, that is, if every state is reachable from every other state with a positive probability when the Markov chain is run for enough steps. Formally,

$$\forall x, x' \in S.\, \exists k \in \mathbb{N}.\, p^{(k)}(x' \mid x) > 0. \tag{6.13}$$

Equivalently, a Markov chain is irreducible iff its transition graph is strongly connected.

6.1.2 Convergence

Let us now consider Markov chains with a unique stationary distribution.[4] A natural next question is whether this Markov chain converges to its stationary distribution. We say that a Markov chain converges to its stationary distribution iff we have

$$\lim_{t \to \infty} q_t = \pi, \tag{6.14}$$

irrespectively of the initial distribution q_0.

[4] Observe that the stationary distribution of an irreducible Markov chain must have full support, that is, assign positive probability to every state.

Remark 6.7: When does a Markov chain converge?

Even if a Markov chain has a unique stationary distribution, it must not converge to it. Consider example (3) in fig. 6.2. Clearly,

$\pi = (\frac{1}{2}, \frac{1}{2})$ is the unique stationary distribution. However, observe that if we start with a suitable initial distribution such as $q_0 = (1, 0)$, at no point in time will the probability of all states be positive, and in particular, the chain will not converge to π. Instead, the chain behaves periodically, i.e., its state distributions are $q_t = (0, 1)$ and $q_{2t} = (1, 0)$ for all $t \in \mathbb{N}$. It turns out that if we exclude such "periodic" Markov chains, then the remaining (irreducible) Markov chains will always converge to their stationary distribution.

Formally, a Markov chain is *aperiodic* if for all states $x \in S$,

$$\exists k_0 \in \mathbb{N}. \forall k \geq k_0. p^{(k)}(x \mid x) > 0. \tag{6.15}$$

In words, a Markov chain is aperiodic iff for every state x, the transition graph has a closed path from x to x with length k for all $k \in \mathbb{N}$ greater than some $k_0 \in \mathbb{N}$.

This additional property leads to the concept of ergodicity.

Definition 6.8 (Ergodicity)**.** A Markov chain is *ergodic* iff there exists a $t \in \mathbb{N}_0$ such that for any $x, x' \in S$ we have

$$p^{(t)}(x' \mid x) > 0, \tag{6.16}$$

whereby $p^{(t)}(x' \mid x)$ is the probability to reach x' from x in exactly t steps. Equivalent conditions are

1. that there exists some $t \in \mathbb{N}_0$ such that all entries of $\boldsymbol{P}^t$ are strictly positive; and
2. that it is irreducible and aperiodic.

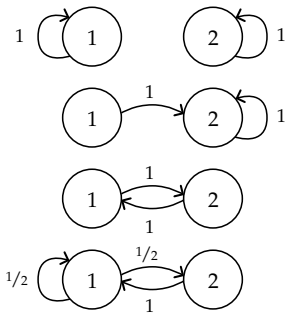


Figure 6.2: Transition graphs of Markov chains: (1) is not ergodic as its transition diagram is not strongly connected; (2) is not ergodic for the same reason; (3) is irreducible but periodic and therefore not ergodic; (4) is ergodic with stationary distribution $\pi(1) = 2/3, \pi(2) = 1/3$.

Example 6.9: Making a Markov chain ergodic

A commonly used strategy to ensure that a Markov chain is ergodic is to add "self-loops" to every vertex in the transition graph. That is, to ensure that at any point in time, the Markov chain remains with positive probability in its current state.

Take a (not necessarily ergodic) but irreducible Markov chain with transition matrix $\boldsymbol{P}$. We define the new Markov chain

$$\boldsymbol{P}' \doteq \frac{1}{2}\boldsymbol{P} + \frac{1}{2}\boldsymbol{I}. \tag{6.17}$$

It is a simple exercise to confirm that $\boldsymbol{P}'$ is stochastic, and hence a valid transition matrix. Also, it follows directly that $\boldsymbol{P}'$ is irreducible (as $\boldsymbol{P}$ is irreducible) and aperiodic as every vertex has a closed path of length 1 to itself, and therefore the chain is ergodic.

Take now $\boldsymbol{\pi}$ to be a stationary distribution of $\boldsymbol{P}$. We have that $\boldsymbol{\pi}$ is also a stationary distribution of $\boldsymbol{P}'$ as

$$\boldsymbol{\pi}\boldsymbol{P}' = \frac{1}{2}\boldsymbol{\pi}\boldsymbol{P} + \frac{1}{2}\boldsymbol{\pi}\boldsymbol{I} = \frac{1}{2}\boldsymbol{\pi} + \frac{1}{2}\boldsymbol{\pi} = \boldsymbol{\pi}. \tag{6.18}$$

using (6.12)

Fact 6.10 (Fundamental theorem of ergodic Markov chains)**.** *An ergodic Markov chain has a unique stationary distribution π (with full support) and*

$$\lim_{t\to\infty} q_t = \pi \tag{6.19}$$

irrespectively of the initial distribution q_0.

Proof. Refer to theorem 4.9 of "Markov chains and mixing times" (Levin and Peres, 2017) for a proof. □

This naturally suggests constructing an ergodic Markov chain such that its stationary distribution coincides with the posterior distribution. If we then sample "sufficiently long", X_t is drawn from a distribution that is "very close" to the posterior distribution.

Remark 6.11: How quickly does a Markov chain converge?

The convergence speed of Markov chains is a rich field of research. "Sufficiently long" and "very close" are commonly made precise by the notions of *rapidly mixing* Markov chains and *total variation distance*.

Definition 6.12 (Total variation distance)**.** The total variation distance between two probability distributions μ and ν on $\mathcal{A}$ is defined by

$$\|\mu - \nu\|_{\mathrm{TV}} \doteq 2 \sup_{A\subseteq\mathcal{A}} |\mu(A) - \nu(A)|\,. \tag{6.20}$$

It defines the distance between μ and ν to be the maximum difference between the probabilities that μ and ν assign to the same event.

As opposed to the KL-divergence (5.44), the total variation distance is a metric. In particular, it is symmetric and satisfies the triangle inequality. It can be shown that

$$\|\mu - \nu\|_{\mathrm{TV}} \leq \sqrt{2\mathrm{KL}(\mu\|\nu)} \tag{6.21}$$

which is known as *Pinsker's inequality*. Moreover, if μ and ν are discrete distributions over the set S, it can be shown that

$$\|\mu - \nu\|_{\mathrm{TV}} = \sum_{i\in S} |\mu(i) - \nu(i)|\,. \tag{6.22}$$

Definition 6.13 (Mixing time)**.** For a Markov chain with stationary distribution π, its *mixing time* with respect to the total variation distance is

$$\tau_{\mathrm{TV}}(\epsilon) \doteq \min\{t \mid \forall q_0 : \|q_t - \pi\|_{\mathrm{TV}} \leq \epsilon\}. \tag{6.23}$$

Thus, the mixing time measures the time required by a Markov chain for the distance to stationarity to be small. A Markov chain is typically said to be *rapidly mixing* if

$$\tau_{\mathrm{TV}}(\epsilon) \in \mathcal{O}(\mathrm{poly}(n, \log(1/\epsilon))). \tag{6.24}$$

That is, a rapidly mixing Markov chain on n states needs to be simulated for at most $\mathrm{poly}(n)$ steps to obtain a "good" sample from its stationary distribution π.

You can find a thorough introduction to mixing times in chapter 4 of "Markov chains and mixing times" (Levin and Peres, 2017). Later chapters introduce methods for showing that a Markov chain is rapidly mixing.

Exercise 6.14: Finding stationary distributions

A news station classifies each day as "good", "fair", or "poor" based on its daily ratings which fluctuate with what is occurring in the news. Moreover, the following table shows the probabilistic relationship between the type of the current day and the probability of the type of the next day conditioned on the type of the current day.

		next day		
		good	fair	poor
current day	good	0.60	0.30	0.10
	fair	0.50	0.25	0.25
	poor	0.20	0.40	0.40

In the long run, what percentage of news days will be classified as "good"?

▷ *Solution*

Readings

For an in-depth treatment of Markov chains refer to chapter 1 of "Markov chains and mixing times" (Levin and Peres, 2017).

6.1.3 Detailed Balance Equation

How can we confirm that the stationary distribution of a Markov chain coincides with the posterior distribution? The detailed balance equation yields a very simple method.

Definition 6.15 (Detailed balance equation / reversibility)**.** A Markov chain satisfies the *detailed balance equation* with respect to a distribution π iff

$$\pi(x)p(x' \mid x) = \pi(x')p(x \mid x') \tag{6.25}$$

holds for any $x, x' \in S$. A Markov chain that satisfies the detailed balance equation with respect to π is called *reversible* with respect to π.

Lemma 6.16. *Given a finite Markov chain, if the Markov chain is reversible with respect to π then π is a stationary distribution.*[5]

[5] Note that reversibility of π is only a sufficient condition for stationarity of π, it is not necessary! In particular, there are irreversible ergodic Markov chains.

Proof. Let $\pi \doteq q_t$. We have,

$$\begin{aligned}
q_{t+1}(x) &= \sum_{x' \in S} p(x \mid x') q_t(x') && \text{using the Markov property (6.6)} \\
&= \sum_{x' \in S} p(x \mid x') \pi(x') \\
&= \sum_{x' \in S} p(x' \mid x) \pi(x) && \text{using the detailed balance equation (6.25)} \\
&= \pi(x) \sum_{x' \in S} p(x' \mid x) \\
&= \pi(x). && \text{using that } \textstyle\sum_{x' \in S} p(x' \mid x) = 1
\end{aligned}$$

□

That is, if we can show that the detailed balance equation (6.25) holds for some distribution q, then we know that q is the stationary distribution of the Markov chain.

Next, reconsider our posterior distribution $p(x) = \frac{1}{Z} q(x)$ from eq. (6.4). If we substitute the posterior for π in the detailed balance equation, we obtain

$$\frac{1}{Z} q(x) p(x' \mid x) = \frac{1}{Z} q(x') p(x \mid x'), \tag{6.26}$$

or equivalently,

$$q(x)p(x' \mid x) = q(x')p(x \mid x'). \tag{6.27}$$

In words, we do not need to know the true posterior p to check that the stationary distribution of our Markov chain coincides with p, it suffices to know the finite measure q!

6.1.4 *Ergodic Theorem*

If we now suppose that we can construct a Markov chain whose stationary distribution coincides with the posterior distribution — we will see later that this is possible — it is not apparent that this allows us to estimate expectations over the posterior distribution. Note that although constructing such a Markov chain allows us to obtain samples from the posterior distribution, they are *not* independent. In fact, due to the structure of a Markov chain, by design, they are strongly dependent. Thus, the law of large numbers and Hoeffding's inequality do not apply. By itself, it is not even clear that an estimator relying on samples from a single Markov chain will be unbiased.

Theoretically, we could simulate many Markov chains separately and obtain one sample from each of them. This, however, is extremely inefficient.

It turns out that there is a way to generalize the (strong) law of large numbers to Markov chains.

Theorem 6.17 (Ergodic theorem)**.** *Given an ergodic Markov chain* $(X_t)_{t\in\mathbb{N}_0}$ *over a finite state space* S *with stationary distribution* π *and a function* $f : S \to \mathbb{R}$*,*

$$\frac{1}{n}\sum_{i=1}^{n} f(x_i) \overset{\text{a.s.}}{\to} \sum_{x\in S} \pi(x) f(x) = \mathbb{E}_{x\sim\pi}[f(x)] \tag{6.28}$$

as $n \to \infty$ *where* $x_i \sim X_i \mid x_{i-1}$*.*

Proof. See appendix C of "Markov chains and mixing times" (Levin and Peres, 2017) for a proof. □

This result is the fundamental reason for why Markov chain Monte Carlo methods are possible. There are analogous results for continuous domains.

Note, however, that the ergodic theorem only tells us that simulating a single Markov chain yields an unbiased estimator. It does not tell us anything about the rate of convergence and variance of such an estimator. The convergence rate depends on the mixing time of the Markov chain, which is difficult to establish in general.

In practice, one observes that Markov chain Monte Carlo methods have a so-called *"burn-in" time* during which the distribution of the Markov chain does not yet approximate the posterior distribution well. Typically, the first t_0 samples are therefore discarded,

$$\mathbb{E}[f(X)] \approx \frac{1}{T - t_0} \sum_{t=t_0+1}^{T} f(X_t). \tag{6.29}$$

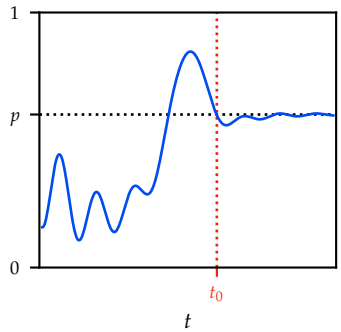


Figure 6.3: Illustration of the "burn-in" time t_0 of a Markov chain approximating the posterior $p(y^\star = 1 \mid \boldsymbol{X}, \boldsymbol{y})$ of Bayesian logistic regression. The true posterior p is shown in gray. The distribution of the Markov chain at time t is shown in red.

It is not clear in general how T and t_0 should be chosen such that the estimator is unbiased, rather they have to be tuned.

Another widely used heuristic is to first find the mode of the posterior distribution and then start the Markov chain at that point. This tends to increase the rate of convergence drastically, as the Markov chain does not have to "walk to the location in the state space where most probability mass will be located".

6.2 *Elementary Sampling Methods*

We will now examine methods for constructing and sampling from a Markov chain with the goal of approximating samples from the posterior distribution p. Note that in this setting the state space of the Markov chain is $\mathbb{R}^n$ and a single state at time t is described by the random vector $\mathbf{X} \doteq [X_1, \dots, X_n]$.

6.2.1 *Metropolis-Hastings Algorithm*

Suppose we are given a *proposal distribution* $r(\boldsymbol{x}' \mid \boldsymbol{x})$ which, given we are in state $\boldsymbol{x}$, proposes a new state $\boldsymbol{x}'$. Metropolis and Hastings showed that using the *acceptance distribution* $\text{Bern}(\alpha(\boldsymbol{x}' \mid \boldsymbol{x}))$ where

$$\alpha(\boldsymbol{x}' \mid \boldsymbol{x}) \doteq \min\left\{1, \frac{p(\boldsymbol{x}')r(\boldsymbol{x} \mid \boldsymbol{x}')}{p(\boldsymbol{x})r(\boldsymbol{x}' \mid \boldsymbol{x})}\right\} \tag{6.30}$$

$$= \min\left\{1, \frac{q(\boldsymbol{x}')r(\boldsymbol{x} \mid \boldsymbol{x}')}{q(\boldsymbol{x})r(\boldsymbol{x}' \mid \boldsymbol{x})}\right\} \tag{6.31}$$

similarly to the detailed balance equation, the normalizing constant Z cancels

to decide whether to follow the proposal yields a Markov chain with stationary distribution $p(\boldsymbol{x}) = \frac{1}{Z}q(\boldsymbol{x})$.

Algorithm 6.18: Metropolis-Hastings algorithm

```
initialize x ∈ ℝ^n
for t = 1 to T do
    sample x' ~ r(x' | x)
    sample u ~ Unif([0, 1])
    if u ≤ α(x' | x) then update x ← x'
    else update x ← x
```

Theorem 6.19 (Metropolis-Hastings theorem)**.** *Given an arbitrary proposal distribution r, the stationary distribution of the Markov chain simulated by the Metropolis-Hastings algorithm is $p(\boldsymbol{x}) = \frac{1}{Z}q(\boldsymbol{x})$.*

Proof. First, let us define the transition probabilities of the Markov chain. The probability of transitioning from a state $\boldsymbol{x}$ to a state $\boldsymbol{x}'$ is given by $r(\boldsymbol{x}' \mid \boldsymbol{x})\alpha(\boldsymbol{x}' \mid \boldsymbol{x})$ if $\boldsymbol{x} \neq \boldsymbol{x}'$ and the probability of proposing to remain in state $\boldsymbol{x}$, $r(\boldsymbol{x} \mid \boldsymbol{x})$, plus the probability of denying the proposal, otherwise.

$$p(\boldsymbol{x}' \mid \boldsymbol{x}) = \begin{cases} r(\boldsymbol{x}' \mid \boldsymbol{x})\alpha(\boldsymbol{x}' \mid \boldsymbol{x}) & \text{if } \boldsymbol{x} \neq \boldsymbol{x}' \\ r(\boldsymbol{x} \mid \boldsymbol{x}) + \sum_{\boldsymbol{x}'' \neq \boldsymbol{x}} r(\boldsymbol{x}'' \mid \boldsymbol{x})(1 - \alpha(\boldsymbol{x}'' \mid \boldsymbol{x})) & \text{otherwise.} \end{cases} \tag{6.32}$$

We will show that the stationary distribution is p by showing that p satisfies the detailed balance equation (6.25). Let us fix arbitrary states $\boldsymbol{x}$ and $\boldsymbol{x}'$. First, observe that if $\boldsymbol{x} = \boldsymbol{x}'$, then the detailed balance equation is trivially satisfied. Without loss of generality we assume

$$\alpha(\boldsymbol{x} \mid \boldsymbol{x}') = 1, \quad \alpha(\boldsymbol{x}' \mid \boldsymbol{x}) = \frac{q(\boldsymbol{x}')r(\boldsymbol{x} \mid \boldsymbol{x}')}{q(\boldsymbol{x})r(\boldsymbol{x}' \mid \boldsymbol{x})}.$$

For $\boldsymbol{x} \neq \boldsymbol{x}'$, we then have,

$$\begin{aligned}
p(\boldsymbol{x}) \cdot p(\boldsymbol{x}' \mid \boldsymbol{x}) &= \frac{1}{Z} q(\boldsymbol{x}) p(\boldsymbol{x}' \mid \boldsymbol{x}) && \text{using the definition of the distribution } p \\
&= \frac{1}{Z} q(\boldsymbol{x}) r(\boldsymbol{x}' \mid \boldsymbol{x}) \alpha(\boldsymbol{x}' \mid \boldsymbol{x}) && \text{using the transition probabilities of the Markov chain} \\
&= \frac{1}{Z} q(\boldsymbol{x}) r(\boldsymbol{x}' \mid \boldsymbol{x}) \frac{q(\boldsymbol{x}')r(\boldsymbol{x} \mid \boldsymbol{x}')}{q(\boldsymbol{x})r(\boldsymbol{x}' \mid \boldsymbol{x})} && \text{using the definition of the acceptance distribution } \alpha \\
&= \frac{1}{Z} q(\boldsymbol{x}') r(\boldsymbol{x} \mid \boldsymbol{x}') \\
&= \frac{1}{Z} q(\boldsymbol{x}') r(\boldsymbol{x} \mid \boldsymbol{x}') \alpha(\boldsymbol{x} \mid \boldsymbol{x}') && \text{using the definition of the acceptance distribution } \alpha \\
&= \frac{1}{Z} q(\boldsymbol{x}') p(\boldsymbol{x} \mid \boldsymbol{x}') && \text{using the transition probabilities of the Markov chain} \\
&= p(\boldsymbol{x}') \cdot p(\boldsymbol{x} \mid \boldsymbol{x}'). && \text{using the definition of the distribution } p
\end{aligned}$$

□

Note that by the fundamental theorem of ergodic Markov chains (6.19), for convergence to the stationary distribution, it is sufficient for the Markov chain to be ergodic. Ergodicity follows immediately when the transition probabilities $p(\cdot \mid \boldsymbol{x})$ have full support. For example, if the proposal distribution $r(\cdot \mid \boldsymbol{x})$ has full support, the full support of $p(\cdot \mid \boldsymbol{x})$ follows immediately from eq. (6.32). The rate of convergence of Metropolis-Hastings depends strongly on the choice of the proposal distribution, and we will explore different choices of proposal distribution in the following.

6.2.2 *Gibbs Sampling*

A popular example of a Metropolis-Hastings algorithm is Gibbs sampling as presented in alg. 6.20.

Algorithm 6.20: Gibbs sampling

initialize $\boldsymbol{x} = [x_1, \dots, x_n] \in \mathbb{R}^n$
for $t = 1$ **to** T **do**
  pick a variable i uniformly at random from $\{1, \dots, n\}$
  set $\boldsymbol{x}_{-i} \doteq [x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n]$
  update x_i by sampling according to the posterior distribution $p(x_i \mid \boldsymbol{x}_{-i})$

Intuitively, by re-sampling single coordinates according to the posterior distribution given the other coordinates, Gibbs sampling finds states that are successively "more" likely. Selecting the index i uniformly at random ensures that the underlying Markov chain is ergodic provided the conditional distributions $p(\cdot \mid \boldsymbol{x}_{-i})$ have full support.

Theorem 6.21 (Gibbs sampling as Metropolis-Hastings)**.** *Gibbs sampling is a Metropolis-Hastings algorithm. For any fixed $i \in [n]$, it has proposal distribution*

$$r_i(\boldsymbol{x}' \mid \boldsymbol{x}) \doteq \begin{cases} p(x'_i \mid \boldsymbol{x}'_{-i}) & \boldsymbol{x}' \text{ differs from } \boldsymbol{x} \text{ only in entry } i \\ 0 & \text{otherwise} \end{cases} \tag{6.33}$$

and acceptance distribution $\alpha_i(\boldsymbol{x}' \mid \boldsymbol{x}) \doteq 1$.

Proof. We show that $\alpha_i(\boldsymbol{x}' \mid \boldsymbol{x}) = 1$ follows from the definition of an acceptance distribution in Metropolis-Hastings (6.31) and the choice of proposal distribution (6.33).

By (6.31),

$$\alpha_i(\boldsymbol{x}' \mid \boldsymbol{x}) = \min\left\{1, \frac{p(\boldsymbol{x}')r_i(\boldsymbol{x} \mid \boldsymbol{x}')}{p(\boldsymbol{x})r_i(\boldsymbol{x}' \mid \boldsymbol{x})}\right\}$$

Note that $p(\boldsymbol{x}) = p(x_i, \boldsymbol{x}_{-i}) = p(x_i \mid \boldsymbol{x}_{-i})p(\boldsymbol{x}_{-i})$ using the product rule (1.14). Therefore,

$$\begin{aligned} &= \min\left\{1, \frac{p(x'_i \mid \boldsymbol{x}'_{-i})p(\boldsymbol{x}'_{-i})r_i(\boldsymbol{x} \mid \boldsymbol{x}')}{p(x_i \mid \boldsymbol{x}_{-i})p(\boldsymbol{x}_{-i})r_i(\boldsymbol{x}' \mid \boldsymbol{x})}\right\} \\ &= \min\left\{1, \frac{p(x'_i \mid \boldsymbol{x}'_{-i})p(\boldsymbol{x}'_{-i})p(x_i \mid \boldsymbol{x}_{-i})}{p(x_i \mid \boldsymbol{x}_{-i})p(\boldsymbol{x}_{-i})p(x'_i \mid \boldsymbol{x}'_{-i})}\right\} && \text{using the proposal distribution (6.33)} \\ &= \min\left\{1, \frac{p(\boldsymbol{x}'_{-i})}{p(\boldsymbol{x}_{-i})}\right\} \\ &= 1. && \square \quad \text{using that } p(\boldsymbol{x}'_{-i}) = p(\boldsymbol{x}_{-i}) \end{aligned}$$

If the index i is chosen uniformly at random as in alg. 6.20, then the proposal distribution is $r(\boldsymbol{x}' \mid \boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} r_i(\boldsymbol{x}' \mid \boldsymbol{x})$, which analogously to theorem 6.21 has the associated acceptance distribution

$$\alpha(\boldsymbol{x}' \mid \boldsymbol{x}) = \min\left\{1, \frac{p(\boldsymbol{x}')r(\boldsymbol{x} \mid \boldsymbol{x}')}{p(\boldsymbol{x})r(\boldsymbol{x}' \mid \boldsymbol{x})}\right\} = \min\left\{1, \frac{\sum_{i=1}^{n} p(\boldsymbol{x}')r_i(\boldsymbol{x} \mid \boldsymbol{x}')}{\sum_{i=1}^{n} p(\boldsymbol{x})r_i(\boldsymbol{x}' \mid \boldsymbol{x})}\right\} = 1.$$

Corollary 6.22 (Convergence of Gibbs sampling)**.** *As Gibbs sampling is a specific example of an MH-algorithm, the stationary distribution of the simulated Markov chain is* $p(\boldsymbol{x})$.

Note that for the proposals of Gibbs sampling, we have

$$p(x_i \mid \boldsymbol{x}_{-i}) = \frac{p(x_i, \boldsymbol{x}_{-i})}{\sum_{x_i} p(x_i, \boldsymbol{x}_{-i})} = \frac{q(x_i, \boldsymbol{x}_{-i})}{\sum_{x_i} q(x_i, \boldsymbol{x}_{-i})}. \tag{6.34}$$

using the definition of condition probability (1.11) and the sum rule (1.10)

Under many models, this probability can be efficiently evaluated due to the conditioning on the remaining coordinates $\boldsymbol{x}_{-i}$. If X_i has finite support, the normalizer can be computed exactly.

Exercise 6.23: Practical examples of Gibbs sampling

In this exercise, we look at some examples where Gibbs sampling is useful.

1. Consider the distribution

   $$p(x, y) \doteq \binom{n}{x} y^{x+\alpha-1}(1-y)^{n-x+\beta-1}, \quad x \in [n], y \in [0, 1].$$

   Convince yourself that it is hard to sample directly from p and prove that it is an easy task if one uses Gibbs sampling. That is, show that the conditional distributions $p(x \mid y)$ and $p(y \mid x)$ are easy to sample from.
   Hint: Take a look at the Beta distribution (1.67).
2. Consider the following generative model $p(\mu, \lambda, x_{1:n})$ given by the likelihood $x_{1:n} \mid \mu, \lambda \overset{\text{iid}}{\sim} \mathcal{N}(\mu, \lambda^{-1})$ and the priors

   $$\mu \sim \mathcal{N}(\mu_0, \lambda_0^{-1}) \quad \text{and} \quad \lambda \sim \text{Gamma}(\alpha, \beta).$$

   We would like to sample from the posterior $p(\mu, \lambda \mid x_{1:n})$. Show that

   $$\mu \mid \lambda, x_{1:n} \sim \mathcal{N}(m_\lambda, l_\lambda^{-1}) \quad \text{and} \quad \lambda \mid \mu, x_{1:n} \sim \text{Gamma}(a_\mu, b_\mu),$$

   and derive $m_\lambda, l_\lambda, a_\mu, b_\mu$. Such a prior is called a *semi-conjugate prior* to the likelihood, as the prior on μ is conjugate for any fixed value of λ and vice-versa.
3. The *Pareto distribution* was originally used to model the distribution of wealth in a society, but is also used to model many

Gamma distribution The PDF of the *gamma distribution* $\text{Gamma}(\alpha, \beta)$ is defined as

$$\text{Gamma}(x; \alpha, \beta) \propto x^{\alpha-1}e^{-\beta x}, \quad x \in \mathbb{R}_{>0}.$$

A random variable $X \sim \text{Gamma}(\alpha, \beta)$ measures the waiting time until $\alpha > 0$ events occur in a Poisson process with rate $\beta > 0$. In particular, when $\alpha = 1$ then the gamma distribution coincides with the exponential distribution with rate β.

other phenomena such as the size of cities, the frequency of words, and the returns on stocks. Formally, the Pareto distribution is defined by the following PDF,

$$\text{Pareto}(x; \alpha, c) = \frac{\alpha c^\alpha}{x^{\alpha+1}} \mathbb{1}\{x \geq c\}, \quad x \in \mathbb{R} \tag{6.35}$$

where the *tail index* $\alpha > 0$ models the "weight" of the right tail of the distribution (larger α means lighter tail) and $c > 0$ corresponds to a cutoff threshold. The distribution is supported on $[c, \infty)$, and as $\alpha \to \infty$ it approaches a point density at c.

Let us assume that $x_{1:n} \mid \alpha, c \overset{\text{iid}}{\sim} \text{Pareto}(\alpha, c)$ and assume the improper prior $p(\alpha, c) \propto \mathbb{1}\{\alpha, c > 0\}$ which essentially corresponds to a uniform prior (i.e., "no prior"). Derive the posterior $p(\alpha, c \mid x_{1:n})$. Then, also derive the conditional distributions $p(\alpha \mid c, x_{1:n})$ and $p(c \mid \alpha, x_{1:n})$, and observe that they correspond to known distributions / are easy to sample from.

▷ *Solution*

Readings

For more on Gibbs sampling read chapter 11.3 of "Pattern recognition and machine learning" (Bishop and Nasrabadi, 2006).

6.3 Sampling as Optimization

In this section, we discuss more advanced sampling methods. The main idea that we will study is the interpretation of sampling as an optimization problem. We will build towards an optimization view of sampling step-by-step, and first introduce what is commonly called the "energy" of a distribution.

6.3.1 Gibbs Distributions

Gibbs distributions are a special class of distributions that are widely used in machine learning, and which are characterized by an energy.

Definition 6.24 (Gibbs distribution)**.** Formally, a *Gibbs distribution* (also called a *Boltzmann distribution*) is a continuous distribution p whose PDF is of the form

$$p(\boldsymbol{x}) = \frac{1}{Z} \exp(-f(\boldsymbol{x})). \tag{6.36}$$

$f : \mathbb{R}^n \to \mathbb{R}$ is also called an *energy function*. When the energy function f is convex, its Gibbs distribution is called *log-concave*.

A useful property is that Gibbs distributions always have full support.[6] It is often easier to reason about "energies" rather than probabilities as they are neither restricted to be non-negative nor do they have to integrate to 1. Note that the Gibbs distribution belongs to the exponential family (5.57) with sufficient statistic $-f(\boldsymbol{x})$.

[6] This can easily be seen as $\exp(\cdot) > 0$.

Exercise 6.25: Maximum entropy property of Gibbs distribution

1. Let X be a random variable supported on the finite set $\mathcal{T} \subset \mathbb{R}$.[7] Show that the Gibbs distribution with energy function $\frac{1}{T}f(x)$ for some temperature scalar $T \in \mathbb{R}$ is the distribution with maximum entropy of all distributions supported on $\mathcal{T}$ that satisfy the constraint $\mathbb{E}[f(X)] < \infty$.
   Hint: Solve the dual problem.
2. What happens for $T \to \{0, \infty\}$?

▷ *Solution*

[7] The same result can be shown to hold for arbitrary compact subsets.

Observe that the posterior distribution can always be interpreted as a Gibbs distribution as long as prior and likelihood have full support,

$$\begin{aligned} p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}, y_{1:n}) &= \frac{1}{Z} p(\boldsymbol{\theta}) p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta}) && \text{using Bayes' rule (1.59)} \\ &= \frac{1}{Z} \exp(-[-\log p(\boldsymbol{\theta}) - \log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta})]). \end{aligned} \tag{6.37}$$

Thus, defining the energy function

$$f(\boldsymbol{\theta}) \doteq -\log p(\boldsymbol{\theta}) - \log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta}) \tag{6.38}$$

$$= -\log p(\boldsymbol{\theta}) - \sum_{i=1}^{n} \log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}), \tag{6.39}$$

yields

$$p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}, y_{1:n}) = \frac{1}{Z} \exp(-f(\boldsymbol{\theta})). \tag{6.40}$$

Note that f coincides with the loss function used for MAP estimation (1.96). For a uniform prior, the regularization term vanishes and the energy reduces to the negative log-likelihood $\ell_{\text{nll}}(\boldsymbol{\theta}; \mathcal{D})$ (i.e., the loss function of maximum likelihood estimation (1.91)).

Exercise 6.26: Energy function of Bayesian logistic regression

Recall from eq. (5.13) that the energy function of Bayesian logistic regression is

$$f(\boldsymbol{w}) = \lambda \|\boldsymbol{w}\|_2^2 + \sum_{i=1}^{n} \log(1 + \exp(-y_i \boldsymbol{w}^\top \boldsymbol{x}_i)), \tag{6.41}$$

which coincided with the standard optimization objective of (reg-

ularized) logistic regression.

Show that the posterior distribution of Bayesian logistic regression is log-concave.

▷ *Solution*

Using that the posterior is a Gibbs distribution, we can rewrite the acceptance distribution of Metropolis-Hastings,

$$\alpha(\boldsymbol{x}' \mid \boldsymbol{x}) = \min\left\{1, \frac{r(\boldsymbol{x} \mid \boldsymbol{x}')}{r(\boldsymbol{x}' \mid \boldsymbol{x})} \exp(f(\boldsymbol{x}) - f(\boldsymbol{x}'))\right\}. \tag{6.42}$$

this is obtained by substituting the PDF of a Gibbs distribution for the posterior

Example 6.27: Metropolis-Hastings with Gaussian proposals

Let us consider Metropolis-Hastings with the Gaussian proposal distribution,

$$r(\boldsymbol{x}' \mid \boldsymbol{x}) \doteq \mathcal{N}(\boldsymbol{x}'; \boldsymbol{x}, \tau \boldsymbol{I}). \tag{6.43}$$

Due to the symmetry of Gaussians, we have

$$\frac{r(\boldsymbol{x} \mid \boldsymbol{x}')}{r(\boldsymbol{x}' \mid \boldsymbol{x})} = \frac{\mathcal{N}(\boldsymbol{x}; \boldsymbol{x}', \tau \boldsymbol{I})}{\mathcal{N}(\boldsymbol{x}'; \boldsymbol{x}, \tau \boldsymbol{I})} = 1.$$

Hence, the acceptance distribution is defined by

$$\alpha(\boldsymbol{x}' \mid \boldsymbol{x}) = \min\{1, \exp(f(\boldsymbol{x}) - f(\boldsymbol{x}'))\}. \tag{6.44}$$

Intuitively, when a state with lower energy is proposed, that is $f(\boldsymbol{x}') \leq f(\boldsymbol{x})$, then the proposal will always be accepted. In contrast, if the energy of the proposed state is higher, the acceptance probability decreases exponentially in the difference of energies $f(\boldsymbol{x}) - f(\boldsymbol{x}')$. Thus, Metropolis-Hastings minimizes the energy function, which corresponds to minimizing the negative log-likelihood and negative log-prior. The variance in the proposals τ helps in getting around local optima, but the search direction is uniformly random (i.e., "uninformed").

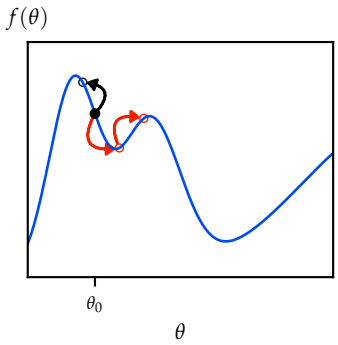


Figure 6.4: Metropolis-Hastings and Langevin dynamics minimize the energy function $f(\theta)$ shown in blue. Suppose we start at the black dot θ_0, then the black and red arrows denote possible subsequent samples. Metropolis-Hastings uses an "uninformed" search direction, whereas Langevin dynamics uses the gradient of $f(\theta)$ to make "more promising" proposals. The random proposals help get past local optima.

6.3.2 *From Energy to Surprise (and back)*

Energy-based models are a well-known class of models in machine learning where an energy function f is learned from data. These energy functions do not need to originate from a probability distribution, yet they induce a probability distribution via their Gibbs distribution $p(\boldsymbol{x}) \propto \exp(-f(\boldsymbol{x}))$. As we have seen in exercise 6.25, this Gibbs distribution is the associated maximum entropy distribution. Observe that

the surprise about $\boldsymbol{x}$ under this distribution is given by

$$\mathrm{S}[p(\boldsymbol{x})] = f(\boldsymbol{x}) + \log Z. \tag{6.45}$$

That is, up to a constant shift, the energy of $\boldsymbol{x}$ *coincides* with the surprise about $\boldsymbol{x}$. Energies are therefore sufficient for comparing the "likelihood" of points, and they do not require normalization.[8]

[8] Intuitively, an energy can be used to compare the "likelihood" of two points $\boldsymbol{x}$ and $\boldsymbol{x}'$ whereas the probability $\boldsymbol{x}$ makes a statement about the "likelihood" of $\boldsymbol{x}$ relative to *all* other points.

What kind of energies could we use? In section 6.3.1, we discussed the use of the negative log-posterior or negative log-likelihood as energies. In general, *any* loss function $\ell(\boldsymbol{x})$ can be thought of as an energy function with an associated maximum entropy distribution $p(\boldsymbol{x}) \propto \exp(-\ell(\boldsymbol{x}))$.

Exercise 6.28: Energy reduction of Gibbs sampling

Let $p(\boldsymbol{x})$ be a probability density over $\mathbb{R}^d$, which we want to sample from. Assume that p is a Gibbs distribution with energy function $f : \mathbb{R}^d \to \mathbb{R}$.

In this exercise, we will study a single round of Gibbs sampling with initial state $\boldsymbol{x}$ and final state $\boldsymbol{x}'$ where

$$x'_j = \begin{cases} x'_i & \text{if } j = i \\ x_j & \text{otherwise} \end{cases}$$

for some fixed index i and $x'_i \sim p(\cdot \mid \boldsymbol{x}_{-i})$.

Show that

$$\mathbb{E}_{x'_i \sim p(\cdot \mid \boldsymbol{x}_{-i})}\left[f(\boldsymbol{x}')\right] \leq f(\boldsymbol{x}) - \mathrm{S}[p(x_i \mid \boldsymbol{x}_{-i})] + \mathrm{H}[p(\cdot \mid \boldsymbol{x}_{-i})]. \tag{6.46}$$

That is, the energy is expected to decrease if the surprise of x_i given $\boldsymbol{x}_{-i}$ is larger than the expected surprise of the new x'_i given $\boldsymbol{x}_{-i}$, i.e., $\mathrm{S}[p(x_i \mid \boldsymbol{x}_{-i})] \geq \mathrm{H}[p(\cdot \mid \boldsymbol{x}_{-i})]$.

Hint: Recall the framing of Gibbs sampling as a variant of Metropolis-Hastings and relate this to the acceptance distribution of Metropolis-Hastings when p is a Gibbs distribution.

▷ *Solution*

6.3.3 Langevin Dynamics

Until now, we have looked at Metropolis-Hastings algorithms with proposal distributions that do not explicitly take into account the curvature of the energy function around the current state. Langevin dynamics adapts the Gaussian proposals of the Metropolis-Hastings algorithm we have seen in example 6.27 to search the state space in an

"informed" direction. The simple idea is to bias the sampling towards states with lower energy, thereby making it more likely that a proposal is accepted.

A natural idea is to shift the proposal distribution perpendicularly to the gradient of the energy function. This yields the following proposal distribution,

$$r(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}'; \boldsymbol{x} - \eta_t \boldsymbol{\nabla} f(\boldsymbol{x}), 2\eta_t \boldsymbol{I}). \tag{6.47}$$

The resulting variant of Metropolis-Hastings is known as the *Metropolis adjusted Langevin algorithm* (MALA) or *Langevin Monte Carlo* (LMC). It can be shown that, as $\eta_t \to 0$, we have for the acceptance probability $\alpha(\boldsymbol{x}' \mid \boldsymbol{x}) \to 1$ using that the acceptance probability is 1 if $\boldsymbol{x}' = \boldsymbol{x}$. Hence, the Metropolis-Hastings acceptance step can be omitted once the rejection probability becomes negligible. The algorithm which always accepts the proposal of eq. (6.47) is known as the *unadjusted Langevin algorithm* (ULA).

Observe that if the stationary distribution is the posterior distribution

$$p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}, y_{1:n}) = \frac{1}{Z} \exp \underbrace{\left(\log p(\boldsymbol{\theta}) + \sum_{i=1}^{n} \log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}) \right)}_{-f(\boldsymbol{\theta})}$$

with energy f as we discussed in section 6.3.1, then the proposal $\boldsymbol{\theta}'$ of MALA/LMC can be equivalently formulated as

$$\begin{aligned} \boldsymbol{\theta}' &\leftarrow \boldsymbol{\theta} - \eta_t \boldsymbol{\nabla} f(\boldsymbol{\theta}) + \epsilon \\ &= \boldsymbol{\theta} + \eta_t \left(\boldsymbol{\nabla} \log p(\boldsymbol{\theta}) + \sum_{i=1}^{n} \boldsymbol{\nabla} \log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}) \right) + \epsilon \end{aligned} \tag{6.48}$$

where $\epsilon \sim \mathcal{N}(\boldsymbol{0}, 2\eta_t \boldsymbol{I})$.

Remark 6.29: ULA is a discretized diffusion

The unadjusted Langevin algorithm can be seen as a discretization of Langevin dynamics, which is a continuous-time stochastic process with a drift and with random stationary and independent Gaussian increments. The randomness is modeled by a Wiener process.

Definition 6.30 (Wiener process)**.** The *Wiener process* (also known as *Brownian motion*) is a sequence of random vectors $\{\mathbf{W}_t\}_{t \geq 0}$ such that

1. $\mathbf{W}_0 = \boldsymbol{0}$,
2. with probability 1, $\mathbf{W}_t$ is continuous in t,

3. the process has independent increments,[9] and
4. $\mathbf{W}_{t+u} - \mathbf{W}_t \sim \mathcal{N}(\mathbf{0}, u\boldsymbol{I})$.

Consider the continuous-time stochastic process $\boldsymbol{\theta}$ defined by the stochastic differential equation (SDE)

$$d\boldsymbol{\theta}_t = \underbrace{-\boldsymbol{\nabla} f(\boldsymbol{\theta}_t)\, dt}_{\text{drift}} + \underbrace{\sqrt{2}\, d\mathbf{W}_t}_{\text{noise}}, \tag{6.49}$$

Such a stochastic process is also called a *diffusion (process)* and eq. (6.49) specifically is called *Langevin dynamics*. Here, the first term is called the "drift" of the process, and the second term is called its "noise". Note that if the noise term is zero then eq. (6.49) is simply an ordinary differential equation (ODE).

A diffusion can be discretized using the Euler-Maruyama method (also called "forward Euler") to obtain a discrete approximation $\boldsymbol{\theta}_k \approx \boldsymbol{\theta}(\tau_k)$ where τ_k denotes the k-th time step. Choosing the time steps such that $\Delta t_k \doteq \tau_{k+1} - \tau_k = \eta_k$ yields the approximation

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \boldsymbol{\nabla} f(\boldsymbol{\theta}_k)\, \Delta t_k + \sqrt{2}\, \Delta \mathbf{W}_k \tag{6.50}$$

where $\Delta \mathbf{W}_k \doteq \mathbf{W}_{\tau_{k+1}} - \mathbf{W}_{\tau_k} \sim \mathcal{N}(\mathbf{0}, \Delta t_k \boldsymbol{I})$. Observe that this coincides with the update rule of Langevin dynamics from eq. (6.48).

For log-concave distributions, the mixing time of the Markov chain underlying Langevin dynamics can be shown to be polynomial in the dimension n (Vempala and Wibisono, 2019).

Exercise 6.31: Mixing time of Langevin dynamics

In this exercise, we will show that for certain Gibbs distributions $p(\boldsymbol{\theta}) \propto \exp(-f(\boldsymbol{\theta}))$, Langevin dynamics is rapidly mixing. To do this, we will observe that Langevin dynamics can be seen as a continuous-time optimization algorithm in the space of distributions.

First, we consider a simpler and more widely-known optimization algorithm, namely the *gradient flow*

$$d\boldsymbol{x}_t = -\boldsymbol{\nabla} f(\boldsymbol{x}_t)\, dt. \tag{6.51}$$

Note that gradient descent is simply the discrete-time approximation of gradient flow just as ULA is the discrete-time approximation of Langevin dynamics. In the analysis of ODEs such as the gradient flow, so-called *Lyapunov functions* are commonly used to prove convergence of $\boldsymbol{x}_t$ to a fixed point (also called an *equilibrium*).

[9] That is, the "future" increments $\mathbf{W}_{t+u} - \mathbf{W}_t$ for $u \geq 0$ are independent of past values $\mathbf{W}_s$ for $s < t$.

Let us assume that f is α-strongly convex for some $\alpha > 0$, that is,

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \boldsymbol{\nabla} f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}) + \frac{\alpha}{2} \|\boldsymbol{y} - \boldsymbol{x}\|_2^2 \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n. \tag{6.52}$$

In words, f is lower bounded by a quadratic function with curvature α. Moreover, assume w.l.o.g. that f minimized at $f(\mathbf{0}) = 0$.[10]

1. Show that f satisfies the *Polyak-Łojasiewicz (PL) inequality*, i.e.,

$$f(\boldsymbol{x}) \leq \frac{1}{2\alpha} \|\boldsymbol{\nabla} f(\boldsymbol{x})\|_2^2 \quad \forall \boldsymbol{x} \in \mathbb{R}^n. \tag{6.53}$$

2. Prove $\frac{d}{dt} f(\boldsymbol{x}_t) \leq -2\alpha f(\boldsymbol{x}_t)$.

Thus, $\mathbf{0}$ is the fixed point of eq. (6.51) and the Lyapunov function f is monotonically decreasing along the trajectory of $\boldsymbol{x}_t$. We recall *Grönwall's inequality* which states that for any real-valued continuous functions $g(t)$ and $\beta(t)$ on the interval $[0, T] \subset \mathbb{R}$ such that $\frac{d}{dt} g(t) \leq \beta(t) g(t)$ for all $t \in [0, T]$ we have

$$g(t) \leq g(0) \exp\left(\int_0^t \beta(s)\, ds \right) \quad \forall t \in [0, T]. \tag{6.54}$$

3. Conclude that $f(\boldsymbol{x}_t) \leq e^{-2\alpha t} f(\boldsymbol{x}_0)$.

Now that we have proven the convergence of gradient flow using f as Lyapunov function, we will follow the same template to prove the convergence of Langevin dynamics to the distribution $p(\boldsymbol{\theta}) \propto \exp(-f(\boldsymbol{\theta}))$. We will use that the evolution of $\{\boldsymbol{\theta}_t\}_{t \geq 0}$ following the Langevin dynamics (6.49) is equivalently characterized by their densities $\{q_t\}_{t \geq 0}$ following the *Fokker-Planck equation*

$$\frac{\partial q_t}{\partial t} = \boldsymbol{\nabla} \cdot (q_t \boldsymbol{\nabla} f) + \Delta q_t. \tag{6.55}$$

Here, $\boldsymbol{\nabla}\cdot$ and Δ are the divergence and Laplacian operators, respectively.[11] Intuitively, the first term of the Fokker-Planck equation corresponds to the drift and its second term corresponds to the diffusion (i.e., the Gaussian noise).[12]

4. Show that $\Delta q_t = \boldsymbol{\nabla} \cdot (q_t \boldsymbol{\nabla} \log q_t)$, implying that the Fokker-Planck equation simplifies to

$$\frac{\partial q_t}{\partial t} = \boldsymbol{\nabla} \cdot \left(q_t \boldsymbol{\nabla} \log \frac{q_t}{p} \right). \tag{6.56}$$

Hint: The Laplacian of a scalar field φ is $\Delta \varphi \doteq \boldsymbol{\nabla} \cdot (\boldsymbol{\nabla} \varphi)$.

Observe that the Fokker-Planck equation already implies that p is indeed a stationary distribution, as if $q_t = p$ then $\frac{\partial q_t}{\partial t} = 0$. Moreover, note the similarity of the integrand of $\mathrm{KL}(q_t \| p)$, $q_t \log \frac{q_t}{p}$, to eq. (6.56). We will therefore use the KL-divergence as Lyapunov function.

[10] This can always be achieved by shifting the coordinate system and subtracting a constant from f.

[11] For ease of notation, we omit the explicit dependence of q_t, p, and f on $\boldsymbol{\theta}$.

[12] Recall that the divergence $\boldsymbol{\nabla} \cdot \mathbf{F}$ of a vector field $\mathbf{F}$ measures the change of volume under the flow of $\mathbf{F}$. That is, if in the small neighborhood of a point $\boldsymbol{x}$, $\mathbf{F}$ points towards $\boldsymbol{x}$, then the divergence at $\boldsymbol{x}$ is negative as the volume shrinks. If $\mathbf{F}$ points away from $\boldsymbol{x}$, then the divergence at $\boldsymbol{x}$ is positive as the volume increases.
The Laplacian $\Delta \varphi = \boldsymbol{\nabla} \cdot (\boldsymbol{\nabla} \varphi)$ of a scalar field φ can be understood intuitively as measuring "heat dissipation". That is, if $\varphi(\boldsymbol{x})$ is smaller than the average value of φ in a small neighborhood of $\boldsymbol{x}$, then the Laplacian at $\boldsymbol{x}$ is positive.
Regarding the Fokker-Planck equation (6.55), the second term Δq_t can therefore be understood as locally dissipating the probability mass of q_t (which is due to the diffusion term in the SDE).
On the other hand, the term $\boldsymbol{\nabla} \cdot (q_t \boldsymbol{\nabla} f)$ can be understood as a Laplacian of f "weighted" by q_t. Intuitively, the vector field $\boldsymbol{\nabla} f$ moves flow in the direction of high energy, and hence, its divergence is larger in regions of lower energy and smaller in regions of higher energy. This term therefore corresponds to a drift from regions of high energy to regions of low energy.

5. Prove $\frac{d}{dt}\mathrm{KL}(q_t\|p) = -\mathrm{J}(q_t\|p)$. Here,

$$\mathrm{J}(q_t\|p) \doteq \mathbb{E}_{\boldsymbol{\theta}\sim q_t}\left[\left\|\boldsymbol{\nabla}\log\frac{q_t(\boldsymbol{\theta})}{p(\boldsymbol{\theta})}\right\|_2^2\right] \tag{6.57}$$

denotes the *relative Fisher information* of p with respect to q_t. *Hint: For any distribution q on $\mathbb{R}^n$,*

$$\int_{\mathbb{R}^n}(\boldsymbol{\nabla}\cdot q\mathbf{F})\varphi\,d\boldsymbol{x} = -\int_{\mathbb{R}^n} q\,\boldsymbol{\nabla}\varphi\cdot\mathbf{F}\,d\boldsymbol{x} \tag{6.58}$$

follows for any vector field $\mathbf{F}$ and scalar field φ from the divergence theorem and the product rule of the divergence operator.

Thus, the relative Fisher information can be seen as the negated time-derivative of the KL-divergence, and as $\mathrm{J}(q_t\|p) \geq 0$ it follows that the KL-divergence is decreasing along the trajectory.

The *log-Sobolev inequality* (LSI) is satisfied by a distribution p with a constant $\alpha > 0$ if for all q:

$$\mathrm{KL}(q\|p) \leq \frac{1}{2\alpha}\mathrm{J}(q\|p). \tag{6.59}$$

It is a classical result that if f is α-strongly convex then p satisfies the LSI with constant α (Bakry and Émery, 2006).

6. Show that if f is α-strongly convex for some $\alpha > 0$ (we say that p is "strongly log-concave"), then $\mathrm{KL}(q_t\|p) \leq e^{-2\alpha t}\mathrm{KL}(q_0\|p)$.
7. Conclude that under the same assumption on f, Langevin dynamics is rapidly mixing, i.e., $\tau_{\mathrm{TV}}(\epsilon) \in \mathcal{O}(\mathrm{poly}(n, \log(1/\epsilon)))$.

To summarize, we have seen that Langevin dynamics is an optimization scheme in the space of distributions, and that its convergence can be analyzed analogously to classical optimization schemes. Notably, in this exercise we have studied continuous-time Langevin dynamics. Convergence guarantees for discrete-time approximations can be derived using the same techniques. If this interests you, refer to "Rapid convergence of the unadjusted Langevin algorithm: Isoperimetry suffices" (Vempala and Wibisono, 2019).

▷ *Solution*

6.3.4 Stochastic Gradient Langevin Dynamics

Note that computing the gradient of the energy function, which corresponds to computing exact gradients of the log-prior and log-likelihood, in every step can be expensive. The proposal step of MALA/LMC can be made more efficient by approximating the gradient with an unbi-

ased gradient estimate, leading to *stochastic gradient Langevin dynamics* (SGLD) shown in alg. 6.32 (Welling and Teh, 2011). Observe that SGLD (6.60) differs from MALA/LMC (6.48) only by using a sample-based approximation of the gradient.

Algorithm 6.32: Stochastic gradient Langevin dynamics, SGLD

initialize $\boldsymbol{\theta}$
for $t = 1$ **to** T **do**
  sample $i_1, \dots, i_m \sim \text{Unif}(\{1, \dots, n\})$ independently
  sample $\epsilon \sim \mathcal{N}(\mathbf{0}, 2\eta_t \boldsymbol{I})$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_t \Big(\boldsymbol{\nabla} \log p(\boldsymbol{\theta}) + \frac{n}{m} \sum_{j=1}^{m} \boldsymbol{\nabla} \log p(y_{i_j} \mid \boldsymbol{x}_{i_j}, \boldsymbol{\theta}) \Big) + \epsilon$ // (6.60)

Intuitively, in the initial phase of the algorithm, the stochastic gradient term dominates, and therefore, SGLD corresponds to a variant of stochastic gradient ascent. In the later phase, the update rule is dominated by the injected noise ϵ, and will effectively be Langevin dynamics. SGLD transitions smoothly between the two phases.

Under additional assumptions, SGLD is guaranteed to converge to the posterior distribution for decreasing learning rates $\eta_t = \Theta(t^{-1/3})$ (Raginsky et al., 2017; Xu et al., 2018). SGLD does not use the acceptance step from Metropolis-Hastings as asymptotically, SGLD corresponds to Langevin dynamics and the Metropolis-Hastings rejection probability goes to zero for a decreasing learning rate.

Readings

For more details on SGLD, read the original paper, "Bayesian learning via stochastic gradient Langevin dynamics" (Welling and Teh, 2011).

6.3.5 *Hamiltonian Monte Carlo*

As MALA and SGLD can be seen as a sampling-based analogue of GD and SGD, a similar analogue for (stochastic) gradient descent with momentum is the (stochastic gradient) *Hamiltonian Monte Carlo* (HMC) algorithm, which we discuss in the following (Duane et al., 1987; Chen et al., 2014).

We have seen that if we want to sample from a distribution

$$p(\boldsymbol{x}) \propto \exp(-f(\boldsymbol{x}))$$

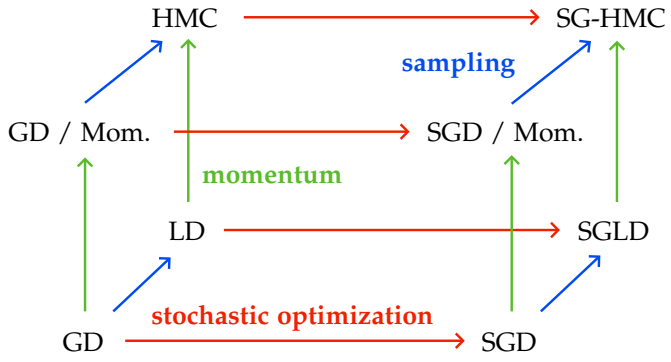


Figure 6.5: A commutative diagram of optimization algorithms. Langevin dynamics (LD) is the non-stochastic variant of SGLD.

with energy function f, we can construct a Markov chain whose distribution converges to p. We have also seen that for this approach to work, the chain must move through all areas of significant probability with reasonable speed.

If one is faced with a distribution p which is multimodal (i.e., that has several "peaks"), one has to ensure that the chain will explore all modes, and can therefore "jump between different areas of the space".

So in general, *local* updates are doomed to fail. Methods such as Metropolis-Hastings with Gaussian proposals, or even Langevin Monte Carlo might face this issue, as they do not jump to distant areas of the state space with significant acceptance probability. It will therefore take a long time to move from one peak to another.

The HMC algorithm is an instance of Metropolis-Hastings which uses momentum to propose distant points that conserve energy, with high acceptance probability. The general idea of HMC is to *lift* samples $\boldsymbol{x}$ to a higher-order space by considering an auxiliary variable $\boldsymbol{y}$ with the same dimension as $\boldsymbol{x}$. We also lift the distribution p to a distribution on the $(\boldsymbol{x}, \boldsymbol{y})$-space by defining a distribution $p(\boldsymbol{y} \mid \boldsymbol{x})$ and setting $p(\boldsymbol{x}, \boldsymbol{y}) \doteq p(\boldsymbol{y} \mid \boldsymbol{x})p(\boldsymbol{x})$. It is common to pick $p(\boldsymbol{y} \mid \boldsymbol{x})$ to be a Gaussian with zero mean and variance $m\boldsymbol{I}$. Hence,

$$p(\boldsymbol{x}, \boldsymbol{y}) \propto \exp\left(-\frac{1}{2m}\left\|\boldsymbol{y}\right\|_2^2 - f(\boldsymbol{x})\right). \tag{6.61}$$

Physicists might recognize the above as the canonical distribution of a Newtonian system if one takes $\boldsymbol{x}$ as the position and $\boldsymbol{y}$ as the momentum. $H(\boldsymbol{x}, \boldsymbol{y}) \doteq \frac{1}{2m}\left\|\boldsymbol{y}\right\|_2^2 + f(\boldsymbol{x})$ is called the *Hamiltonian*. HMC then takes a step in this higher-order space according to the Hamiltonian dynamics,[13]

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{\nabla}_{\boldsymbol{y}} H, \quad \frac{d\boldsymbol{y}}{dt} = -\boldsymbol{\nabla}_{\boldsymbol{x}} H, \tag{6.62}$$

[13] That is, HMC follows the trajectory of these dynamics for some time.

reaching some new point $(\boldsymbol{x}', \boldsymbol{y}')$ and *projecting* back to the state space by selecting $\boldsymbol{x}'$ as the new sample. This is illustrated in fig. 6.6. In the next iteration, we resample the momentum $\boldsymbol{y}' \sim p(\cdot \mid \boldsymbol{x}')$ and repeat the procedure.

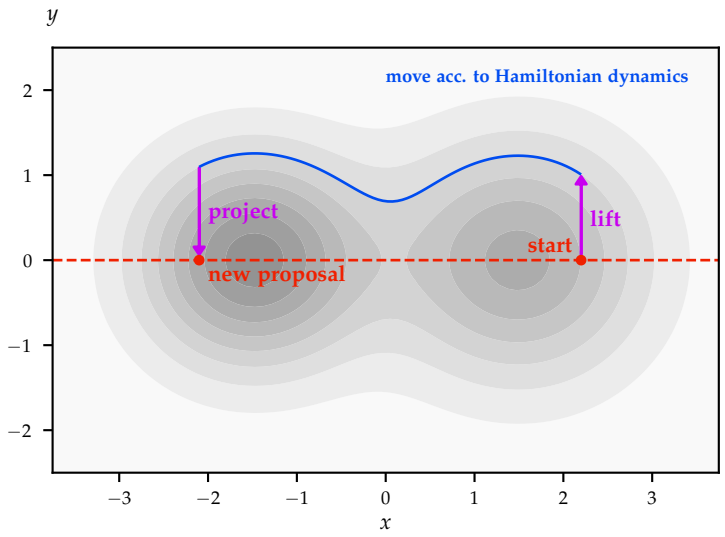


Figure 6.6: Illustration of Hamiltonian Monte Carlo. Shown is the contour plot of a distribution p, which is a mixture of two Gaussians, in the (x, y)-space.

First, the initial point in the state space is lifted to the (x, y)-space. Then, we move according to Hamiltonian dynamics and finally project back onto the state space.

In an implementation of this algorithm, one has to solve eq. (6.62) numerically rather than exactly. Typically, this is done using the *Leapfrog method*, which for a step size τ computes

$$\boldsymbol{y}(t + \tau/2) = \boldsymbol{y}(t) - \frac{\tau}{2} \nabla_{\boldsymbol{x}} f(\boldsymbol{x}(t)) \tag{6.63a}$$

$$\boldsymbol{x}(t + \tau) = \boldsymbol{x}(t) + \frac{\tau}{m} \boldsymbol{y}(t + \tau/2) \tag{6.63b}$$

$$\boldsymbol{y}(t + \tau) = \boldsymbol{y}(t + \tau/2) - \frac{\tau}{2} \nabla_{\boldsymbol{x}} f(\boldsymbol{x}(t + \tau)). \tag{6.63c}$$

Then, one repeats this procedure L times to arrive at a point $(\boldsymbol{x}', \boldsymbol{y}')$. To correct for the resulting discretization error, the proposal is either accepted or rejected in a final Metropolis-Hastings acceptance step. If the proposal distribution is symmetric (which we will confirm in a moment), the acceptance probability is

$$\alpha((\boldsymbol{x}', \boldsymbol{y}') \mid (\boldsymbol{x}, \boldsymbol{y})) \doteq \min\{1, \exp(H(\boldsymbol{x}', \boldsymbol{y}') - H(\boldsymbol{x}, \boldsymbol{y}))\}. \tag{6.64}$$

It follows that $p(\boldsymbol{x}, \boldsymbol{y})$ is the stationary distribution of the Markov chain underlying HMC. Due to the independence of $\boldsymbol{x}$ and $\boldsymbol{y}$, this also implies that the projection to $\boldsymbol{x}$ yields a Markov chain with stationary distribution $p(\boldsymbol{x})$.

So why is the proposal distribution symmetric? This follows from the time-reversibility of Hamiltonian dynamics. It is straightforward to check that the dynamics from eq. (6.62) are identical if we replace t

with $-t$ and $\boldsymbol{y}$ with $-\boldsymbol{y}$. Intuitively, unlike the position $\boldsymbol{x}$, the momentum $\boldsymbol{y}$ is reversed when time is reversed as it depends on the velocity which is the time-derivative of the position.[14] In simpler terms, time-reversibility states that if we observe the evolution of a system (e.g., two billiard balls colliding), we cannot distinguish whether we are observing the system evolve forward or backward in time. The Leapfrog method maintains the time-reversibility of the dynamics.

[14] The momentum of the i-th coordinate is $y_i = m_i v_i$ where m_i is the mass and $v_i = \frac{dx_i}{dt}$ is the velocity.

Symmetry of the proposal distribution is ensured by proposing the point $(\boldsymbol{x}', -\boldsymbol{y}')$.[15] Intuitively, this is simply to ensure that the system is run backward in time as often as it is run forward in time. Recall that the momentum is resampled before each iteration (i.e., the proposed momentum is "discarded") and observe that $p(\boldsymbol{x}', -\boldsymbol{y}') = p(\boldsymbol{x}', \boldsymbol{y}')$,[16] so we can safely disregard the direction of time when computing the acceptance probability in eq. (6.64).

[15] More formally, the proposal distribution is the Dirac delta at $(\boldsymbol{x}', -\boldsymbol{y}')$.

[16] We use here that $p(\boldsymbol{y} \mid \boldsymbol{x})$ was chosen to be symmetric around zero.

Exercise 6.33: Hamiltonian Monte Carlo

1. Prove that if the dynamics are solved exactly (as opposed to numerically using the Leapfrog method), then the acceptance probability of the MH-step is always 1.
2. Prove that the Langevin Monte Carlo algorithm from (6.47) can be seen as a special case of HMC if only one Leapfrog step is used ($L = 1$) and $m = 1$.

▷ *Solution*

To summarize, Markov chain Monte Carlo methods use a Markov chain to approximately sample from an intractable distribution. Note that unlike for variational inference, the convergence of many methods can be guaranteed. Moreover, for log-concave distributions (e.g., with Bayesian logistic regression), the underlying Markov chain converges quickly to the stationary distribution. Methods such as Langevin dynamics and Hamiltonian Monte Carlo aim to accelerate mixing by proposing points with a higher acceptance probability than Metropolis-Hastings with "undirected" Gaussian proposals. In general, the convergence (mixing time) may be slow, meaning that, in practice, accuracy and efficiency have to be traded.

Optional Readings

- Ma, Chen, Jin, Flammarion, and Jordan (2019).
  Sampling can be faster than optimization.
- Teh, Thiery, and Vollmer (2016).
  Consistency and fluctuations for stochastic gradient Langevin dynamics.

- Chen, Fox, and Guestrin (2014).
  Stochastic gradient hamiltonian monte carlo.

7

Bayesian Deep Learning

7.1 Artificial Neural Networks

In practice, it is often seen that models perform better when labels may nonlinearly depend on the inputs. One widely used family of nonlinear functions are *artificial "deep" neural networks*,[1]

$$f : \mathbb{R}^d \to \mathbb{R}^k, \quad f(x;\theta) \doteq \varphi(W_L \varphi(W_{L-1}(\cdots \varphi(W_1 x))))) \tag{7.1}$$

where $\theta \doteq [W_1, \ldots, W_L]$ is a vector of *weights* (written as matrices $W_l \in \mathbb{R}^{n_l \times n_{l-1}}$)[2] and $\varphi : \mathbb{R} \to \mathbb{R}$ is a component-wise nonlinear function. Thus, a deep neural network can be seen as nested ("deep") linear functions composed with nonlinearities.

[1] In the following, we will refrain from using the characterizations "artificial" and "deep" for better readability.

[2] where $n_0 = d$ and $n_L = k$

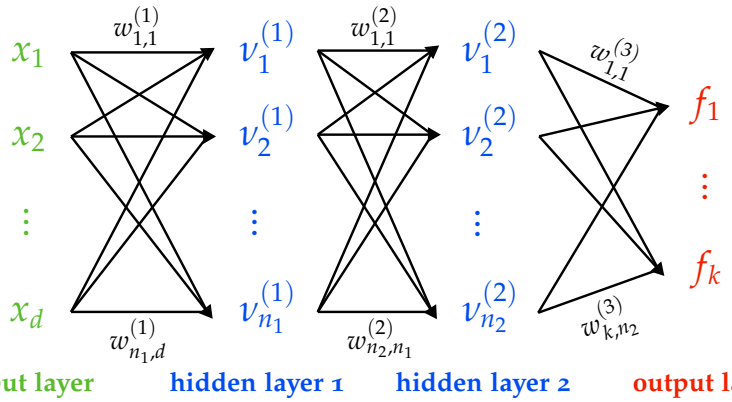


Figure 7.1: Computation graph of a neural network with two hidden layers. This type of neural network is also called a *multilayer perceptron*.

A neural network can be visualized by a *computation graph*. An example for such a computation graph is given in fig. 7.1. The columns of the computation graph are commonly called *layers*, whereby the left-most column is the *input layer*, the right-most column is the *output layer*, and the remaining columns are the *hidden layers*. The inputs are (as we have previously) referred to as $x \doteq [x_1, \ldots, x_d]$. The outputs (i.e., vertices of the output layer) are often referred to as *logits* and

named $\boldsymbol{f} \doteq [f_1, \dots, f_k]$. The *activations* of an individual (hidden) layer l of the neural network are described by

$$\boldsymbol{v}^{(l)} \doteq \boldsymbol{\varphi}(\boldsymbol{W}_l \boldsymbol{v}^{(l-1)}) \tag{7.2}$$

where $\boldsymbol{v}^{(0)} = \boldsymbol{x}$. The activation of the i-th node is $v_i^{(l)} = \boldsymbol{v}^{(l)}(i)$.

7.1.1 Activation Functions

The non-linearity φ is called an *activation function*. The following two activation functions are particularly common:

1. The *hyperbolic tangent* (Tanh) is defined as

$$\mathrm{Tanh}(z) \doteq \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)} \in (-1, 1). \tag{7.3}$$

   Tanh is a scaled and shifted variant of the sigmoid function (5.10) which we have previously seen in the context of logistic regression as $\mathrm{Tanh}(z) = 2\sigma(2z) - 1$.
2. The *rectified linear unit* (ReLU) is defined as

$$\mathrm{ReLU}(z) \doteq \max\{z, 0\} \in [0, \infty). \tag{7.4}$$

   In particular, the ReLU activation function leads to "sparser" gradients as it selects the halfspace of inputs with positive sign. Moreover, the gradients of ReLU do not "vanish" as $z \to \pm\infty$ which can lead to faster training.

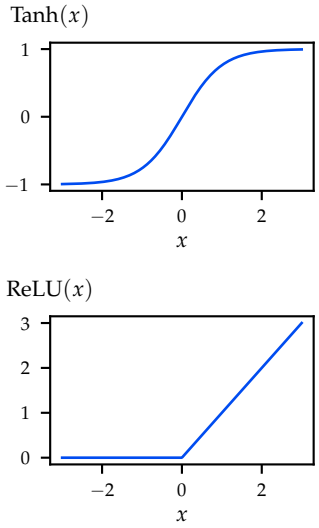


Figure 7.2: The Tanh and ReLU activation functions, respectively.

Remark 7.1: Universal approximation

It is important that the activation function is non-linear because otherwise, any composition of layers would still represent a linear function. Non-linear activation functions allow the network to represent arbitrary functions. The *universal approximation theorem* states that any artificial neural network with just a single hidden layer (with arbitrary width) and activation function φ where φ is not a polynomial can approximate any continuous function to an arbitrary degree of accuracy.

7.1.2 Classification

Although we mainly focus on regression, neural networks can equally well be used for classification. If we want to classify inputs into c separate classes, we can simply construct a neural network with c outputs, $\boldsymbol{f} = [f_1, \dots, f_c]$, and normalize them into a probability distribution. Often, the *softmax function* is used for normalization,

$$\sigma_i(\boldsymbol{f}) \doteq \frac{\exp(f_i)}{\sum_{j=1}^{c} \exp(f_j)} \tag{7.5}$$

where $\sigma_i(\boldsymbol{f})$ corresponds to the probability mass of class i. Note that the softmax corresponds to a Gibbs distribution with energies $-\boldsymbol{f}$. In particular, exercise 6.25 implies that the softmax is the maximum entropy distribution over c classes satisfying $\mathbb{E}_{i\sim\sigma(\boldsymbol{f})}[f_i] \in \mathbb{R}$ for any logits $\boldsymbol{f}$.

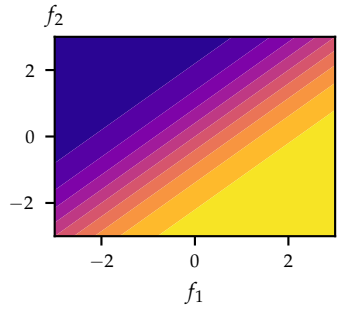


Figure 7.3: Softmax $\sigma_1(f_1, f_2)$ for a binary classification problem. Blue denotes a small probability and yellow denotes a large probability of belonging to class 1, respectively.

Exercise 7.2: Softmax is a generalization of the logistic function

Show that for a two-class classification problem (i.e., $c = 2$), the softmax function is equivalent to the logistic function (5.10) for the univariate model $f \doteq f_1 - f_0$. That is, $\sigma_1(\boldsymbol{f}) = \sigma(f)$ and $\sigma_0(\boldsymbol{f}) = 1 - \sigma(f)$.

Thus, the softmax function is a generalization of the logistic function to more than two classes.

▷ *Solution*

7.1.3 *Maximum Likelihood Estimation*

We will study neural networks under the lens of supervised learning (cf. section 1.2) where we are provided some independently-sampled (noisy) data $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$ generated according to an unknown process $(\boldsymbol{x}, \boldsymbol{y}) \sim p$, which we wish to approximate.

Upon initialization, the network does generally not approximate this process well, so a key element of deep learning is "learning" a parameterization $\boldsymbol{\theta}$ that is a good approximation. To this end, one typically considers a loss function $\ell(\boldsymbol{\theta}; \boldsymbol{y})$ which quantifies the approximation error of the network outputs $\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta})$. In the classical setting of regression, i.e., $y \in \mathbb{R}$ and $k = 1$, ℓ is often taken to be the *(empirical) mean squared error*,

$$\ell_{\text{mse}}(\boldsymbol{\theta}; \mathcal{D}) \doteq \frac{1}{n} \sum_{i=1}^{n} (f(\boldsymbol{x}_i; \boldsymbol{\theta}) - y_i)^2. \tag{7.6}$$

As we have already seen in section 2.1.1 in the context of linear regression, minimizing mean squared error corresponds to maximum likelihood estimation under a Gaussian likelihood.

In the setting of classification where $\boldsymbol{y} \in \{0, 1\}^c$ is a one-hot encoding of class membership,[3] it is instead common to interpret the outputs of a neural network as probabilities akin to our discussion in section 7.1.2. We denote by $q_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{x})$ the resulting probability distribution over classes with PMF $[\sigma_1(\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta})), \ldots, \sigma_c(\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta}))]$, and aim to find $\boldsymbol{\theta}$ such that $q_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) \approx p(\boldsymbol{y} \mid \boldsymbol{x})$. In this context, it is common to minimize the cross-entropy,

$$\mathrm{H}[p \| q_{\boldsymbol{\theta}}] = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim p}[-\log q_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})]$$

[3] That is, exactly one component of $\boldsymbol{y}$ is 1 and all others are 0, indicating to which class the given example belongs.

using the definition of cross-entropy (5.40)

$$\approx \underbrace{-\frac{1}{n}\sum_{i=1}^{n} \log q_{\boldsymbol{\theta}}(\boldsymbol{y}_i \mid \boldsymbol{x}_i)}_{\doteq \ell_{\text{ce}}(\boldsymbol{\theta}; \mathcal{D})} \tag{7.7}$$

using Monte Carlo sampling

which can be understood as minimizing the surprise about the training data under the model. ℓ_{ce} is called the *cross-entropy loss*. Disregarding the constant $1/n$, we can rewrite the cross-entropy loss as

$$\propto -\sum_{i=1}^{n} \log q_{\boldsymbol{\theta}}(\boldsymbol{y}_i \mid \boldsymbol{x}_i) = \ell_{\text{nll}}(\boldsymbol{\theta}; \mathcal{D}) \tag{7.8}$$

Recall that $\ell_{\text{nll}}(\boldsymbol{\theta}; \mathcal{D})$ is the *negative log-likelihood* of the training data, and thus, empirically minimizing cross-entropy can equivalently be interpreted as maximum likelihood estimation.[4] Furthermore, recall from exercise 5.3 that for a two-class classification problem the cross-entropy loss is equivalent to the logistic loss.

[4] We have previously seen this in section 5.4.6. Note that minimizing the cross-entropy, $\mathrm{H}[p \| q_{\boldsymbol{\theta}}]$, is equivalent to minimizing forward-KL, $\mathrm{KL}(p \| q_{\boldsymbol{\theta}})$.

7.1.4 Backpropagation

A crucial property of neural networks is that they are differentiable. That is, we can compute gradients $\boldsymbol{\nabla}_{\boldsymbol{\theta}} \ell$ of $\boldsymbol{f}$ with respect to the parameterization of the model $\boldsymbol{\theta} = \boldsymbol{W}_{1:L}$ and some loss function $\ell(\boldsymbol{f}; \boldsymbol{y})$. Being able to obtain these gradients efficiently allows for "learning" a particular function from data using first-order optimization methods.

The algorithm for computing gradients of a neural network is called *backpropagation* and is essentially a repeated application of the chain rule. Note that using the chain rule for every path through the network is computationally infeasible, as this quickly leads to a combinatorial explosion as the number of hidden layers is increased. The key insight of backpropagation is that we can use the *feed-forward* structure of our neural network to memoize computations of the gradient, yielding a linear time algorithm.

Obtaining gradients by backpropagation is often called *automatic differentiation (auto-diff)*.

Readings

For a thorough introduction to backpropagation, refer to section 4.2 of "Computational Intelligence Lab" (Hofmann, 2022).

Computing the exact gradient for each data point is still fairly expensive when the size of the neural network is large. Typically, stochastic gradient descent is used to obtain unbiased gradient estimates using batches of only m of the n data points, where $m \ll n$.

7.2 Bayesian Neural Networks

How can we adapt our Bayesian approach to neural networks? We use the same strategy which we already used for Bayesian logistic regression, we impose a Gaussian prior on the weights,

$$\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \sigma_p^2 \boldsymbol{I}). \tag{7.9}$$

Similarly, we can use a Gaussian likelihood to describe how well the data is described by the model $\boldsymbol{f}$,

$$y \mid \boldsymbol{x}, \boldsymbol{\theta} \sim \mathcal{N}(f(\boldsymbol{x}; \boldsymbol{\theta}), \sigma_n^2). \tag{7.10}$$

Thus, instead of considering weights as point estimates which are learned exactly, *Bayesian neural networks* learn a distribution over the weights of the network. In principle, other priors and likelihoods can be used, yet Gaussians are typically chosen due to their closedness properties, which we have seen in section 1.6 and many times since.

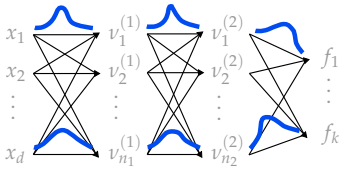


Figure 7.4: Bayesian neural networks model a distribution over the weights of a neural network.

7.2.1 Maximum A Posteriori Estimation

Before studying Bayesian inference, let us first consider MAP estimation in the context of neural networks.

The MAP estimate of the weights is obtained by

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) + \sum_{i=1}^{n} \log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}). \tag{7.11}$$

In section 7.1.3, we have seen that the negative log-likelihood under a Gaussian likelihood (7.10) is the squared error between label and prediction,

$$\log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}) = -\frac{(y_i - f(\boldsymbol{x}_i; \boldsymbol{\theta}))^2}{2\sigma_n^2} + \text{const.} \tag{7.12}$$

Obtaining the MAP estimate instead, simply corresponds to adding an ℓ_2-regularization term to the squared error loss,

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_p^2} \|\boldsymbol{\theta}\|_2^2 + \frac{1}{2\sigma_n^2} \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i; \boldsymbol{\theta}))^2. \tag{7.13}$$

using that for an isotropic Gaussian prior, $\log p(\boldsymbol{\theta}) = -\frac{1}{2\sigma_p^2} \|\boldsymbol{\theta}\|_2^2 + \text{const}$

As we have already seen in remark 1.62 and the context of Bayesian linear regression (and ridge regression), using a Gaussian prior is equivalent to applying weight decay.[5] Using gradient ascent, we obtain the following update rule,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}(1 - \lambda \eta_t) + \eta_t \sum_{i=1}^{n} \boldsymbol{\nabla} \log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}) \tag{7.14}$$

where $\lambda \doteq 1/\sigma_p^2$. The gradients of the likelihood can be obtained using automatic differentiation.

[5] Recall that weight decay regularizes weights by shrinking them towards zero.

7.2.2 Heteroscedastic Noise

Equation (7.10) uses the scalar parameter σ_n^2 to model the aleatoric uncertainty (label noise), similarly to how we modeled the label noise in Bayesian linear regression and Gaussian processes. Such a noise model is called *homoscedastic* as the noise is assumed to be uniform across the domain. In many settings, however, the noise is inherently *heteroscedastic*. That is, the noise varies depending on the input and which "region" of the domain the input is from. This behavior is visualized in fig. 7.5.

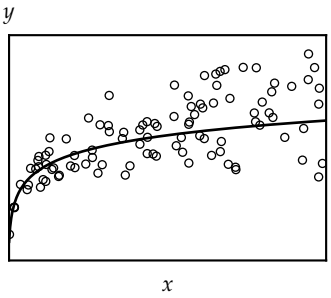


Figure 7.5: Illustration of data with variable (heteroscedastic) noise. The noise increases as the inputs increase in magnitude. The noise-free function is shown in black.

There is a natural way of modeling heteroscedastic noise with Bayesian neural networks. We use a neural network with two outputs f_1 and f_2 and define

$$y \mid \boldsymbol{x}, \boldsymbol{\theta} \sim \mathcal{N}(\mu(\boldsymbol{x}; \boldsymbol{\theta}), \sigma^2(\boldsymbol{x}; \boldsymbol{\theta})) \quad \text{where} \tag{7.15a}$$

$$\mu(\boldsymbol{x}; \boldsymbol{\theta}) \doteq f_1(\boldsymbol{x}; \boldsymbol{\theta}), \tag{7.15b}$$

$$\sigma^2(\boldsymbol{x}; \boldsymbol{\theta}) \doteq \exp(f_2(\boldsymbol{x}; \boldsymbol{\theta})). \tag{7.15c}$$

we exponentiate f_2 to ensure non-negativity of the variance

Using this model, the likelihood term from eq. (7.11) is

$$\begin{aligned}
\log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}) &= \log \mathcal{N}(y_i; \mu(\boldsymbol{x}_i; \boldsymbol{\theta}), \sigma^2(\boldsymbol{x}_i; \boldsymbol{\theta})) \\
&= \log \frac{1}{\sqrt{2\pi\sigma^2(\boldsymbol{x}_i; \boldsymbol{\theta})}} - \frac{(y_i - \mu(\boldsymbol{x}_i; \boldsymbol{\theta}))^2}{2\sigma^2(\boldsymbol{x}_i; \boldsymbol{\theta})} \\
&= \underbrace{\log \frac{1}{\sqrt{2\pi}}}_{\text{const}} - \frac{1}{2}\left[\log \sigma^2(\boldsymbol{x}_i; \boldsymbol{\theta}) + \frac{(y_i - \mu(\boldsymbol{x}_i; \boldsymbol{\theta}))^2}{\sigma^2(\boldsymbol{x}_i; \boldsymbol{\theta})}\right].
\end{aligned} \tag{7.16}$$

note that the normalizing constant depends on the noise model!

Hence, the model can either explain the label y_i by an accurate model $\mu(\boldsymbol{x}_i; \boldsymbol{\theta})$ or by a large variance $\sigma^2(\boldsymbol{x}_i; \boldsymbol{\theta})$, yet, it is penalized for choosing large variances. Intuitively, this allows to attenuate losses for some data points by attributing them to large variance (when no model reflecting all data points simultaneously can be found). This allows the model to "learn" its aleatoric uncertainty. However, recall that the MAP estimate still corresponds to a point estimate of the weights, so we forgo modeling the epistemic uncertainty.

7.3 Approximate Inference

Naturally, we want to understand the epistemic uncertainty of our model too. However, learning and inference in Bayesian neural networks are generally intractable (even when using a Gaussian prior and likelihood) when the noise is not assumed to be homoscedastic and known.[6] Thus, we are led to the techniques of approximate inference, which we discussed in the previous two chapters.

[6] In this case, the conjugate prior to a Gaussian likelihood is not a Gaussian. See, e.g., `https://en.wikipedia.org/wiki/Conjugate_prior`.

7.3.1 Variational Inference

As we have discussed in chapter 5, we can apply black box stochastic variational inference which — in the context of neural networks — is also known as *Bayes by Backprop*. As variational family, we use the family of independent Gaussians which we have already encountered in example 5.7.[7] Recall the fundamental objective of variational inference (5.30),

[7] Independent Gaussians are useful because they can be encoded using only $2d$ parameters, which is crucial when the size of the neural network is large.

$$\begin{aligned}
&\arg\min_{q\in\mathcal{Q}} \mathrm{KL}(q\|p(\cdot \mid \boldsymbol{x}_{1:n}, y_{1:n})) \\
&= \arg\max_{q\in\mathcal{Q}} L(q, p; \mathcal{D}) && \text{using eq. (5.62)} \\
&= \arg\max_{q\in\mathcal{Q}} \mathbb{E}_{\boldsymbol{\theta}\sim q}[\log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta})] - \mathrm{KL}(q\|p(\cdot)). && \text{using eq. (5.63d)}
\end{aligned}$$

The KL-divergence $\mathrm{KL}(q\|p(\cdot))$ can be expressed in closed-form for Gaussians.[8] Recall that we can obtain unbiased gradient estimates of the expectation using the reparameterization trick (5.85),

[8] see eq. (5.50)

$$\mathbb{E}_{\boldsymbol{\theta}\sim q}[\log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta})] = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\Big[\log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\Sigma}^{1/2}\boldsymbol{\epsilon}+\boldsymbol{\mu}}\Big].$$

As $\boldsymbol{\Sigma}$ is the diagonal matrix $\mathrm{diag}\{\sigma_1^2, \ldots, \sigma_d^2\}$, $\boldsymbol{\Sigma}^{1/2} = \mathrm{diag}\{\sigma_1, \ldots, \sigma_d\}$. The gradients of the likelihood can be obtained using backpropagation.

We can now use the variational posterior q_λ to perform approximate inference,

$$\begin{aligned}
p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n}) &= \int p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}, y_{1:n})\, d\boldsymbol{\theta} && \text{using the sum rule (1.10) and product rule (1.14)} \\
&= \mathbb{E}_{\boldsymbol{\theta}\sim p(\cdot|\boldsymbol{x}_{1:n}, y_{1:n})}[p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta})] && \text{interpreting the integral as an expectation over the posterior} \\
&\approx \mathbb{E}_{\theta\sim q_\lambda}[p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta})] && \text{approximating the posterior with the variational posterior } q_\lambda \\
&\approx \frac{1}{m}\sum_{i=1}^{m} p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}^{(i)}) && \text{using Monte Carlo sampling} \qquad (7.17)
\end{aligned}$$

for independent samples $\boldsymbol{\theta}^{(i)} \overset{\text{iid}}{\sim} q_\lambda$,

$$= \frac{1}{m}\sum_{i=1}^{m} \mathcal{N}(y^\star; \mu(\boldsymbol{x}^\star; \boldsymbol{\theta}^{(i)}), \sigma^2(\boldsymbol{x}^\star; \boldsymbol{\theta}^{(i)})). \qquad (7.18) \quad \text{using the neural network}$$

Intuitively, variational inference in Bayesian neural networks can be interpreted as averaging the predictions of multiple neural networks drawn according to the variational posterior q_λ.

Using the Monte Carlo samples $\boldsymbol{\theta}^{(i)}$, we can also estimate the mean of our predictions,

$$\mathbb{E}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, y_{1:n}] \approx \frac{1}{m}\sum_{i=1}^{m} \mu(\boldsymbol{x}^\star; \boldsymbol{\theta}^{(i)}) \doteq \overline{\mu}(\boldsymbol{x}^\star), \qquad (7.19)$$

and the variance of our predictions,

$$\begin{aligned}\mathrm{Var}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, y_{1:n}] &= \mathbb{E}_{\boldsymbol{\theta}}\left[\mathrm{Var}_{y^\star}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}]\right] + \mathrm{Var}_{\boldsymbol{\theta}}\left[\mathbb{E}_{y^\star}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}]\right] && \text{using the law of total variance (1.50)} \\ &= \mathbb{E}_{\boldsymbol{\theta}}\left[\sigma^2(\boldsymbol{x}^\star; \boldsymbol{\theta})\right] + \mathrm{Var}_{\boldsymbol{\theta}}[\mu(\boldsymbol{x}^\star; \boldsymbol{\theta})]. && \text{using the likelihood (7.15a)}\end{aligned}$$

Recall from eq. (2.22) that the first term corresponds to the aleatoric uncertainty of the data and the second term corresponds to the epistemic uncertainty of the model. We can approximate them using the Monte Carlo samples $\boldsymbol{\theta}^{(i)}$,

$$\begin{aligned}\mathrm{Var}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, y_{1:n}] \approx \frac{1}{m}\sum_{i=1}^{m} \sigma^2(\boldsymbol{x}^\star; \boldsymbol{\theta}^{(i)}) \\ + \frac{1}{m-1}\sum_{i=1}^{m}(\mu(\boldsymbol{x}^\star; \boldsymbol{\theta}^{(i)}) - \overline{\mu}(\boldsymbol{x}^\star))^2\end{aligned} \tag{7.20}$$

using a sample mean (1.68) and sample variance (1.69).

7.3.2 *Markov Chain Monte Carlo*

As we have discussed in chapter 6, an alternative method to approximating the full posterior distribution is to sample from it directly. By the ergodic theorem (6.28), we can use any of the discussed sampling strategies to obtain samples $\boldsymbol{\theta}^{(t)}$ such that

$$p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n}) \approx \frac{1}{T}\sum_{t=1}^{T} p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}^{(t)}). \qquad \text{see (6.29)}$$

Here, we omit the offset t_0 which is commonly used to avoid the "burn-in" period for simplicity. Algorithms such as SGLD or SG-HMC are often used as they rely only on stochastic gradients of the loss function which can be computed efficiently using automatic differentiation.

Typically, for large networks, we cannot afford to store all T samples of models. Thus, we need to summarize the iterates.[9] One approach is to keep track of m *snapshots* of weights $[\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(m)}]$ according to some schedule and use those for inference (e.g., by averaging the predictions of the corresponding neural networks). This approach of sampling a subset of some data is generally called *subsampling*.

[9] That is, combine the individual samples of weights $\boldsymbol{\theta}^{(i)}$.

Another approach is to summarize (that is, approximate) the weights using sufficient statistics (e.g., a Gaussian).[10] In other words, we learn the Gaussian approximation,

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad \text{where} \tag{7.21a}$$

$$\boldsymbol{\mu} \doteq \frac{1}{T}\sum_{i=1}^{T} \boldsymbol{\theta}^{(i)}, \tag{7.21b}$$

[10] A statistic is *sufficient* for a family of probability distributions if the samples from which it is calculated yield no more information than the statistic with respect to the learned parameters. We provide a formal definition in section 8.2.

using a sample mean (1.68)

$$\boldsymbol{\Sigma} \doteq \frac{1}{T-1}\sum_{i=1}^{T}(\boldsymbol{\theta}^{(i)}-\boldsymbol{\mu})(\boldsymbol{\theta}^{(i)}-\boldsymbol{\mu})^\top, \tag{7.21c}$$

using a sample variance (1.69)

using sample means and sample (co)variances. This can be implemented efficiently using running averages of the first and second moments,

$$\boldsymbol{\mu} \leftarrow \frac{1}{T+1}(T\boldsymbol{\mu}+\boldsymbol{\theta}) \quad \text{and} \quad \boldsymbol{A} \leftarrow \frac{1}{T+1}(T\boldsymbol{A}+\boldsymbol{\theta}\boldsymbol{\theta}^\top) \tag{7.22}$$

upon observing the fresh sample $\boldsymbol{\theta}$. $\boldsymbol{\Sigma}$ can easily be calculated from these moments,

$$\boldsymbol{\Sigma} = \frac{T}{T-1}(\boldsymbol{A}-\boldsymbol{\mu}\boldsymbol{\mu}^\top). \tag{7.23}$$

using the characterization of sample variance in terms of estimators of the first and second moment (1.70)

To predict, we can sample weights $\boldsymbol{\theta}$ from the learned Gaussian.

Remark 7.3: Stochastic weight averaging

It turns out that this approach works well even without injecting additional Gaussian noise during training, e.g., when using SGD rather than SGLD. Simply taking the mean of the iterates of SGD is called *stochastic weight averaging* (SWA). Describing the iterates of SGD by Gaussian sufficient statistics (analogously to eq. (7.21)), is known as the *stochastic weight averaging-Gaussian* (SWAG) method (Izmailov et al., 2018).

7.3.3 *Dropout and Dropconnect*

We will now discuss two approximate inference techniques that are tailored to neural networks. The first is "dropout"/"dropconnect" regularization. Traditionally, *dropout regularization* (Hinton et al., 2012; Srivastava et al., 2014) randomly omits vertices of the computation graph during training, see fig. 7.7. In contrast, *dropconnect regularization* (Wan et al., 2013) randomly omits edges of the computation graph. The key idea that we will present here is to interpret this type of regularization as performing variational inference.

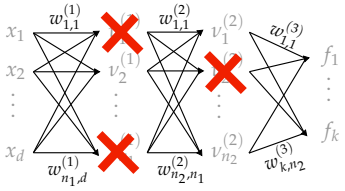


Figure 7.6: Illustration of dropout regularization. Some vertices of the computation graph are randomly omitted. In contrast, dropconnect regularization randomly omits edges of the computation graph.

In our exposition, we will focus on dropconnect, but the same approach also applies to dropout (Gal and Ghahramani, 2016). Suppose that we omit an edge of the computation graph (i.e., set its weight to zero) with probability p. Then our variational posterior is given by

$$q(\boldsymbol{\theta} \mid \boldsymbol{\lambda}) \doteq \prod_{j=1}^{d} q_j(\theta_j \mid \lambda_j) \tag{7.24}$$

where d is the number of weights of the neural network and

$$q_j(\theta_j \mid \lambda_j) \doteq p\delta_0(\theta_j) + (1-p)\delta_{\lambda_j}(\theta_j). \tag{7.25}$$

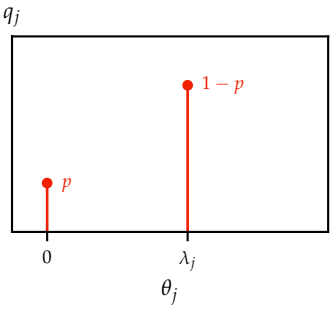


Figure 7.7: Interpretation of dropconnect regularization as variational inference. The only coordinates where the variational posterior q_j has positive probability are 0 and λ_j.

Here, δ_α is the Dirac delta function with point mass at α.[11] The variational parameters $\boldsymbol{\lambda}$ correspond to the "original" weights of the network. In words, the variational posterior expresses that the j-th weight has value 0 with probability p and value λ_j with probability $1-p$. For fixed weights $\boldsymbol{\lambda}$, sampling from the variational posterior $q_{\boldsymbol{\lambda}}$ corresponds to sampling a vector $\boldsymbol{z}$ with entries $\boldsymbol{z}(i) \sim \text{Bern}(p)$, yielding $\boldsymbol{z} \odot \boldsymbol{\lambda}$ which is one of 2^d possible subnetworks.[12]

[11] see example 1.18

[12] $\boldsymbol{A} \odot \boldsymbol{B}$ denotes the Hadamard (element-wise) product.

The weights $\boldsymbol{\lambda}$ can be learned by maximizing the ELBO, analogously to black-box variational inference. The KL-divergence term of the ELBO is not tractable for the variational family described by eq. (7.25), instead a common approach is to use a mixture of Gaussians:

$$q_j(\theta_j \mid \lambda_j) \doteq p\mathcal{N}(\theta_j; 0, 1) + (1-p)\mathcal{N}(\theta_j; \lambda_j, 1). \tag{7.26}$$

In this case, it can be shown that $\text{KL}(q_{\boldsymbol{\lambda}} \| p(\cdot)) \approx \frac{p}{2} \|\boldsymbol{\lambda}\|_2^2$ for sufficiently large d (Gal and Ghahramani, 2015, proposition 1). Thus,

$$\begin{aligned}
&\arg\max_{\boldsymbol{\lambda} \in \Lambda} L(q_{\boldsymbol{\lambda}}, p; \mathcal{D}) \\
&= \arg\max_{\boldsymbol{\lambda} \in \Lambda} \mathbb{E}_{\boldsymbol{\theta} \sim q_{\boldsymbol{\lambda}}}[\log p(\boldsymbol{y}_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta})] - \text{KL}(q \| p(\cdot)) && \text{using eq. (5.63d)} \\
&\approx \arg\min_{\boldsymbol{\lambda} \in \Lambda} -\frac{1}{m} \sum_{i=1}^{m} \log p(\boldsymbol{y}_{1:n} \mid \boldsymbol{x}_{1:n}, \boldsymbol{\theta})\big|_{\boldsymbol{\theta} = \boldsymbol{z}^{(i)} \odot \boldsymbol{\lambda} + \boldsymbol{\epsilon}^{(i)}} + \frac{p}{2} \|\boldsymbol{\lambda}\|_2^2 && \text{using Monte Carlo sampling} \quad (7.27)
\end{aligned}$$

where we reparameterize $\boldsymbol{\theta} \sim q_{\boldsymbol{\lambda}}$ by $\boldsymbol{\theta} = \boldsymbol{z} \odot \boldsymbol{\lambda} + \boldsymbol{\epsilon}$ with $\boldsymbol{z}(i) \sim \text{Bern}(p)$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Equation (7.27) is the standard ℓ_2-regularized loss function of a neural network with weights $\boldsymbol{\lambda}$ and dropconnect, and it is straightforward to obtain unbiased gradient estimates by automatic differentiation.

Crucially, for the interpretation of dropconnect regularization as variational inference to be valid, we also need to perform dropconnect regularization during inference,

$$\begin{aligned}
p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n}) &\approx \mathbb{E}_{\boldsymbol{\theta} \sim q_{\boldsymbol{\lambda}}}[p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta})] \\
&\approx \frac{1}{m} \sum_{i=1}^{m} p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}^{(i)}) && \text{using Monte Carlo sampling} \quad (7.28)
\end{aligned}$$

where $\boldsymbol{\theta}^{(i)} \overset{\text{iid}}{\sim} q_{\boldsymbol{\lambda}}$ are independent samples. This coincides with our earlier discussion of variational inference for Bayesian neural networks in eq. (7.17). In words, we average the predictions of m neural networks for each of which we randomly "drop out" weights.

7.3.4 *Probabilistic Ensembles*

We have seen that variational inference in the context of Bayesian neural networks can be interpreted as averaging the predictions of m neural networks drawn according to the variational posterior.

A natural adaptation of this idea is to immediately learn the weights of m neural networks. The idea is to randomly choose m training sets by sampling uniformly from the data with replacement. Then, using our analysis from section 7.2.1, we obtain m MAP estimates of the weights $\boldsymbol{\theta}^{(i)}$, yielding the approximation

$$\begin{aligned} p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n}) &= \mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot \mid \boldsymbol{x}_{1:n}, y_{1:n})}[p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta})] \\ &\approx \frac{1}{m} \sum_{i=1}^{m} p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}^{(i)}). \end{aligned} \tag{7.29}$$

using bootstrapping

Here, the randomly chosen m training sets induce "diversity" of the models. In practice, in the context of deep neural networks where the global minimizer of the loss can rarely be identified, it is common to use the full training data to train each of the m neural networks. Random initialization and random shuffling of the training data is typically enough to ensure sufficient diversity between models (Lakshminarayanan et al., 2017).

Note that eq. (7.29) is not equivalent to Monte Carlo sampling, although it looks very similar. The key difference is that this approach does not sample from the true posterior distribution p, but instead from the empirical posterior distribution $\hat{p}$ given the (re-sampled) MAP estimates. Intuitively, this can be understood as the difference between sampling from a distribution p directly (Monte Carlo sampling) versus sampling from an approximate (empirical) distribution $\hat{p}$ (corresponding to the training data), which itself is constructed from samples of the true distribution p. This approach is known as *bootstrapping* or *bagging* (short for *b*ootstrap *agg*rega*ting*) and plays a central role in model-free reinforcement learning. We will return to this concept in section 11.4.1.

7.4 Calibration

A key challenge of Bayesian deep learning (and also other Bayesian methods) is the calibration of models. We say that a model is *well-calibrated* if its confidence coincides with its accuracy across many predictions. Consider a classification model that predicts that the label of a given input belongs to some class with probability 80%. If the model is well-calibrated, then the prediction is correct about 80% of the time. In other words, during calibration, we adjust the probability estimation of the model.

We will first mention two methods of estimating the calibration of a model, namely the marginal likelihood and reliability diagrams. Then, in section 7.4.3, we survey commonly used heuristics for empirically improving the calibration.

7.4.1 Evidence

A popular method (which we already encountered multiple times) is to use the evidence of a validation set $\boldsymbol{x}_{1:m}^{\text{val}}$ of size m given the training set $\boldsymbol{x}_{1:n}^{\text{train}}$ of size n for estimating the model calibration. Here, the evidence can be understood as describing how well the validation set is described by the model trained on the training set. We obtain,

$$\begin{aligned}
&\log p(y_{1:m}^{\text{val}} \mid \boldsymbol{x}_{1:m}^{\text{val}}, \boldsymbol{x}_{1:n}^{\text{train}}, y_{1:n}^{\text{train}}) \\
&= \log \int p(y_{1:m}^{\text{val}} \mid \boldsymbol{x}_{1:m}^{\text{val}}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:n}^{\text{train}}, y_{1:n}^{\text{train}}) \, d\boldsymbol{\theta} && \text{using the sum rule (1.10) and product rule (1.14)} \\
&\approx \log \int p(y_{1:m}^{\text{val}} \mid \boldsymbol{x}_{1:m}^{\text{val}}, \boldsymbol{\theta}) q_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) \, d\boldsymbol{\theta} && \text{approximating with the variational posterior} \\
&= \log \int \prod_{i=1}^{m} p(y_i^{\text{val}} \mid \boldsymbol{x}_i^{\text{val}}, \boldsymbol{\theta}) q_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) \, d\boldsymbol{\theta} && \text{using the independence of the data} \qquad (7.30)
\end{aligned}$$

The resulting integrals are typically very small which leads to numerical instabilities. Therefore, it is common to maximize a lower bound to the evidence instead,

$$\begin{aligned}
&= \log \mathbb{E}_{\boldsymbol{\theta} \sim q_{\boldsymbol{\lambda}}} \left[\prod_{i=1}^{m} p(y_i^{\text{val}} \mid \boldsymbol{x}_i^{\text{val}}, \boldsymbol{\theta}) \right] && \text{interpreting the integral as an expectation over the variational posterior} \\
&\geq \mathbb{E}_{\boldsymbol{\theta} \sim q_{\boldsymbol{\lambda}}} \left[\sum_{i=1}^{m} \log p(y_i^{\text{val}} \mid \boldsymbol{x}_i^{\text{val}}, \boldsymbol{\theta}) \right] && \text{using Jensen's inequality (5.37)} \qquad (7.31) \\
&\approx \frac{1}{k} \sum_{j=1}^{k} \sum_{i=1}^{m} \log p(y_i^{\text{val}} \mid \boldsymbol{x}_i^{\text{val}}, \boldsymbol{\theta}^{(j)}) && \text{using Monte Carlo sampling} \qquad (7.32)
\end{aligned}$$

for independent samples $\boldsymbol{\theta}^{(j)} \overset{\text{iid}}{\sim} q_{\boldsymbol{\lambda}}$.

7.4.2 Reliability Diagrams

Reliability diagrams take a frequentist perspective to estimate the calibration of a model. For simplicity, we assume a calibration problem with two classes, 1 and -1 (similarly to logistic regression).[13]

[13] Reliability diagrams generalize beyond this restricted example.

We group the predictions of a validation set into M interval bins of size $1/M$ according to the class probability predicted by the model, $\mathbb{P}(Y_i = 1 \mid \boldsymbol{x}_i)$. We then compare within each bin, how often the model thought the inputs belonged to the class (confidence) with how often the inputs actually belonged to the class (frequency). Formally, we define B_m as the set of samples falling into bin m and let

$$\text{freq}(B_m) \doteq \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}\{Y_i = 1\} \qquad (7.33)$$

be the proportion of samples in bin m that belong to class 1 and let

$$\text{conf}(B_m) \doteq \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{P}(Y_i = 1 \mid \boldsymbol{x}_i) \qquad (7.34)$$

be the average confidence of samples belonging to class 1 within the bin m.

Thus, a model is well calibrated if $\text{freq}(B_m) \approx \text{conf}(B_m)$ for each bin $m \in [M]$. There are two common metrics of calibration that quantify how "close" a model is to being well calibrated.

1. The *expected calibration error* (ECE) is the average deviation of a model from perfect calibration,

$$\ell_{\text{ECE}} \doteq \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\text{freq}(B_m) - \text{conf}(B_m)\right| \tag{7.35}$$

   where n is the size of the validation set.
2. The *maximum calibration error* (MCE) is the maximum deviation of a model from perfect calibration among all bins,

$$\ell_{\text{MCE}} \doteq \max_{m \in [M]} \left|\text{freq}(B_m) - \text{conf}(B_m)\right|. \tag{7.36}$$

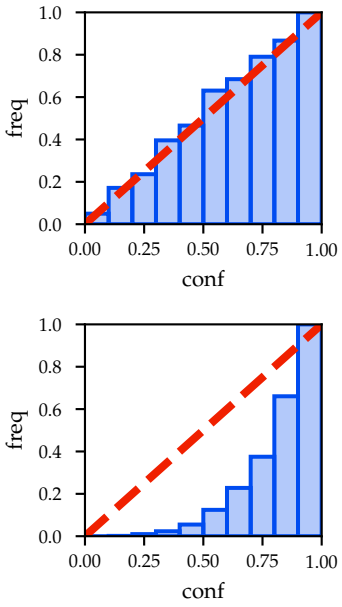


Figure 7.8: Examples of reliability diagrams with ten bins. A perfectly calibrated model approximates the diagonal dashed red line. The first reliability diagram shows a well-calibrated model. In contrast, the second reliability diagram shows an overconfident model.

7.4.3 *Improving Calibration*

We now survey a few heuristics which can be used empirically to improve model calibration.

1. *Histogram binning* assigns a calibrated score $q_m \doteq \text{freq}(B_m)$ to each bin during validation. Then, during inference, we return the calibrated score q_m of the bin corresponding to the prediction of the model.
2. *Isotonic regression* extends histogram binning by using variable bin boundaries. We find a piecewise constant function $\boldsymbol{f} \doteq [f_1, \dots, f_M]$ that minimizes the bin-wise squared loss,

$$\min_{M, \boldsymbol{f}, \boldsymbol{a}} \quad \sum_{m=1}^{M} \sum_{i=1}^{n} \mathbb{1}\{a_m \leq \mathbb{P}(Y_i = 1 \mid \boldsymbol{x}_i) < a_{m+1}\}(f_m - y_i)^2 \tag{7.37a}$$

$$\text{subject to} \quad 0 = a_1 \leq \cdots \leq a_{M+1} = 1, \tag{7.37b}$$

$$f_1 \leq \cdots \leq f_M \tag{7.37c}$$

   where $\boldsymbol{f}$ are the calibrated scores and $\boldsymbol{a} \doteq [a_1, \dots, a_{M+1}]$ are the bin boundaries. We then return the calibrated score f_m of the bin corresponding to the prediction of the model.
3. *Platt scaling* adjusts the logits z_i of the output layer to

$$q_i \doteq \sigma(a z_i + b) \tag{7.38}$$

   and then learns parameters $a, b \in \mathbb{R}$ to maximize the likelihood.
4. *Temperature scaling* is a special and widely used instance of Platt scaling where $a \doteq 1/T$ and $b \doteq 0$ for some temperature scalar

$T \in \mathbb{R}$,

$$q_i \doteq \sigma\left(\frac{z_i}{T}\right). \tag{7.39}$$

Intuitively, for a larger temperature T, the probability is distributed more evenly among the classes (without changing the ranking), yielding a more uncertain prediction. In contrast, for a lower temperature T, the probability is concentrated more towards the top choices, yielding a less uncertain prediction. As seen in exercise 6.25, temperature scaling can be motivated as tuning the mean of the softmax distribution.

Optional Readings

- Guo, Pleiss, Sun, and Weinberger (2017).
  On calibration of modern neural networks.
- Blundell, Cornebise, Kavukcuoglu, and Wierstra (2015).
  Weight uncertainty in neural network.
- Kendall and Gal (2017).
  What uncertainties do we need in bayesian deep learning for computer vision?.

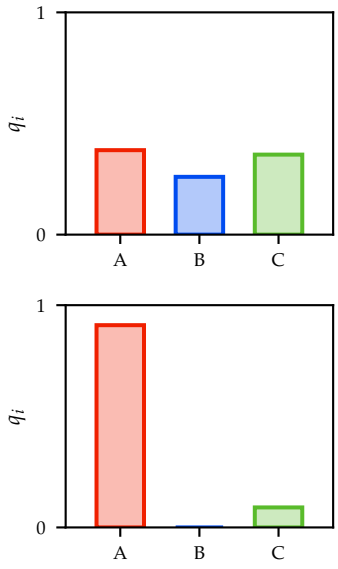


Figure 7.9: Illustration of temperature scaling for a classifier with three classes. On the top, we have a prediction with a high temperature, yielding a very uncertain prediction (in favor of class A). Below, we have a prediction with a low temperature, yielding a prediction that is strongly in favor of class A. Note that the ranking ($A \succ C \succ B$) is preserved.

PART II

Sequential Decision-Making

8

Active Learning

By now, we have seen that probabilistic machine learning is very useful for estimating the uncertainty in our models (epistemic uncertainty) and in the data (aleatoric uncertainty). We have been focusing on the setting of supervised learning where we are given a set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ of labeled data, yet we often encounter settings where we have only little data and acquiring new data is costly.

In this chapter — and in the following chapter on Bayesian optimization — we will discuss how one can use uncertainty to effectively collect more data. In other words, we want to figure out where in the domain we should sample to obtain the most useful information. Throughout most of this chapter, we focus on the most common way of quantifying "useful information", namely the expected reduction in entropy which is also called the *mutual information*. In section 8.5, we briefly discuss other ways of quantifying usefulness of information.

8.1 Conditional Entropy

We begin by introducing the notion of conditional entropy. Recall that the entropy $\mathrm{H}[\mathbf{X}]$ of a random vector $\mathbf{X}$ can be interpreted as our average surprise when observing realizations $\boldsymbol{x} \sim \mathbf{X}$. Thus, entropy can be considered as a quantification of the uncertainty about a random vector (or equivalently, its distribution).[1]

[1] We discussed entropy extensively in section 5.4.

A natural extension is to consider the entropy of $\mathbf{X}$ given the occurrence of an event corresponding to another random variable (e.g., $\mathbf{Y} = \boldsymbol{y}$ for a random vector $\mathbf{Y}$),

$$\mathrm{H}[\mathbf{X} \mid \mathbf{Y} = \boldsymbol{y}] \doteq \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x} \mid \boldsymbol{y})}[-\log p(\boldsymbol{x} \mid \boldsymbol{y})]. \tag{8.1}$$

Instead of averaging over the surprise of samples from the distribution $p(\boldsymbol{x})$ (like the entropy $\mathrm{H}[\mathbf{X}]$), this quantity simply averages over the surprise of samples from the conditional distribution $p(\boldsymbol{x} \mid \boldsymbol{y})$.

Definition 8.1 (Conditional entropy)**.** The *conditional entropy* of a random vector $\mathbf{X}$ given the random vector $\mathbf{Y}$ is defined as

$$\begin{aligned} \mathrm{H}[\mathbf{X} \mid \mathbf{Y}] &\doteq \mathbb{E}_{\boldsymbol{y} \sim p(\boldsymbol{y})}[\mathrm{H}[\mathbf{X} \mid \mathbf{Y} = \boldsymbol{y}]] && (8.2) \\ &= \mathbb{E}_{(\boldsymbol{x}, \boldsymbol{y}) \sim p(\boldsymbol{x}, \boldsymbol{y})}[-\log p(\boldsymbol{x} \mid \boldsymbol{y})]. && (8.3) \end{aligned}$$

Intuitively, the conditional entropy of $\mathbf{X}$ given $\mathbf{Y}$ describes our average surprise about realizations of $\mathbf{X}$ given a particular realization of $\mathbf{Y}$, averaged over all such possible realizations of $\mathbf{Y}$. In other words, conditional entropy corresponds to the expected remaining uncertainty in $\mathbf{X}$ after we observe $\mathbf{Y}$. Note that, in general, $\mathrm{H}[\mathbf{X} \mid \mathbf{Y}] \neq \mathrm{H}[\mathbf{Y} \mid \mathbf{X}]$.

It is crucial to stress the difference between $\mathrm{H}[\mathbf{X} \mid \mathbf{Y} = \boldsymbol{y}]$ and the conditional entropy $\mathrm{H}[\mathbf{X} \mid \mathbf{Y}]$. The former simply corresponds to a Bayesian update of our uncertainty in $\mathbf{X}$ after we have observed the realization $\boldsymbol{y} \sim \mathbf{Y}$. In contrast, conditional entropy *predicts* how much uncertainty will remain about $\mathbf{X}$ (in expectation) after we *will* observe $\mathbf{Y}$.

Definition 8.2 (Joint entropy)**.** One can also define the *joint entropy* of random vectors $\mathbf{X}$ and $\mathbf{Y}$,

$$\mathrm{H}[\mathbf{X}, \mathbf{Y}] \doteq \mathbb{E}_{(\boldsymbol{x}, \boldsymbol{y}) \sim p(\boldsymbol{x}, \boldsymbol{y})}[-\log p(\boldsymbol{x}, \boldsymbol{y})], \tag{8.4}$$

as the combined uncertainty about $\mathbf{X}$ and $\mathbf{Y}$. Observe that joint entropy is symmetric.

This gives the *chain rule for entropy*,

$$\begin{aligned} \mathrm{H}[\mathbf{X}, \mathbf{Y}] &= \mathrm{H}[\mathbf{Y}] + \mathrm{H}[\mathbf{X} \mid \mathbf{Y}] && (8.5) \\ &= \mathrm{H}[\mathbf{X}] + \mathrm{H}[\mathbf{Y} \mid \mathbf{X}]. && (8.6) \end{aligned}$$

(8.5): using the product rule (1.14) and the definition of conditional entropy (8.2)

(8.6): using symmetry of joint entropy

That is, the joint entropy of $\mathbf{X}$ and $\mathbf{Y}$ is given by the uncertainty about $\mathbf{X}$ and the additional uncertainty about $\mathbf{Y}$ given $\mathbf{X}$. Moreover, this also yields *Bayes' rule for entropy*,

$$\mathrm{H}[\mathbf{X} \mid \mathbf{Y}] = \mathrm{H}[\mathbf{Y} \mid \mathbf{X}] + \mathrm{H}[\mathbf{X}] - \mathrm{H}[\mathbf{Y}]. \tag{8.7}$$

using the chain rule for entropy (8.5) twice

A very intuitive property of entropy is its monotonicity: when conditioning on additional observations the entropy can never increase,

$$\mathrm{H}[\mathbf{X} \mid \mathbf{Y}] \leq \mathrm{H}[\mathbf{X}]. \tag{8.8}$$

Colloquially, this property is also called the "*information never hurts*" principle. We will derive a proof in the following section.

8.2 Mutual Information

Recall that our fundamental objective is to reduce entropy, as this corresponds to reduced uncertainty in the variables, which we want to predict. Thus, we are interested in how much information we "gain" about the random vector $\mathbf{X}$ by choosing to observe a random vector $\mathbf{Y}$. By our interpretation of conditional entropy from the previous section, this is described by the following quantity.

Definition 8.3 (Mutual information, MI)**.** The *mutual information* of $\mathbf{X}$ and $\mathbf{Y}$ (also known as the *information gain*) is defined as

$$\begin{aligned} \mathrm{I}(\mathbf{X};\mathbf{Y}) &\doteq \mathrm{H}[\mathbf{X}] - \mathrm{H}[\mathbf{X} \mid \mathbf{Y}] && (8.9)\\ &= \mathrm{H}[\mathbf{X}] + \mathrm{H}[\mathbf{Y}] - \mathrm{H}[\mathbf{X},\mathbf{Y}]. && (8.10) \end{aligned}$$

In words, we subtract the uncertainty left about $\mathbf{X}$ after observing $\mathbf{Y}$ from our initial uncertainty about $\mathbf{X}$. This measures the reduction in our uncertainty in $\mathbf{X}$ (as measured by entropy) upon observing $\mathbf{Y}$. Unlike conditional entropy, it follows from the definition that mutual information is symmetric. That is,

$$\mathrm{I}(\mathbf{X};\mathbf{Y}) = \mathrm{I}(\mathbf{Y};\mathbf{X}). \tag{8.11}$$

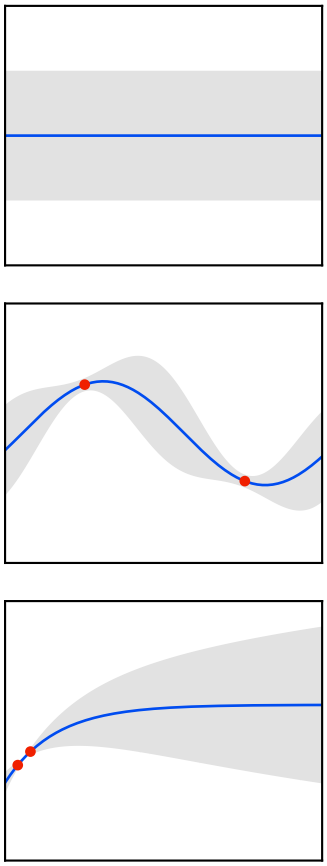


Figure 8.1: Information gain. The first graph shows the prior. The second graph shows a selection of samples with a large information gain (large reduction in uncertainty). The third graph shows a selection of samples with a small information gain (small reduction in uncertainty).

Exercise 8.4: Mutual information and KL divergence

Show that by expanding the definition of mutual information,

$$\begin{aligned} \mathrm{I}(\mathbf{X};\mathbf{Y}) &= \mathbb{E}_{(x,y)\sim p}\left[\log \frac{p(x,y)}{p(x)p(y)}\right] \\ &= \mathrm{KL}(p(x,y)\|p(x)p(y)) && (8.12)\\ &= \mathbb{E}_{y\sim p}[\mathrm{KL}(p(x \mid y)\|p(x))], && (8.13) \end{aligned}$$

where $p(x,y)$ denotes the joint distribution and $p(x), p(y)$ denote the marginal distributions of $\mathbf{X}$ and $\mathbf{Y}$.

▷ *Solution*

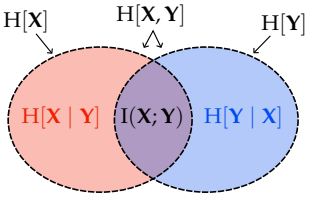


Figure 8.2: Relationship between mutual information and entropy, expressed as a Venn diagram.

Thus, the mutual information between $\mathbf{X}$ and $\mathbf{Y}$ can be understood as the approximation error (or information loss) when assuming that $\mathbf{X}$ and $\mathbf{Y}$ are independent.

In particular, using Gibbs' inequality (cf. exercise 5.19), this relationship shows that $\mathrm{I}(\mathbf{X};\mathbf{Y}) \geq 0$ with equality when $\mathbf{X}$ and $\mathbf{Y}$ are independent, and also proves the *information never hurts* principle (8.8) as

$$0 \leq \mathrm{I}(\mathbf{X};\mathbf{Y}) = \mathrm{H}[\mathbf{X}] - \mathrm{H}[\mathbf{X} \mid \mathbf{Y}]. \tag{8.14}$$

Example 8.5: Mutual information of Gaussians

Given the Gaussian random vector $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and the noisy observation $\mathbf{Y} = \mathbf{X} + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma_n^2 \boldsymbol{I})$, we want to find the information gain of $\mathbf{X}$ when observing $\mathbf{Y}$. Using our definitions from this chapter, we obtain

$$
\begin{aligned}
\mathrm{I}(\mathbf{X}; \mathbf{Y}) &= \mathrm{I}(\mathbf{Y}; \mathbf{X}) && \text{using symmetry (8.11)} \\
&= \mathrm{H}[\mathbf{Y}] - \mathrm{H}[\mathbf{Y} \mid \mathbf{X}] && \text{by mutual information (8.9)} \\
&= \mathrm{H}[\mathbf{Y}] - \mathrm{H}[\boldsymbol{\epsilon}] && \text{given } \mathbf{X}\text{, the only randomness in } \mathbf{Y} \text{ originates from } \boldsymbol{\epsilon} \\
&= \frac{1}{2} \log\Big((2\pi e)^d \det\Big(\boldsymbol{\Sigma} + \sigma_n^2 \boldsymbol{I} \Big) \Big) - \frac{1}{2} \log\Big((2\pi e)^d \det\Big(\sigma_n^2 \boldsymbol{I} \Big) \Big) && \text{using the entropy of Gaussians (5.36)} \\
&= \frac{1}{2} \log \frac{\det(\boldsymbol{\Sigma} + \sigma_n^2 \boldsymbol{I})}{\det(\sigma_n^2 \boldsymbol{I})} \\
&= \frac{1}{2} \log \det\Big(\boldsymbol{I} + \sigma_n^{-2} \boldsymbol{\Sigma} \Big). && (8.15)
\end{aligned}
$$

Intuitively, the larger the noise σ_n^2 in relation to the covariance of $\mathbf{X}$, the smaller the information gain.

8.2.1 *Data Processing Inequality and Sufficient Statistics*

Definition 8.6 (Conditional mutual information)**.** The *conditional mutual information* of $\mathbf{X}$ and $\mathbf{Y}$ given $\mathbf{Z}$ is defined as

$$
\begin{aligned}
\mathrm{I}(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z}) &\doteq \mathrm{H}[\mathbf{X} \mid \mathbf{Z}] - \mathrm{H}[\mathbf{X} \mid \mathbf{Y}, \mathbf{Z}]. && (8.16) \\
&= \mathrm{H}[\mathbf{X}, \mathbf{Z}] + \mathrm{H}[\mathbf{Y}, \mathbf{Z}] - \mathrm{H}[\mathbf{Z}] - \mathrm{H}[\mathbf{X}, \mathbf{Y}, \mathbf{Z}] && (8.17) \quad \text{using the relationship of joint and conditional entropy (8.5)} \\
&= \mathrm{I}(\mathbf{X}; \mathbf{Y}, \mathbf{Z}) - \mathrm{I}(\mathbf{X}; \mathbf{Z}). && (8.18)
\end{aligned}
$$

Thus, the conditional mutual information corresponds to the reduction of uncertainty in $\mathbf{X}$ when observing $\mathbf{Y}$, given we already observed $\mathbf{Z}$. It also follows that conditional mutual information is symmetric:

$$
\mathrm{I}(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z}) = \mathrm{I}(\mathbf{Y}; \mathbf{X} \mid \mathbf{Z}) \,. \tag{8.19}
$$

Exercise 8.7: Non-monotonicity of cond. mutual information

We have seen in this chapter that entropy is monotonically decreasing as we condition on new information, and called this the "information never hurts" principle (8.8). However, we will see in this exercise that the same does not hold for mutual information, that is, information about a random vector $\mathbf{Z}$ may reduce the mutual information between random vectors $\mathbf{X}$ and $\mathbf{Y}$.

1. Show that if $\mathbf{X} \perp \mathbf{Z}$ then $\mathrm{I}(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z}) \geq \mathrm{I}(\mathbf{X}; \mathbf{Y})$.

2. Show that if $\mathbf{X} \perp \mathbf{Z} \mid \mathbf{Y}$ then $\mathrm{I}(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z}) \leq \mathrm{I}(\mathbf{X}; \mathbf{Y})$.
3. Note that the condition $\mathbf{X} \perp \mathbf{Z} \mid \mathbf{Y}$ is the Markov property, namely, $\{\mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$ form a Markov chain with graphical model $\mathbf{X} \to \mathbf{Y} \to \mathbf{Z}$. This situation often occurs when data is processed sequentially. Prove

$$\mathrm{I}(\mathbf{X}; \mathbf{Z}) \leq \mathrm{I}(\mathbf{X}; \mathbf{Y}). \tag{8.20}$$

   which is also known as the *data processing inequality*, and which says that processing cannot increase the information contained in a signal.

▷ *Solution*

The data processing inequality allows us to formalize a concept which we have seen multiple times already, namely the notion of a sufficient statistic. Consider the Markov chain $\boldsymbol{\lambda} \to \mathbf{X} \to \boldsymbol{s}(\mathbf{X})$, for example, $\boldsymbol{\lambda}$ may be parameters of the distribution of $\mathbf{X}$. By the data processing inequality (8.20), $\mathrm{I}(\boldsymbol{\lambda}; \boldsymbol{s}(\mathbf{X})) \leq \mathrm{I}(\boldsymbol{\lambda}; \mathbf{X})$. If the data processing inequality is satisfied with equality then $\boldsymbol{s}(\mathbf{X})$ is called a *sufficient statistic* of $\mathbf{X}$ for the inference of $\boldsymbol{\lambda}$.

8.2.2 Synergy and Redundancy

To understand the behavior of mutual information under conditioning, it is helpful to consider the *interaction information*

$$\mathrm{I}(\mathbf{X}; \mathbf{Y}; \mathbf{Z}) \doteq \mathrm{I}(\mathbf{X}; \mathbf{Y}) - \mathrm{I}(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z}). \tag{8.21}$$

If the interaction is positive then some information about $\mathbf{X}$ that is provided by $\mathbf{Y}$ is also provided by $\mathbf{Z}$ (i.e., conditioning on $\mathbf{Z}$ decreases MI between $\mathbf{X}$ and $\mathbf{Y}$), and we say that there is *redundancy* between $\mathbf{Y}$ and $\mathbf{Z}$ (with respect to $\mathbf{X}$). Conversely, if the interaction is negative then learning about $\mathbf{Z}$ increases what can be learned from $\mathbf{Y}$ about $\mathbf{X}$, and we say that there is *synergy* between between $\mathbf{Y}$ and $\mathbf{Z}$.

Exercise 8.8: Interaction information

1. Show that interaction information is symmetric.
2. Let $X_1, X_2 \sim \mathrm{Bern}(p)$ for some $p \in (0, 1)$ and independent. We denote by $Y \doteq X_1 \oplus X_2$ their XOR. Compute $\mathrm{I}(Y; X_1; X_2)$.

▷ *Solution*

Readings

Read sections 2.2 through 2.6 of "Elements of information theory" (Cover and Thomas, 2006) for additional background on entropy, mutual information and the KL-divergence.

8.2.3 *Mutual Information as Utility Function*

Following our introduction of mutual information, it is natural to answer the question "where should I collect data?" by saying "wherever mutual information is maximized". More concretely, assume we are given a set $\mathcal{X}$ of possible observations of f, where $y_{\boldsymbol{x}}$ denotes a single such observation at $\boldsymbol{x} \in \mathcal{X}$,

$$y_{\boldsymbol{x}} \doteq f_{\boldsymbol{x}} + \epsilon_{\boldsymbol{x}}, \tag{8.22}$$

$f_{\boldsymbol{x}} \doteq f(\boldsymbol{x})$, and $\epsilon_{\boldsymbol{x}}$ is some zero-mean Gaussian noise. For a set of observations $S = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$, we can write $\boldsymbol{y}_S = \boldsymbol{f}_S + \boldsymbol{\epsilon}$ where

$$\boldsymbol{y}_S \doteq \begin{bmatrix} y_{\boldsymbol{x}_1} \\ \vdots \\ y_{\boldsymbol{x}_n} \end{bmatrix}, \quad \boldsymbol{f}_S \doteq \begin{bmatrix} f_{\boldsymbol{x}_1} \\ \vdots \\ f_{\boldsymbol{x}_n} \end{bmatrix}, \quad \text{and} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma_n^2 \boldsymbol{I}).$$

Note that both $\boldsymbol{y}_S$ and $\boldsymbol{f}_S$ are random vectors. Our goal is then to find a subset $S \subseteq \mathcal{X}$ of size n maximizing the information gain between our model f and $\boldsymbol{y}_S$.

This yields the maximization objective,

$$I(S) \doteq \mathrm{I}(\boldsymbol{f}_S; \boldsymbol{y}_S) = \mathrm{H}[\boldsymbol{f}_S] - \mathrm{H}[\boldsymbol{f}_S \mid \boldsymbol{y}_S]. \tag{8.23}$$

Here, $\mathrm{H}[\boldsymbol{f}_S]$ corresponds to the uncertainty about $\boldsymbol{f}_S$ before obtaining the observations $\boldsymbol{y}_S$ and $\mathrm{H}[\boldsymbol{f}_S \mid \boldsymbol{y}_S]$ corresponds to the uncertainty about $\boldsymbol{f}_S$ after obtaining the observations $\boldsymbol{y}_S$.

Observe that picking a subset of points $S \subseteq \mathcal{X}$ from the domain $\mathcal{X}$ is a combinatorial problem. That is to say, we are optimizing a function over discrete sets. In general, such combinatorial optimization problems tend to be very difficult. It can be shown that maximizing mutual information is $\mathcal{NP}$-hard.

8.3 *Submodularity of Mutual Information*

We will look at optimizing mutual information in the following section. First, we want to introduce the notion of submodularity which is important in the analysis of discrete functions.

Definition 8.9 (Marginal gain)**.** Given a (discrete) function $F : \mathcal{P}(\mathcal{X}) \to \mathbb{R}$, the *marginal gain* of $\boldsymbol{x} \in \mathcal{X}$ given $A \subseteq \mathcal{X}$ is defined as

$$\Delta_F(\boldsymbol{x} \mid A) \doteq F(A \cup \{\boldsymbol{x}\}) - F(A). \tag{8.24}$$

Intuitively, the marginal gain describes how much "adding" the additional $\boldsymbol{x}$ to A increases the value of F.

Exercise 8.10: Marginal gain of maximizing mutual information

Show that in the context of maximizing mutual information, the marginal gain is

$$\begin{aligned} \Delta_I(\boldsymbol{x} \mid A) &= \mathrm{I}(f_{\boldsymbol{x}}; y_{\boldsymbol{x}} \mid \boldsymbol{y}_A) && (8.25)\\ &= \mathrm{H}[y_{\boldsymbol{x}} \mid \boldsymbol{y}_A] - \mathrm{H}[\epsilon_{\boldsymbol{x}}]. && (8.26) \end{aligned}$$

▷ *Solution*

That is, when maximizing mutual information, the marginal gain corresponds to the difference between the uncertainty after observing y_A and the entropy of the noise $\mathrm{H}[\epsilon_{\boldsymbol{x}}]$. Altogether, the marginal gain represents the reduction in uncertainty by observing $\{\boldsymbol{x}\}$.

Definition 8.11 (Submodularity)**.** A (discrete) function $F : \mathcal{P}(\mathcal{X}) \to \mathbb{R}$ is *submodular* iff for any $\boldsymbol{x} \in \mathcal{X}$ and any $A \subseteq B \subseteq \mathcal{X}$ it satisfies

$$F(A \cup \{\boldsymbol{x}\}) - F(A) \geq F(B \cup \{\boldsymbol{x}\}) - F(B). \tag{8.27}$$

Equivalently, using our definition of marginal gain, we have that F is submodular iff for any $\boldsymbol{x} \in \mathcal{X}$ and any $A \subseteq B \subseteq \mathcal{X}$,

$$\Delta_F(\boldsymbol{x} \mid A) \geq \Delta_F(\boldsymbol{x} \mid B). \tag{8.28}$$

That is, "adding" $\boldsymbol{x}$ to the smaller set A yields more marginal gain than adding $\boldsymbol{x}$ to the larger set B. In other words, the function F has "diminishing returns". In this way, submodularity can be interpreted as a notion of "concavity" for discrete functions.

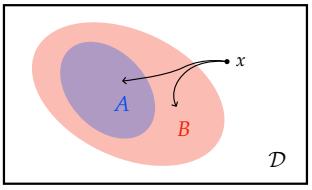


Figure 8.3: Monotone submodularity. The effect of "adding" $\boldsymbol{x}$ to the smaller set A is larger than the effect of adding $\boldsymbol{x}$ to the larger set B.

Definition 8.12 (Monotone submodularity)**.** A function $F : \mathcal{P}(\mathcal{X}) \to \mathbb{R}$ is called *monotone* iff for any $A \subseteq B \subseteq \mathcal{X}$ it satisfies

$$F(A) \leq F(B). \tag{8.29}$$

If F is also submodular, then F is called *monotone submodular*.

Theorem 8.13. *The objective I is monotone submodular.*

Proof. We fix arbitrary subsets $A \subseteq B \subseteq \mathcal{X}$ and any $\boldsymbol{x} \in \mathcal{X}$. We have,

$$I \text{ is submodular} \iff \Delta_I(\boldsymbol{x} \mid A) \geq \Delta_I(\boldsymbol{x} \mid B)$$

by submodularity in terms of marginal gain (8.28)

$$\begin{aligned}
&\iff && \mathrm{H}[y_x \mid \boldsymbol{y}_A] - \mathrm{H}[\epsilon_x] \geq \mathrm{H}[y_x \mid \boldsymbol{y}_B] - \mathrm{H}[\epsilon_x] && \text{using eq. (8.26)}\\
&\iff && \mathrm{H}[y_x \mid \boldsymbol{y}_A] \geq \mathrm{H}[y_x \mid \boldsymbol{y}_B]. && \mathrm{H}[\epsilon_x] \text{ cancels}
\end{aligned}$$

Due to the "information never hurts" principle (8.8) of entropy and as $A \subseteq B$, this is always true. Moreover,

$$\begin{aligned}
I \text{ is monotone} &\iff && I(A) \leq I(B) && \text{by the definition of monotinicity (8.29)}\\
&\iff && \mathrm{I}(\boldsymbol{f}_A; \boldsymbol{y}_A) \leq \mathrm{I}(\boldsymbol{f}_B; \boldsymbol{y}_B) && \text{using the definition of } I \text{ (8.23)}\\
&\iff && \mathrm{I}(\boldsymbol{f}_B; \boldsymbol{y}_A) \leq \mathrm{I}(\boldsymbol{f}_B; \boldsymbol{y}_B) && \text{using } \mathrm{I}(\boldsymbol{f}_B; \boldsymbol{y}_A) = \mathrm{I}(\boldsymbol{f}_A; \boldsymbol{y}_A) \text{ as } \boldsymbol{y}_A \perp \boldsymbol{f}_B \mid \boldsymbol{f}_A\\
&\iff && \mathrm{H}[\boldsymbol{f}_B] - \mathrm{H}[\boldsymbol{f}_B \mid \boldsymbol{y}_A] \leq \mathrm{H}[\boldsymbol{f}_B] - \mathrm{H}[\boldsymbol{f}_B \mid \boldsymbol{y}_B] && \text{using the definition of MI (8.9)}\\
&\iff && \mathrm{H}[\boldsymbol{f}_B \mid \boldsymbol{y}_A] \geq \mathrm{H}[\boldsymbol{f}_B \mid \boldsymbol{y}_B], && \mathrm{H}[\boldsymbol{f}_B] \text{ cancels}
\end{aligned}$$

which is also satisfied due to the "information never hurts" principle (8.8). □

Exercise 8.14: Submodularity means no synergy

The submodularity of I can be interpreted from the perspective of information theory. It turns out that submodularity is equivalent to the absence of synergy between observations. To this end, show that for all $A \subseteq B$,

$$\mathrm{I}(f_x; y_x; \boldsymbol{y}_{B \setminus A} \mid \boldsymbol{y}_A) \geq 0. \tag{8.30}$$

▷ *Solution*

8.4 Maximizing Mutual Information

As we cannot efficiently pick a set $S \subseteq \mathcal{X}$ to maximize mutual information, a natural approach is to maximize mutual information greedily. That is, we pick the locations $\boldsymbol{x}_1$ through $\boldsymbol{x}_n$ individually by greedily finding the location with the maximal mutual information. The following general result for monotone submodular function maximization shows that this greedy approach provides a good approximation.

Theorem 8.15 (Greedy submodular function maximization)**.** *If F is monotone submodular, then greedily maximizing F provides a $(1 - 1/e)$-approximation,*[2]

[2] $1 - 1/e \approx 0.632$

$$F(S_n) \geq \left(1 - \frac{1}{e}\right) \max_{\substack{S \subseteq \mathcal{X} \\ |S| = n}} F(S). \tag{8.31}$$

Proof. Fix any $n \geq 1$. Let $S^\star \in \arg\max\{F(S) \mid S \in \mathcal{X}, |S| \leq n\}$. We can assume $|S^\star| = n$ due to the monotonicity (8.29) of F. We write,

$\{\boldsymbol{x}_1^\star, \ldots, \boldsymbol{x}_n^\star\} \doteq S^\star$. We have,

$$
\begin{aligned}
F(S^\star) &\leq F(S^\star \cup S_t) && \text{using monotonicity (8.29)} \\
&= F(S_t) + \sum_{i=1}^{n} \Delta_F(\boldsymbol{x}_i^\star \mid S_t \cup \{\boldsymbol{x}_1^\star, \ldots, \boldsymbol{x}_{i-1}^\star\}) && \text{using the definition of marginal gain (8.24)} \\
&\leq F(S_t) + \sum_{\boldsymbol{x}^\star \in S^\star} \Delta_F(\boldsymbol{x}^\star \mid S_t) && \text{using submodularity (8.28)} \\
&\leq F(S_t) + n(F(S_{t+1}) - F(S_t)). && \text{using that } S_{t+1} = S_t \cup \{\boldsymbol{x}\} \text{ is chosen such that } \Delta_F(\boldsymbol{x} \mid S_t) \text{ is maximized (8.32)}
\end{aligned}
$$

Let $\delta_t \doteq F(S^\star) - F(S_t)$. Then,

$$\delta_t = F(S^\star) - F(S_t) \leq n(F(S_{t+1}) - F(S_t)) = n(\delta_t - \delta_{t+1}).$$

This implies $\delta_{t+1} \leq (1 - 1/n)\delta_t$ and $\delta_n \leq (1 - 1/n)^n \delta_0 \leq \delta_0/e$, using the well-known inequality $1 - x \leq e^{-x}$ for all $x \in \mathbb{R}$.

Finally, observe that $\delta_0 = F(S^\star) - F(\emptyset) \leq F(S^\star)$ due to the non-negativity of F. We obtain,

$$\delta_n = F(S^\star) - F(S_n) \leq \frac{\delta_0}{e} \leq \frac{F(S^\star)}{e}.$$

Rearranging the terms yields the theorem. □

Readings

The original proof of greedy maximization for submodular functions was given by "An analysis of approximations for maximizing submodular set functions" (Nemhauser et al., 1978).

For more background on maximizing submodular functions, refer to "Submodular function maximization" (Krause and Golovin, 2014).

8.4.1 *Uncertainty Sampling*

When maximizing mutual information, at time t when we have already picked $S_t = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t\}$, we need to solve the following optimization problem,

$$\boldsymbol{x}_{t+1} \doteq \arg\max_{\boldsymbol{x} \in \mathcal{X}} \Delta_I(\boldsymbol{x} \mid S_t) \tag{8.32}$$

$$= \arg\max_{\boldsymbol{x} \in \mathcal{X}} \mathrm{I}(f_{\boldsymbol{x}}; y_{\boldsymbol{x}} \mid \boldsymbol{y}_{S_t}). \tag{8.33}$$ using eq. (8.25)

Note that $f_{\boldsymbol{x}}$ and $y_{\boldsymbol{x}}$ are univariate random variables. Thus, using our formula for the mutual information of conditional linear Gaussians (8.15), we can simplify to,

$$= \arg\max_{\boldsymbol{x} \in \mathcal{X}} \frac{1}{2} \log\left(1 + \frac{\sigma_t^2(\boldsymbol{x})}{\sigma_n^2}\right) \tag{8.34}$$

where $\sigma_t^2(\boldsymbol{x})$ is the (remaining) variance at $\boldsymbol{x}$ after observing S_t. Assuming the label noise is independent of $\boldsymbol{x}$ (i.e., homoscedastic),

$$= \arg\max_{\boldsymbol{x} \in \mathcal{X}} \sigma_t^2(\boldsymbol{x}). \tag{8.35}$$

Therefore, if f is modeled by a Gaussian and we assume homoscedastic noise, greedily maximizing mutual information corresponds to simply picking the point $\boldsymbol{x}$ with the largest variance. This algorithm is also called *uncertainty sampling*.

8.4.2 Heteroscedastic Noise

Uncertainty sampling is clearly problematic if the noise is heteroscedastic. If there are a particular set of inputs with a large aleatoric uncertainty dominating the epistemic uncertainty, uncertainty sampling will continuously choose those points even though the epistemic uncertainty will not be reduced substantially. See fig. 8.4, for an example.

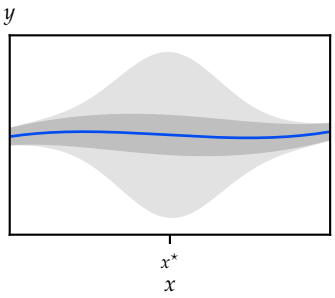


Figure 8.4: Uncertainty sampling with heteroscedastic noise. The epistemic uncertainty of the model is shown in a dark gray. The aleatoric uncertainty of the data is shown in a light gray. Uncertainty sampling would repeatedly pick points around $x^\star$ as they maximize the epistemic uncertainty, even though the aleatoric uncertainty at $x^\star$ is much larger than at the boundary.

Looking at eq. (8.34) suggests a natural fix. Instead of only considering the epistemic uncertainty $\sigma_t^2(\boldsymbol{x})$, we can also consider the aleatoric uncertainty $\sigma_n^2(\boldsymbol{x})$ by modeling heteroscedastic noise, yielding

$$\boldsymbol{x}_{t+1} \doteq \arg\max_{\boldsymbol{x} \in \mathcal{X}} \frac{1}{2} \log\left(1 + \frac{\sigma_t^2(\boldsymbol{x})}{\sigma_n^2(\boldsymbol{x})}\right) = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \frac{\sigma_t^2(\boldsymbol{x})}{\sigma_n^2(\boldsymbol{x})}. \tag{8.36}$$

Thus, we choose locations that trade large epistemic uncertainty with large aleatoric uncertainty. Ideally, we find a location where the epistemic uncertainty is large, and the aleatoric uncertainty is low, which promises a significant reduction of uncertainty around this location.

8.4.3 Classification

While we focused on regression, one can apply active learning also for other settings, such as (probabilistic) classification. In this setting, for any input $\boldsymbol{x}$, a model produces a categorical distribution over labels $y_{\boldsymbol{x}}$.[3] Here, uncertainty sampling corresponds to selecting samples that maximize the entropy of the predicted label $y_{\boldsymbol{x}}$,

[3] see section 1.2

$$\boldsymbol{x}_{t+1} \doteq \arg\max_{\boldsymbol{x} \in \mathcal{X}} \mathrm{H}[y_{\boldsymbol{x}} \mid \boldsymbol{x}_{1:t}, y_{1:t}]. \tag{8.37}$$

The entropy of a categorical distribution is simply a finite sum of weighted surprise terms.

This approach generally leads to sampling points that are close to the decision boundary. Often, the uncertainty is mainly dominated by label noise rather than epistemic uncertainty, and hence, we do not learn much from our observations. This is a similar problem to the one we

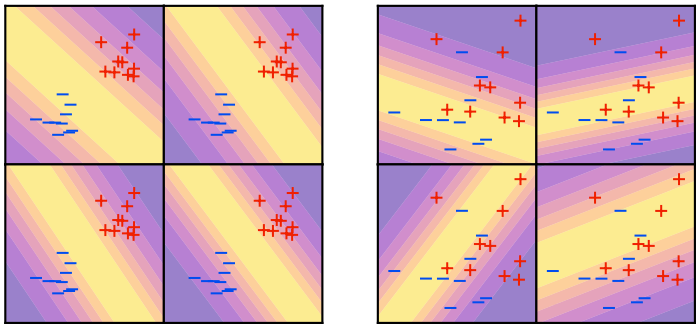


Figure 8.5: Uncertainty sampling in classification. The area with high uncertainty (as measured by entropy) is highlighted in yellow. The shown figures display each a sequence of model updates, each after one new observation. In the left figure, the classes are well-separated and uncertainty is dominated by epistemic uncertainty, whereas in the right figure the uncertainty is dominated by noise. In the latter case, if we mostly choose points $\boldsymbol{x}$ in the area of highest uncertainty (i.e., close to the decision boundary) to make observations, the label noise results in frequently changing models.

encountered with uncertainty sampling in the setting of heteroscedastic noise.

This naturally suggests distinguishing between the aleatoric and epistemic uncertainty of the model f (parameterized by $\boldsymbol{\theta}$). To this end, mutual information can be used similarly as we have done with uncertainty sampling for regression,

$$\begin{aligned}
\boldsymbol{x}_{t+1} &\doteq \arg\max_{\boldsymbol{x}\in\mathcal{X}} \mathrm{I}(\boldsymbol{\theta}; y_{\boldsymbol{x}} \mid \boldsymbol{x}_{1:t}, y_{1:t}) && (8.38)\\
&= \arg\max_{\boldsymbol{x}\in\mathcal{X}} \mathrm{I}(y_{\boldsymbol{x}}; \boldsymbol{\theta} \mid \boldsymbol{x}_{1:t}, y_{1:t}) && \text{using symmetry (8.11)}\\
&= \arg\max_{\boldsymbol{x}\in\mathcal{X}} \mathrm{H}[y_{\boldsymbol{x}} \mid \boldsymbol{x}_{1:t}, y_{1:t}] - \mathrm{H}[y_{\boldsymbol{x}} \mid \boldsymbol{\theta}, \boldsymbol{x}_{1:t}, y_{1:t}] && \text{using the definition of mutual information (8.9)}\\
&= \arg\max_{\boldsymbol{x}\in\mathcal{X}} \mathrm{H}[y_{\boldsymbol{x}} \mid \boldsymbol{x}_{1:t}, y_{1:t}] - \mathbb{E}_{\boldsymbol{\theta}|\boldsymbol{x}_{1:t},y_{1:t}}[\mathrm{H}[y_{\boldsymbol{x}} \mid \boldsymbol{\theta}, \boldsymbol{x}_{1:t}, y_{1:t}]] && (8.39)\\
&\qquad \text{using the definition of conditional entropy (8.2)}\\
&= \arg\max_{\boldsymbol{x}\in\mathcal{X}} \underbrace{\mathrm{H}[y_{\boldsymbol{x}} \mid \boldsymbol{x}_{1:t}, y_{1:t}]}_{\substack{\text{entropy of}\\ \text{predictive posterior}}} - \mathbb{E}_{\boldsymbol{\theta}|\boldsymbol{x}_{1:t},y_{1:t}} \underbrace{\mathrm{H}[y_{\boldsymbol{x}} \mid \boldsymbol{\theta}]}_{\substack{\text{entropy of}\\ \text{likelihood}}}. && (8.40)\\
&\qquad \text{using the definition of entropy (5.32) and assuming } y_{\boldsymbol{x}} \perp \boldsymbol{x}_{1:t}, y_{1:t} \mid \boldsymbol{\theta}
\end{aligned}$$

The first term measures the entropy of the averaged prediction while the second term measures the average entropy of predictions. Thus, the first term looks for points where the average prediction is not confident. In contrast, the second term penalizes points where many of the sampled models are not confident about their prediction, and thus looks for points where the models are confident in their predictions. This identifies those points $\boldsymbol{x}$ where the models *disagree* about the label $y_{\boldsymbol{x}}$ (that is, each model is "confident" but the models predict different labels). For this reason, this approach is known as *Bayesian active learning by disagreement* (BALD).

Note that the second term of the difference acts as a regularizer when compared to eq. (8.37). The second term mirrors our description of aleatoric uncertainty from section 2.3. Recall that we interpreted aleatoric uncertainty as the average uncertainty for all models. Crucially,

here we use entropy to "measure" uncertainty, whereas previously we have been using variance. Therefore, intuitively, eq. (8.39) subtracts the aleatoric uncertainty from the total uncertainty about the label.

Observe that both terms require approximate forms of the posterior distribution. In chapters 5 and 6, we have seen various approaches from variational inference and MCMC methods which can be used here. The first term can be approximated by the predictive distribution of an approximated posterior which is obtained, for example, using variational inference. The nested entropy of the second term is typically easy to compute, as it corresponds to the entropy of the (discrete) likelihood distribution $p(y \mid \boldsymbol{x}, \boldsymbol{\theta})$ of the model $\boldsymbol{\theta}$. The outer expectation of the second term may be approximated using (approximate) samples from the posterior distribution obtained via variational inference, MCMC, or some other method.

Optional Readings

- Gal, Islam, and Ghahramani (2017).
  Deep bayesian active learning with image data.

8.5 Beyond Mutual Information: Experimental Design

The problem of identifying which experiments to conduct in order to maximize learning is studied extensively in the field of *experimental design*. A set of observations S is commonly called a *design*. In the presence of a prior and likelihood, and a different possible posterior distribution for each design S, the field is also called *Bayesian experimental design* (Chaloner and Verdinelli, 1995).

As we highlighted in the beginning of this chapter, how we measure the utility (i.e., the informativeness) of a design S is crucial. Such a measure is called a *design criterion* and a design which is optimal with respect to a design criterion is called an *optimal design*. In the following, we give an overview of the most common design criteria. We continue to focus on the setting of Gaussian likelihood and prior, and for simplicity we assume that the domain $\mathcal{X}$ is finite, so our objective is to "learn" $\boldsymbol{f} \doteq \boldsymbol{f}_{\mathcal{X}}$.[4]

[4] If the domain is not finite, we can alternatively learn the parameters $\boldsymbol{\theta}$ of a parametric model f akin to BALD (cf. section 8.4.3).

- A *d-optimal design* selects S to minimize the determinant of the posterior covariance matrix which in the Gaussian case corresponds to minimizing posterior entropy (MacKay, 1992):

$$\arg\min_S \det(\mathrm{Var}[\boldsymbol{f} \mid \boldsymbol{y}_S]) = \arg\min_S \mathrm{H}[\boldsymbol{f} \mid \boldsymbol{y}_S] = \arg\max_S \mathrm{I}(\boldsymbol{f}; \boldsymbol{y}_S). \tag{8.41}$$

using the entropy of a Gaussian (5.36) and the definition of mutual information (8.9)

This is the criterion that we have been studying so far,[5] and as seen in section 8.4.1, in a regression problem it coincides with uncertainty sampling when the noise is homoscedastic. Geometrically, this criterion can be interpreted as selecting the design which leads to the largest reduction of volume of the uncertainty ellipsoid.

- A common alternative is *a-optimal design* which selects S to minimize the posterior average variance, i.e., the trace of the posterior covariance matrix:

$$\arg\min_S \operatorname{tr}(\operatorname{Var}[\boldsymbol{f} \mid \boldsymbol{y}_S]). \tag{8.42}$$

  This criterion may be more useful when the goal is to obtain a good estimate of the posterior means (i.e., make marginal predictions), and one does not care about the posterior interaction between random variables.
- Yet another alternative is *e-optimal design* which selects S to minimize the maximum eigenvalue of the posterior covariance matrix:

$$\arg\min_S \lambda_{\max}(\operatorname{Var}[\boldsymbol{f} \mid \boldsymbol{y}_S]). \tag{8.43}$$

  Geometrically, this can be interpreted as minimizing the diameter of the uncertainty ellipsoid.

Observe that these design criteria are all grounded in different "scalarizations" of the posterior covariance matrix.

[5] Note that since $\boldsymbol{f}_{\mathcal{X}\setminus S} \perp \boldsymbol{y}_S \mid \boldsymbol{f}_S$, it follows from eq. (8.18) that

$$\mathrm{I}(\boldsymbol{f}; \boldsymbol{y}_S) = \mathrm{I}(\boldsymbol{f}_S; \boldsymbol{y}_S) + \underbrace{\mathrm{I}(\boldsymbol{f}_{\mathcal{X}\setminus S}; \boldsymbol{y}_S \mid \boldsymbol{f}_S)}_{0}.$$

8.6 Beyond Global Learning

Active learning and experimental design are typically concerned with reducing the uncertainty about f *globally* across the domain. However, in many practical applications, we are only interested in a *local* region of the domain.

A common example is determining where f is maximized. The case where we *additionally* seek to only evaluate a limited number of suboptimal points while searching for the maximum will be discussed thoroughly in the next chapter, but for now, we only want to find the location of the maximum. In view of *multi-armed bandits* (which we will introduce in section 9.2.1), this is also known as the *best-arm identification* problem when the domain is finite.

One viable approach is to use standard design criteria that reduce uncertainty globally, however, they attempt to reduce uncertainty in areas of the domain that are already known to be suboptimal. By restricting the "points of interest", one can typically achieve a significantly better sample complexity.

8.6.1 Entropy Search

One canonical family of methods is inspired by the d-optimality criterion. Let $\boldsymbol{x}^* \doteq \arg\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$ denote the location of the maximum and select the observation that maximizes the information gain about the location of the maximum,

$$\boldsymbol{x}_{t+1} \doteq \arg\max_{\boldsymbol{x}\in\mathcal{X}} \mathrm{I}(\boldsymbol{x}^*; y_{\boldsymbol{x}} \mid \boldsymbol{x}_{1:t}, y_{1:t})\,. \tag{8.44}$$

Using symmetry of mutual information (8.11), we can write this in two different ways:

$$= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}_{y_{\boldsymbol{x}}|\boldsymbol{x}_{1:t},y_{1:t}}[\mathrm{H}[\boldsymbol{x}^* \mid \boldsymbol{x}_{1:t}, y_{1:t}, y_{\boldsymbol{x}}]] \tag{8.45}$$

$$= \arg\max_{\boldsymbol{x}\in\mathcal{X}} \mathrm{H}[y_{\boldsymbol{x}} \mid \boldsymbol{x}_{1:t}, y_{1:t}] - \mathbb{E}_{\boldsymbol{x}^*|\boldsymbol{x}_{1:t},y_{1:t}}[\mathrm{H}[y_{\boldsymbol{x}} \mid \boldsymbol{x}_{1:t}, y_{1:t}, \boldsymbol{x}^*]] \tag{8.46}$$

where to simplify the expressions, we used the definition of mutual information (8.9) and the definition of conditional entropy (8.2). The algorithm in eq. (8.45) is known as *entropy search* (ES) (Hennig and Schuler, 2012), and perhaps the most natural as it selects $\boldsymbol{x}_{t+1}$ to minimize the posterior entropy of $\boldsymbol{x}^\star$. However, approximating the entropy of the high-dimensional posterior distribution over $\boldsymbol{x}^\star$ is difficult.

More amenable to approximations is the formulation from eq. (8.46) which is known as *predictive entropy search* (PES) (Hernández-Lobato et al., 2014). Observe that the derivation of PES is analogous to BALD (8.39), only that here we condition on the location of the maximum $\boldsymbol{x}^*$ rather than on the model parameters $\boldsymbol{\theta}$. The first term of eq. (8.46) is the entropy of the predictive distribution of $y_{\boldsymbol{x}}$ which we know to be

$$\mathrm{H}[y_{\boldsymbol{x}} \mid \boldsymbol{x}_{1:t}, y_{1:t}] = \frac{1}{2}\log\Big(2\pi e(\sigma_t^2(\boldsymbol{x}) + \sigma_n^2)\Big). \qquad \text{using eq. (5.35)}$$

The expectation of the second term can be approximated using Monte carlo sampling by drawing m functions from the posterior distribution and taking the maximum of each function as a sample of $\boldsymbol{x}^*$. The main computational difficulty lies in approximating the entropy of the "predictive" distribution $y_{\boldsymbol{x}} \mid \boldsymbol{x}_{1:t}, y_{1:t}, \boldsymbol{x}^*$, but unlike ES, this distribution is one-dimensional.

The dependence of ES and PES on the posterior over $\boldsymbol{x}^*$ is problematic as approximations are typically expensive to compute. An alternative approach is to measure information gain with respect to the maximum value $f^* \doteq \max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$ rather than the location of the maximum $\boldsymbol{x}^*$:

$$\boldsymbol{x}_{t+1} \doteq \arg\max_{\boldsymbol{x}\in\mathcal{X}} \mathrm{I}(f^*; y_{\boldsymbol{x}} \mid \boldsymbol{x}_{1:t}, y_{1:t}) \tag{8.47}$$

$$= \arg\max_{x \in \mathcal{X}} \mathrm{H}[y_x \mid x_{1:t}, y_{1:t}] - \mathbb{E}_{f^* \mid x_{1:t}, y_{1:t}}[\mathrm{H}[y_x \mid x_{1:t}, y_{1:t}, f^*]]. \tag{8.48}$$

using the definition of mutual information (8.9) and the definition of conditional entropy (8.2)

The *max-value entropy search* (MES) approximation ignores the observation noise in $y_x \mid x_{1:t}, y_{1:t}, f^*$ and approximates $f_x \mid x_{1:t}, y_{1:t}, f^*$ by a truncated normal, for which the entropy can be approximated well in closed form (Wang and Jegelka, 2017). Since MES ignores the observation noise, it may significantly overestimate the information gain when the signal-to-noise ratio is low. In contrast, *output-space predictive entropy search* (OPES) approximates $f_x \mid x_{1:t}, y_{1:t}, f^*$ by a Gaussian, which gives $\mathrm{H}[y_x \mid x_{1:t}, y_{1:t}, f^*] \approx \frac{1}{2} \log\left(2\pi e(\sigma_*^2(x) + \sigma_n^2)\right)$ where $\sigma_*^2(x)$ denotes the variance of the Gaussian approximation (Hoffman and Ghahramani, 2015).

It may not be immediately obvious why the maximum function value yields sufficient information to find the location of the maximum. In practice, "optimal value identification" often works quite well. Intuitively, determining the optimal value requires sufficiently exploring all approximately optimal points.

Optional Readings

More details on entropy search methods can be found in sections 8.8 and 8.9 of "Bayesian optimization" (Garnett, 2023).

For an example of a local criterion akin to a-optimality, see "Truncated variance reduction: A unified approach to bayesian optimization and level-set estimation" (Bogunovic et al., 2016).

9

Bayesian Optimization

Often, obtaining data is costly. In the previous chapter, this led us to investigate how we can optimally improve our understanding (i.e., reduce uncertainty) of the process we are trying to model. However, purely improving our understanding is often not good enough. In many cases, we want to use our improving understanding *simultaneously* to reach certain goals. This is a very common problem in artificial intelligence and will concern us for the rest of this course. One common instance of this problem is the setting of optimization.

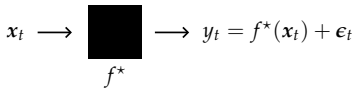


Figure 9.1: Illustration of Bayesian optimization. We pass an input x_t into the unknown function $f^\star$ to obtain noisy observations y_t.

Given some function $f^\star : \mathcal{X} \to \mathbb{R}$, suppose we want to find the

$$\arg\max_{x \in \mathcal{X}} f^\star(x). \tag{9.1}$$

Now, contrary to classical optimization, we are interested in the setting where the function $f^\star$ is unknown to us (like a "black-box"). We are only able to obtain noisy observations of $f^\star$,

$$y_t = f^\star(x_t) + \epsilon_t. \tag{9.2}$$

Moreover, these noise-perturbed evaluations are costly to obtain. We will assume that similar alternatives yield similar results,[1] which is commonly encoded by placing a Gaussian process prior on $f^\star$. This assumed correlation is fundamentally what will allow us to learn a model of $f^\star$ from relatively few samples.[2]

[1] That is, $f^\star$ is "smooth". We will be more precise in the subsequent parts of this chapter. If this were not the case, optimizing the function without evaluating it everywhere would not be possible. Fortunately, many interesting functions obey by this relatively weak assumption.

[2] There are countless examples of this problem in the "real world". Instances are

- drug trials
- chemical engineering — the development of physical products
- recommender systems
- automatic machine learning — automatic tuning of model & hyperparameters
- and many more...

9.1 Exploration-Exploitation Dilemma

In Bayesian optimization, we want to learn a model of $f^\star$ and use this model to optimize $f^\star$ simultaneously. These goals are somewhat contrary. Learning a model of $f^\star$ requires us to explore the input space while using the model to optimize $f^\star$ requires us to focus on the most promising well-explored areas. This trade-off is commonly known as the *exploration-exploitation dilemma*, whereby

- *exploration* refers to choosing points that are "informative" with respect to the unknown function. For example, points that are far away from previously observed points (i.e., have high posterior variance);[3] and
- *exploitation* refers to choosing promising points where we expect the function to have high values. For example, points that have a high posterior mean and a low posterior variance.

In other words, the exploration-exploitation dilemma refers to the challenge of learning enough to understand $f^\star$, but not learning too much to lose track of the objective — optimizing $f^\star$.

[3] We explored this topic (with strategies like uncertainty sampling) in the previous chapter.

9.2 Online Learning and Bandits

Bayesian optimization is closely related to a form of online learning. In *online learning* we are given a set of possible inputs $\mathcal{X}$ and an unknown function $f^\star : \mathcal{X} \to \mathbb{R}$. We are now asked to choose a sequence of inputs $\boldsymbol{x}_1, \dots \boldsymbol{x}_T$ online,[4] and our goal is to maximize our cumulative reward $\sum_{t=1}^{T} f^\star(\boldsymbol{x}_t)$. Depending on what we observe about $f^\star$, there are different variants of online learning. Bayesian optimization is closest to the so-called (stochastic) "bandit" setting.

[4] *Online* is best translated as "sequential". That is, we need to pick $\boldsymbol{x}_{t+1}$ based only on our prior observations $y_1, \dots, y_t$.

9.2.1 Multi-Armed Bandits

The "*multi-armed bandits*" (MAB) problem is a classical, canonical formalization of the exploration-exploitation dilemma. In the MAB problem, we are provided with k possible actions (arms) and want to maximize our reward online within the time horizon T. We do not know the reward distributions of the actions in advance, however, so we need to trade learning the reward distribution with following the most promising action. Bayesian optimization can be interpreted as a variant of the MAB problem where there can be a potentially infinite number of actions (arms), but their rewards are correlated (because of the smoothness of the Gaussian process prior).

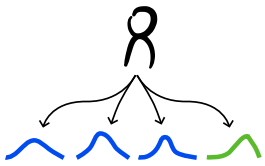


Figure 9.2: Illustration of a multi-armed bandit with four arms, each with a different reward distribution. The agent tries to identify the arm with the most beneficial reward distribution shown in green.

There exists a large body of work on this and similar problems in online decision-making. Much of this work develops theory on how to explore and exploit in the face of uncertainty. The shared prevalence of the exploration-exploitation dilemma signals a deep connection between online learning and Bayesian optimization (and — as we will later come to see — reinforcement learning). Many of the approaches which we will encounter in the context of these topics are strongly related to methods in online learning.

One of the key principles of the theory on multi-armed bandits and reinforcement learning is the principle of *"optimism in the face of un-*

certainty", which suggests that it is a good guideline to explore where we can hope for the best outcomes. We will frequently come back to this general principle in our discussion of algorithms for Bayesian optimization and reinforcement learning.

Readings

Refer to

- "Regret analysis of stochastic and nonstochastic multi-armed bandit problems" (Bubeck et al., 2012),
- "Bandit algorithms" (Lattimore and Szepesvári, 2020), or
- chapter 2 of "Reinforcement learning: An introduction" (Sutton and Barto, 2018)

for a comprehensive overview of multi-armed bandits.

9.2.2 *Regret*

The key performance metric in online learning is the regret.

Definition 9.1 (Regret)**.** The *(cumulative) regret* for a time horizon T associated with choices $\{\boldsymbol{x}_t\}_{t=1}^T$ is defined as

$$R_T \doteq \sum_{t=1}^{T} \underbrace{\left(\max_{\boldsymbol{x}} f^\star(\boldsymbol{x}) - f^\star(\boldsymbol{x}_t)\right)}_{\textit{instantaneous regret}} \tag{9.3}$$

$$= T \max_{\boldsymbol{x}} f^\star(\boldsymbol{x}) - \sum_{t=1}^{T} f^\star(\boldsymbol{x}_t). \tag{9.4}$$

The regret can be interpreted as the additive loss with respect to the *static* optimum $\max_{\boldsymbol{x}} f^\star(\boldsymbol{x})$.

The goal is to find algorithms that achieve *sublinear* regret,

$$\lim_{T\to\infty} \frac{R_T}{T} = 0. \tag{9.5}$$

Importantly, if we use an algorithm which explores *forever*, e.g., by going to a random point $\tilde{\boldsymbol{x}}$ with a constant probability ϵ in each round, then the regret will grow linearly with time. This is because the instantaneous regret is at least $\epsilon(\max_{\boldsymbol{x}} f^\star(\boldsymbol{x}) - f^\star(\tilde{\boldsymbol{x}}))$ and non-decreasing. Conversely, if we use an algorithm which *never* explores, then we might never find the static optimum, and hence, also incur constant instantaneous regret in each round, implying that regret grows linearly with time. Thus, achieving sublinear regret requires *balancing* exploration and exploitation.

Exercise 9.2: Convergence to the static optimum

Show that any algorithm achieves sublinear regret if and only if it converges to the static optimum, that is,

$$\lim_{t \to \infty} f^\star(\boldsymbol{x}_t) = \max_{\boldsymbol{x}} f^\star(\boldsymbol{x}). \tag{9.6}$$

Hint: Use that if a sequence a_n converges to a as $n \to \infty$, then we have for the sequence

$$b_n \doteq \frac{1}{n} \sum_{i=1}^{n} a_i \tag{9.7}$$

that $\lim_{n \to \infty} b_n = a$. This is also known as the Cesàro mean.

▷ *Solution*

Typically, online learning (and Bayesian optimization) consider stationary environments, hence the comparison to the static optimum. Dynamic environments are studied in *online algorithms* (see metrical task systems[5], convex function chasing[6], and generalizations of multi-armed bandits to changing reward distributions) and reinforcement learning. When operating in dynamic environments, other metrics such as the competitive ratio,[7] which compares against the best *dynamic* choice, are useful. As we will later come to see in section 13.1 in the context of reinforcement learning, operating in dynamic environments is deeply connected to a rich field of research called *control*.

9.3 Acquisition Functions

It is common to use a so-called *acquisition function* to greedily pick the next point to sample based on the current model.

Throughout our description of acquisition functions, we will focus on a setting where we model $f^\star$ using a Gaussian process which we denote by f. The methods generalize to other means of learning $f^\star$ such as Bayesian neural networks. The various acquisition functions F are used in the same way as is illustrated in alg. 9.3.

One possible acquisition function is uncertainty sampling (8.34), which we discussed in the previous chapter. However, this acquisition function does not at all take into account the objective of maximizing $f^\star$ and focuses solely on exploration.

Suppose that our model f of $f^\star$ is well-calibrated, in the sense that the true function lies within its confidence bounds. Consider the best lower bound, that is, the maximum of the lower confidence bound.

[5] Metrical task systems are a classical example in online algorithms. Suppose we are moving in a (finite) decision space $\mathcal{X}$. In each round, we are given a "task" $f_t : \mathcal{X} \to \mathbb{R}$ which is more or less costly depending on our state $\boldsymbol{x}_t \in \mathcal{X}$. In many contexts, it is natural to assume that it is also costly to move around in the decision space. This cost is modeled by a metric $d(\cdot, \cdot)$ on $\mathcal{X}$. In *metrical task systems*, we want to minimize our total cost,

$$\sum_{t=1}^{T} f_t(\boldsymbol{x}_t) + d(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}).$$

That is, we want to trade completing our tasks optimally with moving around in the state space. Crucially, we do not know the sequence of tasks f_t in advance. Due to the cost associated with moving in the decision space, previous choices affect the future!

[6] *Convex function chasing* (or *convex body chasing*) generalize metrical task systems to continuous domains $\mathcal{X}$. To make any guarantees about the performance in these settings, one typically has to assume that the tasks f_t are convex. Note that this mirrors our assumption in Bayesian optimization that similar alternatives yield similar results.

[7] To assess the performance in dynamic environments, we typically compare to a dynamic optimum. As these problems are difficult (we are usually not able to guarantee convergence to the dynamic optimum), one considers a multiplicative performance metric similar to the approximation ratio, the *competitive ratio*,

$$\text{cost}(\text{ALG}) \leq \alpha \cdot \text{cost}(\text{OPT}),$$

where OPT corresponds to the dynamic optimal choice (in hindsight).

Algorithm 9.3: Bayesian optimization (with GPs)

initialize $f \sim \mathcal{GP}(\mu_0, k_0)$
for $t = 1$ **to** T **do**
  choose $\boldsymbol{x}_t = \arg\max_{x \in \mathcal{X}} F(\boldsymbol{x}; \mu_{t-1}, k_{t-1})$
  observe $y_t = f(\boldsymbol{x}_t) + \epsilon_t$
  perform a Bayesian update to obtain μ_t and k_t

Now, if the true function is really contained in the confidence bounds, it must hold that the optimum is somewhere above this best lower bound. In particular, we can exclude all regions of the domain where the upper confidence bound (the optimistic estimate of the function value) is lower than the best lower bound. This is visualized in fig. 9.3.

Therefore, we only really care how the function looks like in the regions where the upper confidence bound is larger than the best lower bound. The key idea behind the methods that we will explore is to focus exploration on these plausible maximizers.

Note that it is crucial that our uncertainty about f reflects the "fit" of our model to the unknown function. If the model is not well calibrated or does not describe the underlying function at all, these methods will perform very poorly. This is where we can use the Bayesian philosophy by imposing a prior belief (that may be conservative).

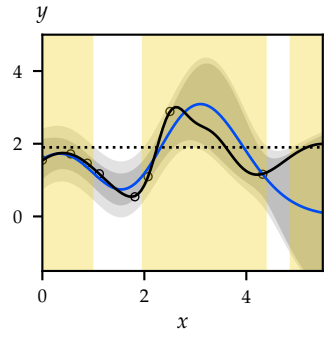


Figure 9.3: Optimism in Bayesian optimization. The **unknown function** is shown in black, our **model** in blue with gray confidence bounds. The dotted black line denotes the maximum lower bound. We can therefore focus our exploration to the yellow regions where the upper confidence bound is higher than the maximum lower bound.

9.3.1 *Upper Confidence Bound*

The principle of *optimism in the face of uncertainty* suggests picking the point where we can hope for the optimal outcome. In this setting, this corresponds to simply maximizing the *upper confidence bound* (UCB),

$$\boldsymbol{x}_{t+1} \doteq \arg\max_{x \in \mathcal{X}} \mu_t(\boldsymbol{x}) + \beta_{t+1}\sigma_t(\boldsymbol{x}), \tag{9.8}$$

where $\sigma_t(\boldsymbol{x}) \doteq \sqrt{k_t(\boldsymbol{x}, \boldsymbol{x})}$ is the standard deviation at $\boldsymbol{x}$ and β_t regulates how confident we are about our model f (i.e., the choice of confidence interval).

This acquisition function naturally trades exploitation by preferring a large posterior mean with exploration by preferring a large posterior variance. Note that if $\beta_t = 0$ then UCB is purely exploitative, whereas, if $\beta_t \to \infty$, UCB recovers uncertainty sampling (i.e., is purely explorative).[8] UCB is an example of an optimism-based method, as it greedily picks the point where we can hope for the best outcome.

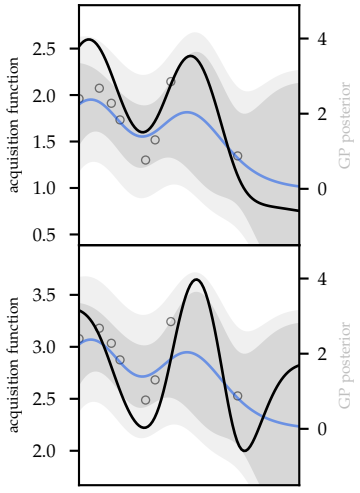


Figure 9.4: Plot of the UCB acquisition function for $\beta = 0.25$ and $\beta = 1$, respectively.

[8] Due to the monotonicity of $(\cdot)^2$, it does not matter whether we optimize the variance or standard deviation at $\boldsymbol{x}$.

As can be seen in fig. 9.4, the UCB acquisition function is generally non-convex. For selecting the next point, we can use approximate global optimization techniques like Lipschitz optimization (in low dimensions) and gradient ascent with random initialization (in high dimensions). Another widely used technique is to sample some random points from the domain, score them according to this criterion, and simply take the best one.

The choice of β_t is crucial for the performance of UCB. Intuitively, for UCB to work even if the unknown function $f^\star$ is not contained in the confidence bounds, we use β_t to re-scale the confidence bounds to enclose $f^\star$ as shown in fig. 9.5. A theoretical analysis requires that β_t is chosen "correctly". Formally, we say that the sequence β_t is chosen correctly if it leads to *well-calibrated confidence intervals*, that is, if with probability at least $1 - \delta$,

$$\forall t \geq 1,\ \forall \boldsymbol{x} \in \mathcal{X}: \quad f^\star(\boldsymbol{x}) \in \mathcal{C}_t(\boldsymbol{x}) \doteq [\mu_{t-1}(\boldsymbol{x}) \pm \beta_t(\delta) \cdot \sigma_{t-1}(\boldsymbol{x})]. \tag{9.9}$$

Bounds on $\beta_t(\delta)$ can be derived both in a "Bayesian" and in a "frequentist" setting. In the Bayesian setting, it is assumed that $f^\star$ is drawn from the prior GP, i.e., $f^\star \sim \mathcal{GP}(\mu_0, k_0)$. However, in many cases this may be an unrealistic assumption. In the frequentist setting, it is assumed instead that $f^\star$ is an element of a reproducing kernel Hilbert space $\mathcal{H}_k(\mathcal{X})$ which depending on the kernel k can encompass a large class of functions. We will discuss the Bayesian setting first and later return to the frequentist setting.

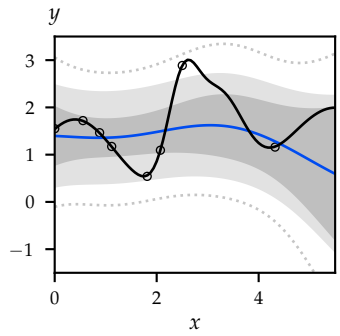


Figure 9.5: Re-scaling the confidence bounds. The dotted gray lines represent updated confidence bounds.

Theorem 9.4 (Bayesian confidence intervals, lemma 5.5 of Srinivas et al. (2010))**.** *Let $\delta \in (0,1)$. Assuming $f^\star \sim \mathcal{GP}(\mu_0, k_0)$ and Gaussian observation noise $\epsilon_t \sim \mathcal{N}(0, \sigma_n^2)$, the sequence*

$$\beta_t(\delta) = \mathcal{O}\left(\sqrt{\log(|\mathcal{X}|t/\delta)}\right) \tag{9.10}$$

satisfies $\mathbb{P}(\forall t \geq 1, \boldsymbol{x} \in \mathcal{X} : f^\star(\boldsymbol{x}) \in \mathcal{C}_t(\boldsymbol{x})) \geq 1 - \delta$.

Exercise 9.5: Bayesian confidence intervals

In this exercise, we derive the above theorem.

1. For fixed $t \geq 1$ and $\boldsymbol{x} \in \mathcal{X}$, prove

$$\mathbb{P}(f^\star(\boldsymbol{x}) \notin \mathcal{C}_t(\boldsymbol{x}) \mid \boldsymbol{x}_{1:t-1}, y_{1:t-1}) \leq e^{-\beta_t^2/2}. \tag{9.11}$$

   Hint: Bound $\mathbb{P}(Z > c)$ for $Z \sim \mathcal{N}(0,1)$ and $c > 0$.
2. Prove theorem 9.4.

▷ *Solution*

Under the assumption of well-calibrated confidence intervals, we can bound the regret of UCB.

Theorem 9.6 (Regret of GP-UCB, theorem 2 of Srinivas et al. (2010))**.** *If $\beta_t(\delta)$ is chosen "correctly" for a fixed $\delta \in (0,1)$, with probability at least $1-\delta$, greedily choosing the upper confidence bound yields cumulative regret*

$$R_T = \mathcal{O}\Big(\beta_T(\delta)\sqrt{\gamma_T T}\Big) \tag{9.12}$$

where

$$\gamma_T \doteq \max_{\substack{S\subseteq\mathcal{X}\\|S|=T}} \mathrm{I}(\boldsymbol{f}_S;\boldsymbol{y}_S) \tag{9.13}$$

is the maximum information gain after T rounds.

Exercise 9.7: Regret of GP-UCB

To develop some intuition, we will derive the above theorem.

1. Show that if eq. (9.9) holds, then for a fixed $t \geq 1$ the instantaneous regret r_t is bounded by $2\beta_t\sigma_{t-1}(\boldsymbol{x}_t)$.
2. Let $S_T \doteq \{\boldsymbol{x}_t\}_{t=1}^T$, and define $\boldsymbol{f}_T \doteq \boldsymbol{f}_{S_T}$ and $\boldsymbol{y}_T \doteq \boldsymbol{y}_{s_T}$. Prove
$$\mathrm{I}(\boldsymbol{f}_T;\boldsymbol{y}_T) = \frac{1}{2}\sum_{t=1}^{T}\log\left(1+\frac{\sigma_{t-1}^2(\boldsymbol{x}_t)}{\sigma_n^2}\right). \tag{9.14}$$
3. Combine (1) and (2) to show theorem 9.6. We assume w.l.o.g. that the sequence $\{\beta_t\}_t$ is monotonically increasing.
Hint: If $s \in [0,M]$ for some $M > 0$ then $s \leq C\cdot\log(1+s)$ with $C \doteq M/\log(1+M)$.

▷ *Solution*

Observe that if the information gain is sublinear in T then we achieve sublinear regret and, in particular, converge to the true optimum. The information gain γ_T measures how much can be learned about $f^\star$ within T rounds. If the function is assumed to be smooth (perhaps even linear), then the information gain is smaller than if the function was assumed to be "rough". Intuitively, the smoother the functions encoded by the prior, the smaller is the class of functions to choose from and the more can be learned from a single observation about "neighboring" points.

Theorem 9.8 (Information gain of common kernels)**.** *Due to submodularity, we have the following bounds on the information gain of common kernels:*

- linear kernel
$$\gamma_T = \mathcal{O}(d\log T), \tag{9.15}$$
- Gaussian kernel
$$\gamma_T = \mathcal{O}\Big((\log T)^{d+1}\Big), \tag{9.16}$$

- Matérn kernel *for* $\nu > \frac{1}{2}$

$$\gamma_T = \mathcal{O}\left(T^{\frac{d}{2\nu+d}} (\log T)^{\frac{2\nu}{2\nu+d}}\right). \tag{9.17}$$

Proof. Refer to theorem 5 of "Gaussian process optimization in the bandit setting: No regret and experimental design" (Srinivas et al., 2010) and remark 2 of "On information gain and regret bounds in gaussian process bandits" (Vakili et al., 2021). □

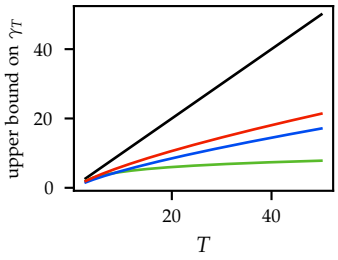


Figure 9.6: Information gain of **independent**, **linear**, **Gaussian**, and **Matérn** ($\nu \approx 0.5$) kernels with $d = 2$ (up to constant factors). The kernels with sublinear information gain have strong diminishing returns (due to their strong dependence between "close" points). In contrast, the independent kernel has no dependence between points in the domain, and therefore no diminishing returns. Intuitively, the "smoother" the class of functions modeled by the kernel, the stronger are the diminishing returns.

The information gain of common kernels is illustrated in fig. 9.6. Notably, when all points in the domain are independent, the information gain is linear in T. This is because when the function $f^\star$ may be arbitrarily "rough", we cannot generalize from a single observation to "neighboring" points, and as there are infinitely many points in the domain $\mathcal{X}$ there are no diminishing returns. As one would expect, in this case, theorem 9.6 does not yield sublinear regret. However, we can see from theorem 9.8 that the information gain is sublinear for linear, Gaussian, and most Matérn kernels. Moreover, observe that unless the function is linear, the information gain grows exponentially with the dimension d. This is because the number of "neighboring" points (with respect to Euclidean geometry) decreases exponentially with the dimension which is also known as the *curse of dimensionality*.

Exercise 9.9: Sublinear regret of GP-UCB for a linear kernel

Assume that $f^\star \sim \mathcal{GP}(0, k)$ where k is the linear kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}'.$$

In addition, we assume that for any $\boldsymbol{x} \in \mathcal{X}$, $\|\boldsymbol{x}\|_2 \leq 1$. Moreover, recall that the points in a finite set $S \subseteq \mathcal{X}$ can be written in a matrix form (the "design matrix") which we denote by $\boldsymbol{X}_S \in \mathbb{R}^{d \times |S|}$.

1. Prove that $\gamma_T = \mathcal{O}(d \log T)$.
2. Deduce from (1) and theorem 9.6 that $\lim_{T\to\infty} R_T / T = 0$.

▷ *Solution*

As mentioned, the size of the confidence intervals can also be analyzed under a frequentist assumption on $f^\star$.

Theorem 9.10 (Frequentist confidence intervals, theorem 2 of Chowdhury and Gopalan (2017))**.** *Let* $\delta \in (0, 1)$. *Assuming* $f^\star \in \mathcal{H}_k(\mathcal{X})$, *we have that with probability at least* $1 - \delta$, *the sequence*

$$\beta_t(\delta) = \|f^\star\|_k + \sigma_n \sqrt{2(\gamma_t + \log(1/\delta))} \tag{9.18}$$

satisfies $\mathbb{P}(\forall t \geq 1, \boldsymbol{x} \in \mathcal{X} : f^\star(\boldsymbol{x}) \in \mathcal{C}_t(\boldsymbol{x})) \geq 1 - \delta$.

That is, β_t depends on the information gain of the kernel as well as on the "complexity" of $f^\star$ which is measured in terms of the norm of the underlying reproducing kernel Hilbert space $\mathcal{H}_k(\mathcal{X})$. We remark that theorem 9.10 holds also under the looser assumption that observations are perturbed by σ_n-sub-Gaussian noise (cf. eq. (1.86)) instead of Gaussian noise.

Remark 9.11: Bayesian vs frequentist assumption

Theorems 9.4 and 9.10 provide different bounds on $\beta_t(\delta)$ based on fundamentally different assumptions on the ground truth $f^\star$: The Bayesian assumption is that $f^\star$ is drawn from the prior GP, whereas the frequentist assumption is that $f^\star$ is an element of a reproducing kernel Hilbert space $\mathcal{H}_k(\mathcal{X})$. The frequentist assumption holds uniformly for all functions $f^\star$ with $\|f^\star\|_k < \infty$, whereas the Bayesian assumption holds only under the Bayesian "belief" that $f^\star$ is drawn from the prior GP.

Interestingly, neither assumption encompasses the other. This is because if $f \sim \mathcal{GP}(0, k)$ then it can be shown that almost surely $\|f\|_k = \infty$, which implies that $f \notin \mathcal{H}_k(\mathcal{X})$ (Srinivas et al., 2010).

This concludes our discussion of the UCB algorithm. We have seen that its regret can be analyzed under both Bayesian and frequentist assumptions on $f^\star$.

9.3.2 *Improvement*

Another well-known family of methods is based on keeping track of a running optimum $\hat{f}_t$, and scoring points according to their improvement upon the running optimum. The *improvement* of $\boldsymbol{x}$ after round t is measured by

$$I_t(\boldsymbol{x}) \doteq (f(\boldsymbol{x}) - \hat{f}_t)_+ \tag{9.19}$$

where we use $(\cdot)_+$ to denote $\max\{0, \cdot\}$.

The *probability of improvement* (PI) picks the point that maximizes the probability to improve upon the running optimum,

$$\begin{aligned} \boldsymbol{x}_{t+1} &\doteq \arg\max_{\boldsymbol{x}\in\mathcal{X}} \mathbb{P}(I_t(\boldsymbol{x}) > 0 \mid \boldsymbol{x}_{1:t}, y_{1:t}) && (9.20)\\ &= \arg\max_{\boldsymbol{x}\in\mathcal{X}} \mathbb{P}(f(\boldsymbol{x}) > \hat{f}_t \mid \boldsymbol{x}_{1:t}, y_{1:t}) && (9.21)\\ &= \arg\max_{\boldsymbol{x}\in\mathcal{X}} \Phi\left(\frac{\mu_t(\boldsymbol{x}) - \hat{f}_t}{\sigma_t(\boldsymbol{x})}\right) && (9.22) \end{aligned}$$

using linear transformations of Gaussians (1.123)

where Φ denotes the CDF of the standard normal distribution and we use that $f(\boldsymbol{x}) \mid \boldsymbol{x}_{1:t}, y_{1:t} \sim \mathcal{N}(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x}))$. Probability of improvement

tends to be biased in favor of exploitation, as it prefers points with large posterior mean and small posterior variance which is typically true "close" to the previously observed maximum $\hat{f}_t$.

Probability of improvement looks at *how likely* a point is to improve upon the running optimum. An alternative is to look at *how much* a point is expected to improve upon the running optimum. This acquisition function is called the *expected improvement* (EI),

$$x_{t+1} \doteq \arg\max_{x \in \mathcal{X}} \mathbb{E}[I_t(x) \mid x_{1:t}, y_{1:t}]\,. \tag{9.23}$$

Intuitively, EI seeks a large expected improvement (exploitation) while also preferring states with a large variance (exploration). Expected improvement yields the same regret bound as UCB (Nguyen et al., 2017).

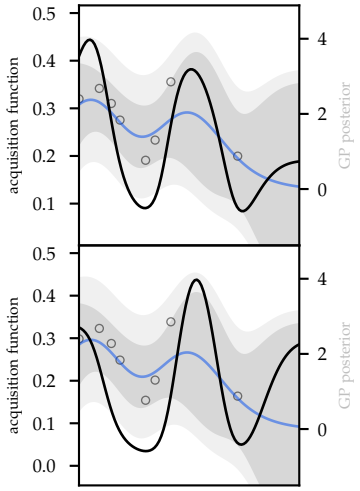


Figure 9.7: Plot of the PI and EI acquisition functions, respectively.

Exercise 9.12: Closed-form expected improvement

Let us denote the acquisition function of EI from eq. (9.23) by $\mathrm{EI}_t(x)$. In this exercise, we derive a closed-form expression.

1. Show that

$$\mathrm{EI}_t(x) = \int_{(\hat{f}_t - \mu_t(x))/\sigma_t(x)}^{+\infty} (\mu_t(x) + \sigma_t(x)\epsilon - \hat{f}_t) \cdot \phi(\epsilon)\, d\epsilon \tag{9.24}$$

   where ϕ is the PDF of the univariate standard normal distribution (1.6).
   Hint: Reparameterize the posterior distribution

$$f(x) \mid x_{1:t}, y_{1:t} \sim \mathcal{N}(\mu_t(x), \sigma_t^2(x))$$

   using a standard normal distribution.

2. Using the above expression, show that

$$\mathrm{EI}_t(x) = (\mu_t(x) - \hat{f}_t)\Phi\left(\frac{\mu_t(x) - \hat{f}_t}{\sigma_t(x)}\right) + \sigma_t(x)\phi\left(\frac{\mu_t(x) - \hat{f}_t}{\sigma_t(x)}\right) \tag{9.25}$$

   where $\Phi(u) \doteq \int_{-\infty}^{u} \phi(\epsilon)\, d\epsilon$ denotes the CDF of the standard normal distribution.

Note that the first term of eq. (9.25) encourages exploitation while the second term encourages exploration. EI can be seen as a special case of UCB where the confidence bounds are scaled depending on x: $\beta_t = \phi(z_t(x))/\Phi(z_t(x))$ for $z_t(x) \doteq (\mu_t(x) - \hat{f}_t)/\sigma_t(x)$.

▷ *Solution*

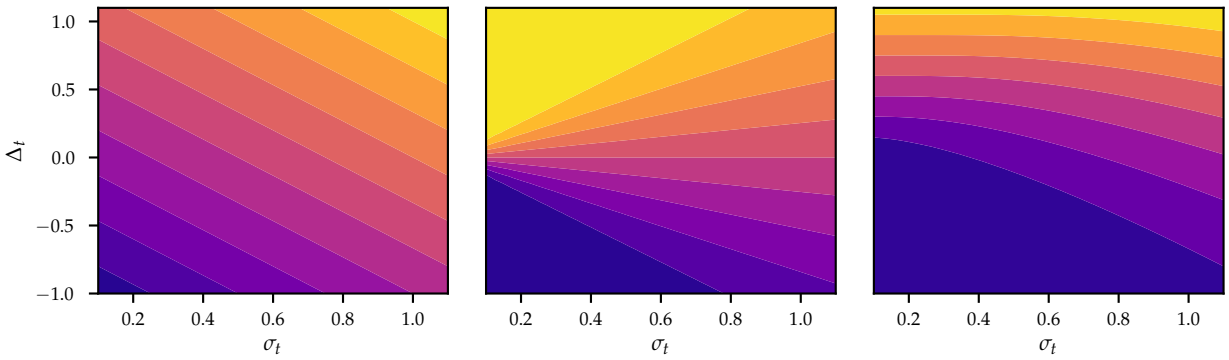


Figure 9.8: Contour lines of acquisition functions for varying $\Delta_t = \mu_t(\boldsymbol{x}) - \hat{f}_t$ and σ_t. The first graph shows contour lines of UCB with $\beta_t = 0.75$, the second of PI, and the third of EI.

9.3.3 *Thompson Sampling*

We can also interpret the principle of *optimism in the face of uncertainty* in a slightly different way than we did with UCB (and EI). Suppose we select the next point according to the probability that it is optimal (assuming that the posterior distribution is an accurate representation of the uncertainty),

$$\pi(\boldsymbol{x} \mid \boldsymbol{x}_{1:t}, y_{1:t}) \doteq \mathbb{P}_{f|\boldsymbol{x}_{1:t},y_{1:t}}\left(f(\boldsymbol{x}) = \max_{\boldsymbol{x}'} f(\boldsymbol{x}')\right) \tag{9.26}$$

$$\boldsymbol{x}_{t+1} \sim \pi(\cdot \mid \boldsymbol{x}_{1:t}, y_{1:t}). \tag{9.27}$$

This approach is called *probability matching*. Probability matching is exploratory as it prefers points with larger variance (as they automatically have a larger chance of being optimal), but at the same time exploitative as it effectively discards points with low posterior mean and low posterior variance. Unfortunately, it is generally difficult to compute π analytically given a posterior.

Instead, it is common to use a sampling-based approximation of π. Observe that the density π can be expressed as an expectation,

$$\pi(\boldsymbol{x} \mid \boldsymbol{x}_{1:t}, y_{1:t}) = \mathbb{E}_{f|\boldsymbol{x}_{1:t},y_{1:t}}\left[\mathbb{1}\{f(\boldsymbol{x}) = \max_{\boldsymbol{x}'} f(\boldsymbol{x}')\}\right], \tag{9.28}$$

which we can approximate using Monte Carlo sampling (typically using a single sample),

$$\approx \mathbb{1}\{\tilde{f}_{t+1}(\boldsymbol{x}) = \max_{\boldsymbol{x}'} \tilde{f}_{t+1}(\boldsymbol{x}')\} \tag{9.29}$$

where $\tilde{f}_{t+1} \sim p(\cdot \mid \boldsymbol{x}_{1:t}, y_{1:t})$ is a sample from our posterior distribution. Observe that this approximation of π coincides with a point density at the maximizer of $\tilde{f}_{t+1}$.

The resulting algorithm is known as *Thompson sampling*. At time $t+1$, we sample a function $\tilde{f}_{t+1} \sim p(\cdot \mid \boldsymbol{x}_{1:t}, y_{1:t})$ from our posterior distri-

bution. Then, we simply maximize $\tilde{f}_{t+1}$,

$$\boldsymbol{x}_{t+1} \doteq \arg\max_{\boldsymbol{x} \in \mathcal{X}} \tilde{f}_{t+1}(\boldsymbol{x}). \tag{9.30}$$

In many cases, the randomness in the realizations of $\tilde{f}_{t+1}$ is already sufficient to effectively trade exploration and exploitation. Similar regret bounds to those of UCB can also be established for Thompson sampling (Russo and Van Roy, 2016; Kandasamy et al., 2018).

9.3.4 *Information-Directed Sampling*

For their analysis of Thompson sampling, Russo and Van Roy (2016) introduce the *regret-information ratio*

$$\Psi_t(\boldsymbol{x}) \doteq \frac{\Delta(\boldsymbol{x})^2}{I_t(\boldsymbol{x})} \tag{9.31}$$

where $\Delta(\boldsymbol{x}) \doteq \max_{\boldsymbol{x}'} f^\star(\boldsymbol{x}') - f^\star(\boldsymbol{x})$ is the instantaneous regret if $\boldsymbol{x}$ were to be observed (so that $r_t = \Delta(\boldsymbol{x}_t)$) and $I_t(\boldsymbol{x})$ is some function capturing the "information gain" associated with observing $\boldsymbol{x}$ in iteration $t+1$. They make the key observation that the regret $\Delta(\cdot)$ decreases when $I_t(\cdot)$ decreases, as a small $I_t(\cdot)$ implies that the algorithm has already learned a lot about the function $f^\star$. The strength of this relationship is quantified by the regret-information ratio.

Theorem 9.13 (Proposition 1 of Russo and Van Roy (2014) and theorem 8 of Kirschner and Krause (2018))**.** *For any iteration $T \geq 1$, let $\sum_{t=1}^{T} I_{t-1}(\boldsymbol{x}_t) \leq \gamma_T$ and suppose that $\Psi_{t-1}(\boldsymbol{x}_t) \leq \overline{\Psi}_T$ for all $t \in [T]$. Then, the cumulative regret is bounded by*

$$R_T \leq \sqrt{\gamma_T \overline{\Psi}_T T}. \tag{9.32}$$

Proof. By eq. (9.31), $r_t = \Delta(\boldsymbol{x}_t) = \sqrt{\Psi_{t-1}(\boldsymbol{x}_t) \cdot I_{t-1}(\boldsymbol{x}_t)}$. Hence,

$$\begin{aligned}
R_T &= \sum_{t=1}^{T} r_t \\
&= \sum_{t=1}^{T} \sqrt{\Psi_{t-1}(\boldsymbol{x}_t) \cdot I_{t-1}(\boldsymbol{x}_t)} \\
&\leq \sqrt{\sum_{t=1}^{T} \Psi_{t-1}(\boldsymbol{x}_t) \cdot \sum_{t=1}^{T} I_{t-1}(\boldsymbol{x}_t)} && \text{using the Cauchy-Schwarz inequality} \\
&\leq \sqrt{\gamma_T \overline{\Psi}_T T}. && \text{using the assumptions on } I_t(\cdot) \text{ and } \Psi_t(\cdot)
\end{aligned}$$

□

Example 9.14: How to measure "information gain"?

One possibility of measuring the "information gain" is

$$I_t(\boldsymbol{x}) \doteq \mathrm{I}(f_{\boldsymbol{x}}; y_{\boldsymbol{x}} \mid \boldsymbol{x}_{1:t}, y_{1:t}) \tag{9.33}$$

which — as you may recall — is precisely the marginal gain of the utility $I(S) = \mathrm{I}(\boldsymbol{f}_S; \boldsymbol{y}_S)$ we were studying in chapter 8. In this case,

$$\begin{aligned}
\sum_{t=1}^{T} I_{t-1}(\boldsymbol{x}_t) &= \sum_{t=1}^{T} \mathrm{I}(f_{\boldsymbol{x}_t}; y_{\boldsymbol{x}_t} \mid \boldsymbol{x}_{1:t-1}, y_{1:t-1}) \\
&= \sum_{t=1}^{T} \mathrm{I}(\boldsymbol{f}_{\boldsymbol{x}_{1:T}}; y_{\boldsymbol{x}_t} \mid \boldsymbol{x}_{1:t-1}, y_{1:t-1}) && \text{using } \mathrm{I}(\mathbf{X}, \mathbf{Z}; \mathbf{Y}) \geq \mathrm{I}(\mathbf{X}; \mathbf{Y}) \text{ which follows from eqs. (8.14) and (8.18) and is called } \textit{monotonicity of MI} \\
&= \mathrm{I}(\boldsymbol{f}_{\boldsymbol{x}_{1:T}}; \boldsymbol{y}_{\boldsymbol{x}_{1:T}}) && \text{by repeated application of eq. (8.18), also called the } \textit{chain rule of MI} \\
&\leq \gamma_T. && \text{by definition of } \gamma_T \text{ (9.13)}
\end{aligned}$$

The regret bound from theorem 9.13 suggests an algorithm which in each iteration chooses the point which minimizes the regret-information ratio (9.31). However, this is not possible since $\Delta(\cdot)$ is unknown due to its dependence on $f^\star$. Kirschner and Krause (2018) propose to use a surrogate to the regret which is based on the current model of $f^\star$,

$$\hat{\Delta}_t(\boldsymbol{x}) \doteq \max_{\boldsymbol{x}' \in \mathcal{X}} u_t(\boldsymbol{x}') - l_t(\boldsymbol{x}). \tag{9.34}$$

Here, $u_t(\boldsymbol{x}) \doteq \mu_t(\boldsymbol{x}) + \beta_{t+1}\sigma_t(\boldsymbol{x})$ and $l_t(\boldsymbol{x}) \doteq \mu_t(\boldsymbol{x}) - \beta_{t+1}\sigma_t(\boldsymbol{x})$ are the upper and lower confidence bounds of the confidence interval $\mathcal{C}_t(\boldsymbol{x})$ of $f^\star(\boldsymbol{x})$, respectively. Similarly to our discussion of UCB, we make the assumption that the sequence β_t is chosen "correctly" (cf. eq. (9.9)) so that the confidence interval is well-calibrated and $\Delta(\boldsymbol{x}) \leq \hat{\Delta}_t(\boldsymbol{x})$ with high probability. The resulting algorithm

$$\boldsymbol{x}_{t+1} \doteq \arg\min_{\boldsymbol{x} \in \mathcal{X}} \left\{ \widehat{\Psi}_t(\boldsymbol{x}) \doteq \frac{\hat{\Delta}_t(\boldsymbol{x})^2}{I_t(\boldsymbol{x})} \right\} \tag{9.35}$$

is known as *information-directed sampling* (IDS).

Theorem 9.15 (Regret of IDS, lemma 8 of Kirschner and Krause (2018))**.** *Let $\beta_t(\delta)$ be chosen "correctly" for a fixed $\delta \in (0, 1)$. Then, if the measure of information gain is $I_t(\boldsymbol{x}) = \mathrm{I}(f_{\boldsymbol{x}}; y_{\boldsymbol{x}} \mid \boldsymbol{x}_{1:t}, y_{1:t})$, with probability at least $1 - \delta$, IDS has cumulative regret*

$$R_T = \mathcal{O}\left(\beta_T(\delta)\sqrt{\gamma_T T}\right). \tag{9.36}$$

Exercise 9.16: Regret of IDS

We derive the above theorem.

1. Prove that for all $t \geq 1$, $\hat{\Delta}_t(\boldsymbol{x}_t^{\mathrm{UCB}}) \leq 2\beta_{t+1}\sigma_t(\boldsymbol{x}_t^{\mathrm{UCB}})$ where we denote by $\boldsymbol{x}_t^{\mathrm{UCB}} \doteq \arg\max_{\boldsymbol{x}\in\mathcal{X}} u_t(\boldsymbol{x})$ the point maximizing the upper confidence bound after iteration t.
2. Using (1) and the assumption on I_t, bound $\widehat{\Psi}_t(\boldsymbol{x}_t^{\mathrm{UCB}})$.
   Hint: You may find the hint of exercise 9.7 (3) useful.
3. Complete the proof of theorem 9.15.

▷ *Solution*

Regret bounds such as theorem 9.15 can be derived also for different measures of information gain. For example, the argument of exercise 9.16 also goes through for the "greedy" measure

$$I_t(\boldsymbol{x}) \doteq \mathrm{I}(f_{\boldsymbol{x}_t^{\mathrm{UCB}}}; y_{\boldsymbol{x}} \mid \boldsymbol{x}_{1:t}, y_{1:t}) \tag{9.37}$$

which focuses exclusively on reducing the uncertainty at $\boldsymbol{x}_t^{\mathrm{UCB}}$ rather than globally. We compare the two measures of information gain in fig. 9.9. Observe that the acquisition function depends critically on the choice of $I_t(\cdot)$ and is less sensitive to the scaling of confidence intervals.

IDS trades exploitation and exploration by balancing the (exploitative) regret surrogate with a measure of information gain (such as those studied in chapter 8) that is purely explorative. In this way, IDS can account for kinds of information which are not addressed by alternative algorithms (Russo and Van Roy, 2014): Depending on the measure of information gain, IDS can select points to obtain *indirect information* about other points or *cumulating information* that does not immediately lead to a higher reward but only when combined with subsequent observations. Moreover, IDS avoids selecting points which yield *irrelevant information*.

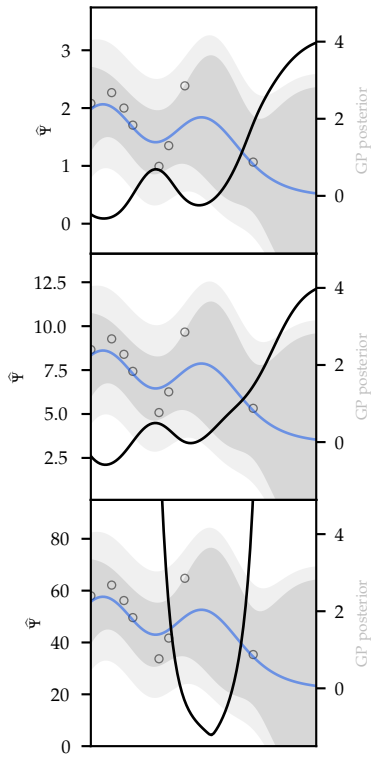


Figure 9.9: Plot of the surrogate regret-information ratio $\widehat{\Psi}$: IDS selects its minimizer. The first two plots use the "global" information gain measure from example 9.14 with $\beta = 0.25$ and $\beta = 0.5$, respectively. The third plot uses the "greedy" information gain measure from eq. (9.37) and $\beta = 1$.

9.4 *Model Selection*

Selecting a model of $f^\star$ is much harder than in the i.i.d. data setting of supervised learning. There are mainly the two following dangers,

- the data sets collected in active learning and Bayesian optimization are *small*; and
- the data points are selected *dependently* on prior observations.

This leads to a specific danger of overfitting. In particular, due to feedback loops between the model and the acquisition function, one may end up sampling the same point repeatedly.

One approach to reduce the chance of overfitting is the use of hyperpriors which we mentioned previously in section 4.4.2. Another

approach that often works fairly well is to occasionally (according to some schedule) select points uniformly at random instead of using the acquisition function. This tends to prevent getting stuck in suboptimal parts of the state space.

Optional Readings

- Srinivas, Krause, Kakade, and Seeger (2010). *Gaussian process optimization in the bandit setting: No regret and experimental design*.
- Golovin, Solnik, Moitra, Kochanski, Karro, and Sculley (2017). *Google vizier: A service for black-box optimization*.
- Romero, Krause, and Arnold (2013). *Navigating the protein fitness landscape with Gaussian processes*.
- Chowdhury and Gopalan (2017). *On kernelized multi-armed bandits*.

10

Markov Decision Processes

We will now turn to the topic of probabilistic planning. Planning deals with the problem of deciding which action an agent should play in a (stochastic) environment.[1] A key formalism for probabilistic planning in *known* environments are so-called Markov decision processes. Starting from the next chapter, we will look at reinforcement learning, which extends probabilistic planning to unknown environments.

[1] An environment is *stochastic* as opposed to deterministic, when the outcome of actions is random.

Consider the setting where we have a sequence of states $(X_t)_{t\in\mathbb{N}_0}$ similarly to Markov chains. But now, the next state X_{t+1} of an agent does not only depend on the previous state X_t but also depends on the last action A_t of this agent.

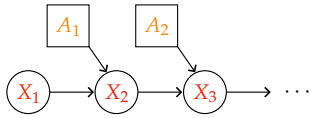


Figure 10.1: Directed graphical model of a Markov decision process with hidden states X_t and actions A_t.

Definition 10.1 ((Finite) Markov decision process, MDP)**.** A *(finite) Markov decision process* is specified by

- a (finite) set of *states* $X \doteq \{1, \dots, n\}$,
- a (finite) set of *actions* $A \doteq \{1, \dots, m\}$,
- *transition probabilities*

$$p(x' \mid x, a) \doteq \mathbb{P}\big(X_{t+1} = x' \mid X_t = x, A_t = a\big) \tag{10.1}$$

  which is also called the *dynamics model*, and
- a *reward function* $r : X \times A \to \mathbb{R}$ which maps the current state x and an action a to some reward.

The reward function may also depend on the next state x', however, we stick to the above model for simplicity. Also, the reward function can be random with mean r. Observe that r induces the sequence of rewards $(R_t)_{t\in\mathbb{N}_0}$, where

$$R_t \doteq r(X_t, A_t), \tag{10.2}$$

which is sometimes used in the literature instead of r.

Crucially, we assume the dynamics model p and the reward function r to be known. That is, we operate in a known environment. For now,

we also assume that the environment is *fully observable*. In other words, we assume that our agent knows its current state. In section 10.4, we discuss how this method can be extended to the partially observable setting.

Our fundamental objective is to learn how the agent should behave to optimize its reward. In other words, given its current state, the agent should decide (optimally) on the action to play. Such a decision map — whether optimal or not — is called a policy.

Definition 10.2 (Policy)**.** A *policy* is a function that maps each state $x \in X$ to a probability distribution over the actions. That is, for any $t > 0$,

$$\pi(a \mid x) \doteq \mathbb{P}(A_t = a \mid X_t = x). \tag{10.3}$$

In other words, a policy assigns to each action $a \in A$, a probability of being played given the current state $x \in X$.

We assume that policies are stationary, that is, do not change over time.

Remark 10.3: Stochastic policies

We will see later in this chapter that in fully observable environments optimal policies are always deterministic. Thus, there is no need to consider stochastic policies in the context of Markov decision processes. For this chapter, you can think of a policy π simply as a deterministic mapping $\pi : X \to A$ from current state to played action. In the context of reinforcement learning, we will later see in section 12.4.5 that randomized policies are important in trading exploration and exploitation.

Observe that a policy induces a Markov chain $(X_t^\pi)_{t \in \mathbb{N}_0}$ with transition probabilities,

$$p^\pi(x' \mid x) \doteq \mathbb{P}\big(X_{t+1}^\pi = x' \mid X_t^\pi = x\big) = \sum_{a \in A} \pi(a \mid x) p(x' \mid x, a). \tag{10.4}$$

This is crucial: if our agent follows a fixed policy (i.e., decision-making protocol) then the evolution of the process is described fully by a Markov chain.

As mentioned, we want to maximize the reward. There are many models of calculating a score from the infinite sequence of rewards $(R_t)_{t \in \mathbb{N}_0}$. For the purpose of our discussion of Markov decision processes and reinforcement learning, we will focus on a very common reward called discounted payoff.

Definition 10.4 (Discounted payoff)**.** The *discounted payoff* (also called *discounted total reward*) from time t is defined as the random variable,

$$G_t \doteq \sum_{m=0}^{\infty} \gamma^m R_{t+m} \tag{10.5}$$

where $\gamma \in [0, 1)$ is the *discount factor*.

Remark 10.5: Other reward models

Other well-known methods for combining rewards into a score are

$$\overset{\text{(instantaneous)}}{G_t \doteq R_t} \quad , \quad \overset{\text{(finite-horizon)}}{G_t \doteq \sum_{m=0}^{T-1} R_{t+m}}, \quad \text{and} \quad \overset{\text{(mean payoff)}}{G_t \doteq \liminf_{T\to\infty} \frac{1}{T} \sum_{m=0}^{T} R_{t+m}}.$$

The methods that we will discuss can also be analyzed using these or other alternative reward models.

We now want to understand the effect of the starting state and initial action on our optimization objective G_t. To analyze this, it is common to use the following two functions:

Definition 10.6 (State value function)**.** The *state value function*,[2]

$$v_t^\pi(x) \doteq \mathbb{E}_\pi[G_t \mid X_t = x], \tag{10.7}$$

measures the average discounted payoff from time t starting from state $x \in X$.

[2] Recall that following a fixed policy π induces a Markov chain $(X_t^\pi)_{t\in\mathbb{N}_0}$. We define

$$\mathbb{E}_\pi[\cdot] \doteq \mathbb{E}_{(X_t^\pi)_{t\in\mathbb{N}_0}}[\cdot] \tag{10.6}$$

as an expectation over all possible sequences of states $(x_t)_{t\in\mathbb{N}_0}$ within this Markov chain.

Definition 10.7 (State-action value function)**.** The *state-action value function* (also called *Q-function*),

$$q_t^\pi(x, a) \doteq \mathbb{E}_\pi[G_t \mid X_t = x, A_t = a] \tag{10.8}$$

$$= r(x, a) + \gamma \sum_{x'\in X} p(x' \mid x, a) \cdot v_{t+1}^\pi(x'), \tag{10.9}$$

by expanding the defition of the discounted payoff (10.5); corresponds to one step in the induced Markov chain

measures the average discounted payoff from time t starting from state $x \in X$ and with playing action $a \in A$. In other words, it combines the immediate return with the value of the next states.

Note that both $v_t^\pi(x)$ and $q_t^\pi(x, a)$ are deterministic scalar-valued functions.

Because we assumed stationary dynamics, rewards, and policies, the discounted payoff starting from a given state x will be independent of the start time t. Thus, we write $v^\pi(x) \doteq v_0^\pi(x)$ and $q^\pi(x, a) \doteq q_0^\pi(x, a)$ without loss of generality.

10.1 Bellman Expectation Equation

Let us now see how we can compute the value function,

$$
\begin{aligned}
v^\pi(x) &= \mathbb{E}_\pi[G_0 \mid X_0 = x] && \text{using the definition of the value function (10.7)} \\
&= \mathbb{E}_\pi\left[\sum_{m=0}^{\infty} \gamma^m R_m \;\middle|\; X_0 = x\right] && \text{using the definition of the discounted payoff (10.5)} \\
&= \mathbb{E}_\pi\left[\gamma^0 R_0 \;\middle|\; X_0 = x\right] + \gamma \mathbb{E}_\pi\left[\sum_{m=0}^{\infty} \gamma^m R_{m+1} \;\middle|\; X_0 = x\right] && \text{using linearity of expectation (1.24)} \\
&= r(x, \pi(x)) + \gamma \mathbb{E}_{x'}\left[\mathbb{E}_\pi\left[\sum_{m=0}^{\infty} \gamma^m R_{m+1} \;\middle|\; X_1 = x'\right] \;\middle|\; X_0 = x\right] && \text{by simplifying the first expectation and conditioning the second expectation on } X_1 \\
&= r(x, \pi(x)) + \gamma \sum_{x' \in X} p(x' \mid x, \pi(x)) \mathbb{E}_\pi\left[\sum_{m=0}^{\infty} \gamma^m R_{m+1} \;\middle|\; X_1 = x'\right] && \text{expanding the expectation on } X_1 \text{ and using conditional independence of the discounted payoff of } X_0 \text{ given } X_1 \\
&= r(x, \pi(x)) + \gamma \sum_{x' \in X} p(x' \mid x, \pi(x)) \mathbb{E}_\pi\left[\sum_{m=0}^{\infty} \gamma^m R_m \;\middle|\; X_0 = x'\right] && \text{shifting the start time of the discounted payoff using stationarity} \\
&= r(x, \pi(x)) + \gamma \sum_{x' \in X} p(x' \mid x, \pi(x)) \mathbb{E}_\pi\left[G_0 \mid X_0 = x'\right] && \text{using the definition of the discounted payoff (10.5)} \\
&= r(x, \pi(x)) + \gamma \sum_{x' \in X} p(x' \mid x, \pi(x)) \cdot v^\pi(x'). && (10.10) \quad \text{using the definition of the value function (10.7)} \\
&= r(x, \pi(x)) + \gamma \mathbb{E}_{x' \mid x, \pi(x)}\left[v^\pi(x')\right]. && (10.11) \quad \text{interpreting the sum as an expectation}
\end{aligned}
$$

This equation is known as the *Bellman expectation equation*, and it shows a recursive dependence of the value function on itself. The intuition is clear: the value of the current state corresponds to the reward from the next action plus the discounted sum of all future rewards obtained from the subsequent states.

For stochastic policies, the above calculation can be extended to yield,

$$
\begin{aligned}
v^\pi(x) &= \sum_{a \in A} \pi(a \mid x) \left(r(x, a) + \gamma \sum_{x' \in X} p(x' \mid x, a) v^\pi(x') \right) && (10.12) \\
&= \mathbb{E}_{a \sim \pi(x)}\left[r(x, a) + \gamma \mathbb{E}_{x' \mid x, a}\left[v^\pi(x')\right] \right] && (10.13) \\
&= \mathbb{E}_{a \sim \pi(x)}[q^\pi(x, a)]. && (10.14)
\end{aligned}
$$

For stochastic policies, by also conditioning on the first action, one can obtain an analogous equation for the state-action value function,

$$
\begin{aligned}
q^\pi(x, a) &= r(x, a) + \gamma \sum_{x' \in X} p(x' \mid x, a) \sum_{a' \in A} \pi(a' \mid x') q^\pi(x', a') && (10.15) \\
&= r(x, a) + \gamma \mathbb{E}_{x' \mid x, a} \mathbb{E}_{a' \sim \pi(x')}\left[q^\pi(x', a')\right]. && (10.16)
\end{aligned}
$$

Note that it does not make sense to consider a similar recursive formula for the state-action value function in the setting of deterministic

policies as the action played when in state $x \in X$ is uniquely determined as $\pi(x)$. In particular,

$$v^\pi(x) = q^\pi(x, \pi(x)). \tag{10.17}$$

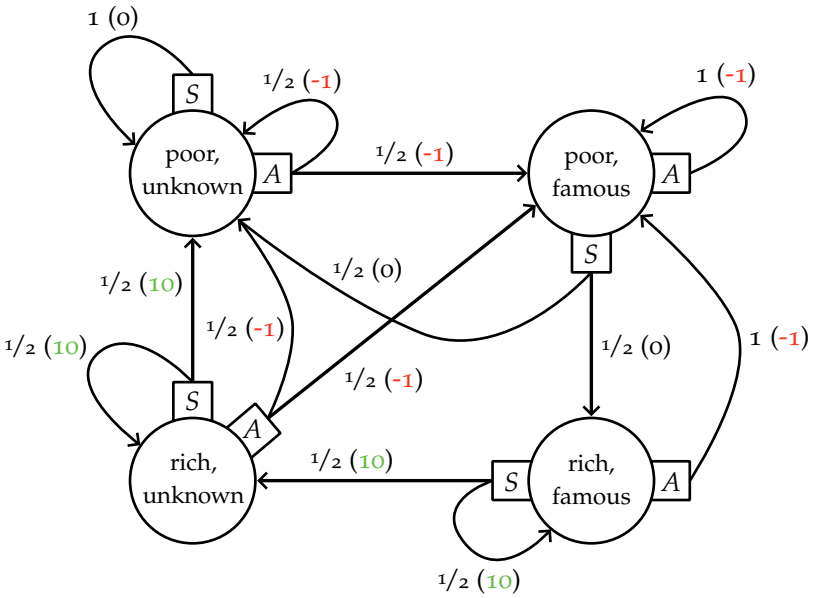


Figure 10.2: Suppose you are building a company. The shown MDP models "how to become rich and famous". Here, the action *S* is short for *saving* and the action *A* is short for *advertising*.

Suppose you begin by being "poor and unknown". Then, the greedy action (i.e., the action maximizing instantaneous reward) is to save. However, within this simplified environment, saving when you are poor and unknown means that you will remain poor and unknown forever. As the potential rewards in other states are substantially larger, this simple example illustrates that following the greedy choice is generally not optimal.

Exercise 10.8: Value functions
Consider the policy, $\pi \equiv S$ (i.e., to always *save*) and let $\gamma = 1/2$. Show that the (rounded) state-action value function q^π is as follows:

	save	advertise
poor, unknown	**0**	0.1
poor, famous	**4.4**	1.2
rich, famous	**17.8**	1.2
rich, unknown	**13.3**	0.1

Shown in bold is the state value function v^π.

▷ *Solution*

10.2 *Policy Evaluation*

Bellman's expectation equation tells us how we can find the value function v^π of a fixed policy π using a system of linear equations!

Using,

$$\boldsymbol{v}^\pi \doteq \begin{bmatrix} v^\pi(1) \\ \vdots \\ v^\pi(n) \end{bmatrix}, \quad \boldsymbol{r}^\pi \doteq \begin{bmatrix} r(1, \pi(1)) \\ \vdots \\ r(n, \pi(n)) \end{bmatrix}, \quad \text{and} \\ \boldsymbol{P}^\pi \doteq \begin{bmatrix} p(1 \mid 1, \pi(1)) & \cdots & p(n \mid 1, \pi(1)) \\ \vdots & \ddots & \vdots \\ p(1 \mid n, \pi(n)) & \cdots & p(n \mid n, \pi(n)) \end{bmatrix} \tag{10.18}$$

and a little bit of linear algebra, the Bellman expectation equation (10.10) is equivalent to

$$\boldsymbol{v}^\pi = \boldsymbol{r}^\pi + \gamma \boldsymbol{P}^\pi \boldsymbol{v}^\pi \tag{10.19}$$

$$\iff \quad (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi) \boldsymbol{v}^\pi = \boldsymbol{r}^\pi$$

$$\iff \quad \boldsymbol{v}^\pi = (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)^{-1} \boldsymbol{r}^\pi. \tag{10.20}$$

Solving this linear system of equations (i.e., performing matrix inversion) takes cubic time in the size of the state space.

10.2.1 *Fixed-point Iteration*

To obtain an (approximate) solution of $\boldsymbol{v}^\pi$, we can use that it is the unique fixed-point of the affine mapping $\boldsymbol{B}^\pi : \mathbb{R}^n \to \mathbb{R}^n$,

$$\boldsymbol{B}^\pi \boldsymbol{v} \doteq \boldsymbol{r}^\pi + \gamma \boldsymbol{P}^\pi \boldsymbol{v}. \tag{10.21}$$

Using this fact (which we will prove in just a moment), we can use fixed-point iteration of $\boldsymbol{B}^\pi$.

Algorithm 10.9: Fixed-point iteration

```
initialize v^π (e.g., as 0)
for t = 1 to T do
    v^π ← B^π v^π = r^π + γ P^π v^π
```

Fixed-point iteration has computational advantages, for example, for sparse transitions.

Theorem 10.10. *$\boldsymbol{v}^\pi$ is the unique fixed-point of $\boldsymbol{B}^\pi$.*

Proof. It is immediate from Bellman's expectation equation (10.19) and the definition of $\boldsymbol{B}^\pi$ (10.21) that $\boldsymbol{v}^\pi$ is a fixed-point of $\boldsymbol{B}^\pi$. To prove uniqueness, we will show that $\boldsymbol{B}^\pi$ is a contraction.

Remark 10.11: Contractions
A *contraction* is a concept from topology. In a Banach space $(\mathcal{X}, \|\cdot\|)$ (a metric space with a norm), $f : \mathcal{X} \to \mathcal{X}$ is a contraction iff there exists a $k < 1$ such that

$$\|f(\boldsymbol{x}) - f(\boldsymbol{y})\| \leq k \cdot \|\boldsymbol{x} - \boldsymbol{y}\| \tag{10.22}$$

for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$. By the *Banach fixed-point theorem*, a contraction admits a unique fixed-point. Intuitively, by iterating the function f, the distance to any fixed-point shrinks by a factor k in each iteration, hence, converges to 0. As we cannot converge to multiple fixed-points simultaneously, the fixed-point of a contraction f must be unique.

Let $\boldsymbol{v} \in \mathbb{R}^n$ and $\boldsymbol{v}' \in \mathbb{R}^n$ be arbitrary initial guesses. We use the ℓ_∞ space,[3]

$$\begin{aligned}
\left\|\boldsymbol{B}^\pi \boldsymbol{v} - \boldsymbol{B}^\pi \boldsymbol{v}'\right\|_\infty &= \left\|\boldsymbol{r}^\pi + \gamma \boldsymbol{P}^\pi \boldsymbol{v} - \boldsymbol{r}^\pi - \gamma \boldsymbol{P}^\pi \boldsymbol{v}'\right\|_\infty && \text{using the definition of } \boldsymbol{B}^\pi \text{ (10.21)} \\
&= \gamma \left\|\boldsymbol{P}^\pi (\boldsymbol{v} - \boldsymbol{v}')\right\|_\infty \\
&\leq \gamma \max_{\boldsymbol{x} \in X} \sum_{x' \in X} p(x' \mid x, \pi(x)) \cdot \left|\boldsymbol{v}(x') - \boldsymbol{v}'(x')\right| . && \text{using the definition of the } \ell_\infty \text{ norm (10.23), expanding the multiplication, and using } \left|\textstyle\sum_i a_i\right| \leq \textstyle\sum_i |a_i| \\
&\leq \gamma \left\|\boldsymbol{v} - \boldsymbol{v}'\right\|_\infty . && \text{using } \textstyle\sum_{x' \in X} p(x' \mid x, \pi(x)) = 1 \text{ and } |\boldsymbol{v}(x') - \boldsymbol{v}'(x')| \leq \|\boldsymbol{v} - \boldsymbol{v}'\|_\infty
\end{aligned} \tag{10.24}$$

Thus, by eq. (10.22), $\boldsymbol{B}^\pi$ is a contraction and by Banach's fixed-point theorem $\boldsymbol{v}^\pi$ is its unique fixed-point. □

[3] The ℓ_∞ *norm* (also called *supremum norm*) is defined as

$$\|\boldsymbol{x}\|_\infty \doteq \max_i |\boldsymbol{x}(i)| . \tag{10.23}$$

Let $\boldsymbol{v}_t^\pi$ be the value function estimate after t iterations. Then, we have for the convergence of fixed-point iteration,

$$\begin{aligned}
\|\boldsymbol{v}_t^\pi - \boldsymbol{v}^\pi\|_\infty &= \left\|\boldsymbol{B}^\pi \boldsymbol{v}_{t-1}^\pi - \boldsymbol{B}^\pi \boldsymbol{v}^\pi\right\|_\infty && \text{using the update rule of fixed-point iteration and } \boldsymbol{B}^\pi \boldsymbol{v}^\pi = \boldsymbol{v}^\pi \\
&\leq \gamma \left\|\boldsymbol{v}_{t-1}^\pi - \boldsymbol{v}^\pi\right\|_\infty && \text{using (10.24)} \\
&= \gamma^t \left\|\boldsymbol{v}_0^\pi - \boldsymbol{v}^\pi\right\|_\infty . && \text{by induction}
\end{aligned} \tag{10.25}$$

This shows that fixed-point iteration converges to $\boldsymbol{v}^\pi$ exponentially quickly.

10.3 Policy Optimization

Recall that our goal was to find an optimal policy,

$$\pi^\star \doteq \arg\max_\pi \mathbb{E}_\pi[G_0]. \tag{10.26}$$

We can alternatively characterize an optimal policy as follows: We define a partial ordering over policies by

$$\pi \geq \pi' \iff v^\pi(x) \geq v^{\pi'}(x) \quad (\forall x \in X). \tag{10.27}$$

$\pi^\star$ is then simply a policy which is maximal according to this partial ordering.

It follows that all optimal policies have identical value functions. Subsequently, we use $v^\star \doteq v^{\pi^\star}$ and $q^\star \doteq q^{\pi^\star}$ to denote the state value function and state-action value function arising from an optimal policy, respectively. As an optimal policy maximizes the value of each state, we have that

$$v^\star(x) = \max_\pi v^\pi(x), \quad q^\star(x, a) = \max_\pi q^\pi(x, a). \tag{10.28}$$

Simply optimizing over each policy is not a good idea as there are m^n deterministic policies in total. It turns out that we can do much better.

10.3.1 Greedy Policies

Consider a policy that acts greedily according to the immediate return. It is fairly obvious that this policy will not perform well because the agent might never get to high-reward states. But what if someone could tell us not just the immediate return, but the long-term value of the states our agent can reach in a single step? If we knew the value of each state our agent can reach, then we can simply pick the action that maximizes the expected value. We will make this approach precise in the next section.

This thought experiment suggests the definition of a greedy policy with respect to a value function.

Definition 10.12 (Greedy policy)**.** The *greedy policy* with respect to a state-action value function q is defined as

$$\pi_q(x) \doteq \arg\max_{a \in A} q(x, a). \tag{10.29}$$

Analogously, we define the *greedy policy* with respect to a state value function v,

$$\pi_v(x) \doteq \arg\max_{a \in A} r(x, a) + \gamma \sum_{x' \in X} p(x' \mid x, a) \cdot v(x'). \tag{10.30}$$

Exercise 10.13: Greedy policies

Show that if q and v arise from the same policy, that is, q is defined in terms of v as per eq. (10.9), then

$$\pi_v \equiv \pi_q. \tag{10.31}$$

This implies that we can use v and q interchangeably.

▷ *Solution*

10.3.2 *Bellman Optimality Equation*

Observe that following the greedy policy π_v, will lead us to a new value function v^{π_v}. With respect to this value function, we can again obtain a greedy policy, of which we can then obtain a new value function. In this way, the correspondence between greedy policies and value functions induces a cyclic dependency, which is visualized in fig. 10.3.

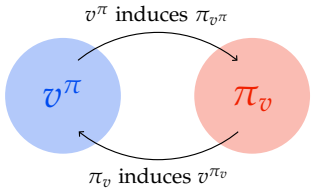


Figure 10.3: Cyclic dependency between **value function** and **greedy policy**.

It turns out that the optimal policy $\pi^\star$ is a fixed-point of this dependency. This is made precise by the following theorem.

Theorem 10.14 (Bellman's theorem)**.** *A policy $\pi^\star$ is optimal iff it is greedy with respect to its own value function. In other words, $\pi^\star$ is optimal iff $\pi^\star(x)$ is a distribution over the set* $\arg\max_{a\in A} q^\star(x,a)$.

In particular, if for every state there is a unique action that maximizes the state-action value function, the policy $\pi^\star$ is deterministic and unique,

$$\pi^\star(x) = \arg\max_{a\in A} q^\star(x,a). \tag{10.32}$$

Proof. It is a direct consequence of eq. (10.28) that a policy is optimal iff it is greedy with respect to $q^\star$. □

This theorem confirms our intuition from the previous section that greedily following an optimal value function is itself optimal. In particular, Bellman's theorem shows that there always exists an optimal policy which is deterministic and stationary.

We have seen, that $\pi^\star$ is a fixed-point of greedily picking the best action according to its state-action value function. The converse is also true.

Corollary 10.15. *The optimal value functions $v^\star$ and $q^\star$ are a fixed-point of the so-called* Bellman update,

$$v^\star(x) = \max_{a\in A} q^\star(x,a), \tag{10.33}$$

$$= \max_{a\in A} r(x,a) + \gamma \mathbb{E}_{x'|x,a}\left[v^\star(x')\right] \tag{10.34}$$

using the definition of the q-function (10.9)

$$q^\star(x,a) = r(x,a) + \gamma \mathbb{E}_{x'|x,a}\left[\max_{a'\in A} q^\star(x',a')\right]. \tag{10.35}$$

Proof. It follows from eq. (10.14) that

$$v^\star(x) = \mathbb{E}_{a\sim\pi^\star(x)}[q^\star(x,a)]. \tag{10.36}$$

Thus, as $\pi^\star$ is greedy with respect to $q^\star$, $v^\star(x) = \max_{a\in A} q^\star(x,a)$.

Equation (10.35) follows analogously from eq. (10.15). □

These equations are also called the *Bellman optimality equations*. Intuitively, the Bellman optimality equations express that the value of a state under an optimal policy must equal the expected return for the best action from that state. Bellman's theorem is also known as *Bellman's optimality principle*, which is a more general concept.

Bellman's optimality principle Bellman's optimality equations for MDPs are one of the main settings of Bellman's optimality principle. However, Bellman's optimality principle has many other important applications, for example in dynamic programming. Broadly speaking, Bellman's optimality principle says that optimal solutions to decision problems can be decomposed into optimal solutions to sub-problems.

Exercise 10.16: Optimal policies

Recall the example of "becoming rich and famous" from fig. 10.2.

1. Show that the policy $\pi \equiv S$, which we considered in exercise 10.8, is not optimal.
2. Instead, consider the policy

$$\pi' \equiv \begin{cases} A & \text{if poor and unknown} \\ S & \text{otherwise} \end{cases}$$

   and let $\gamma = 1/2$. Show that the (rounded) state-action value function $q^{\pi'}$ is as follows:

	save	advertise
poor, unknown	0.8	**1.6**
poor, famous	**4.5**	1.2
rich, famous	**17.8**	1.2
rich, unknown	**13.4**	0.2

   Shown in bold is the state value function $v^{\pi'}$.
3. Is the policy π' optimal?

▷ *Solution*

The two perspectives of Bellman's theorem naturally suggest two separate ways of finding the optimal policy. Policy iteration uses the perspective from eq. (10.32) of $\pi^\star$ as a fixed-point of the dependency between greedy policy and value function. In contrast, value iteration uses the perspective from eq. (10.33) of $v^\star$ as the fixed-point of the Bellman update. Another approach which we will not discuss here is to use a linear program where the Bellman update is interpreted as a set of linear inequalities.

10.3.3 Policy Iteration

Starting from an arbitrary initial policy, policy iteration as shown in alg. 10.17 uses the Bellman expectation equation to compute the value function of that policy (as we have discussed in section 10.2) and then chooses the greedy policy with respect to that value function as its next iterate.

Let π_t be the policy after t iterations. We will now show that policy

Algorithm 10.17: Policy iteration

```
initialize π (arbitrarily)
repeat
    compute v^π
    compute π_{v^π}
    π ← π_{v^π}
until converged
```

iteration converges to the optimal policy. The proof is split into two parts. First, we show that policy iteration improves policies monotonically. Then, we will use this fact to show that policy iteration converges.

Lemma 10.18 (Monotonic improvement of policy iteration)**.** *We have,*

- $v^{\pi_{t+1}}(x) \geq v^{\pi_t}(x)$ *for all* $x \in X$*; and*
- $v^{\pi_{t+1}}(x) > v^{\pi_t}(x)$ *for at least one* $x \in X$*, unless* $v^{\pi_t} \equiv v^\star$*.*

Proof. We consider the Bellman update from (10.33) as the mapping $\boldsymbol{B}^\star : \mathbb{R}^n \to \mathbb{R}^n$,

$$(\boldsymbol{B}^\star \boldsymbol{v})(x) \doteq \max_{a \in A} q(x, a), \tag{10.37}$$

where q is the state-action value function corresponding to the state value function $\boldsymbol{v} \in \mathbb{R}^n$. Recall that after obtaining v^{π_t}, policy iteration first computes the greedy policy w.r.t. v^{π_t}, $\pi_{t+1} \doteq \pi_{v^{\pi_t}}$, and then computes its value function $v^{\pi_{t+1}}$.

To establish the (weak) monotonic improvement of policy iteration, we consider a fixed-point iteration (cf. alg. 10.9) of $\boldsymbol{v}^{\pi_{t+1}}$ initialized by $\boldsymbol{v}^{\pi_t}$. We denote the iterates by $\tilde{\boldsymbol{v}}_t$, in particular, we have that $\tilde{\boldsymbol{v}}_0 = \boldsymbol{v}^{\pi_t}$ and $\lim_{t\to\infty} \tilde{\boldsymbol{v}}_t = \boldsymbol{v}^{\pi_{t+1}}$.[4] First, observe that for the first iteration of fixed-point iteration,

[4] using the convergence of fixed-point iteration (10.25)

$$\begin{aligned}
\tilde{v}_1(x) &= (\boldsymbol{B}^\star \boldsymbol{v}^{\pi_t})(x) && \text{using that } \pi_{t+1} \text{ is greedy wrt. } v^{\pi_t} \\
&= \max_{a \in A} q^{\pi_t}(x, a) && \text{using the definition of the Bellman update (10.37)} \\
&\geq q^{\pi_t}(x, \pi_t(x)) \\
&= v^{\pi_t}(x) \\
&= \tilde{v}_0(x). && \text{using (10.17)}
\end{aligned}$$

Let us now consider a single iteration of fixed-point iteration. We have,

$$\tilde{v}_{t+1}(x) = r(x, \pi_{t+1}(x)) + \gamma \sum_{x' \in X} p(x' \mid x, \pi_{t+1}(x)) \cdot \tilde{v}_t(x'). \qquad \text{using the definition of } \tilde{v}_{t+1} \text{ (10.21)}$$

Using an induction on t, we conclude,

$$\geq r(x, \pi_{t+1}(x)) + \gamma \sum_{x' \in X} p(x' \mid x, \pi_{t+1}(x)) \cdot \tilde{v}_{t-1}(x') \quad \text{using the induction hypothesis, } \tilde{v}_t(x') \geq \tilde{v}_{t-1}(x')$$
$$= \tilde{v}_t(x).$$

This establishes the first claim,

$$\boldsymbol{v}^{\pi_{t+1}} = \lim_{t \to \infty} \tilde{\boldsymbol{v}}_t \geq \tilde{\boldsymbol{v}}_0 = \boldsymbol{v}^{\pi_t}. \tag{10.38}$$

For the second claim, recall from Bellman's theorem (10.33) that $v^\star$ is a (unique) fixed-point of the Bellman update $\boldsymbol{B}^\star$.[5] In particular, we have $v^{\pi_{t+1}} \equiv v^{\pi_t}$ if and only if $v^{\pi_{t+1}} \equiv v^{\pi_t} \equiv v^\star$. In other words, if $v^{\pi_t} \not\equiv v^\star$ then eq. (10.38) is strict for at least one $x \in X$ and $v^{\pi_{t+1}} \not\equiv v^{\pi_t}$. This proves the strict monotonic improvement of policy iteration. □

[5] We will show in eq. (10.39) that $\boldsymbol{B}^\star$ is a contraction, implying that $v^\star$ is the *unique* fixed-point of $\boldsymbol{B}^\star$.

Theorem 10.19 (Convergence of policy iteration)**.** *For finite Markov decision processes, policy iteration converges to an optimal policy.*

Proof. Finite Markov decision processes only have a finite number of deterministic policies (albeit exponentially many). Observe that policy iteration only considers deterministic policies, and recall that there is an optimal policy that is deterministic. As the value of policies strictly increase in each iteration until an optimal policy is found, policy iteration must converge in finite time. □

It can be shown that policy iteration converges to an exact solution in a polynomial number of iterations (Ye, 2011). Each iteration of policy iteration requires computing the value function, which we have seen to be of cubic complexity in the number of states.

10.3.4 Value Iteration

As we have mentioned, another natural approach of finding the optimal policy is to interpret $v^\star$ as the fixed point of the Bellman update. Recall our definition of the Bellman update from eq. (10.37),

$$(\boldsymbol{B}^\star \boldsymbol{v})(x) = \max_{a \in A} q(x, a),$$

where q was the state-action value function associated with the state value function $\boldsymbol{v}$. The value iteration algorithm is shown in alg. 10.20.

We will now prove the convergence of value iteration using the fixed-point interpretation.

Theorem 10.21 (Convergence of value iteration)**.** *Value iteration converges to an optimal policy.*

Algorithm 10.20: Value iteration

initialize $v(x) \leftarrow \max_{a \in A} r(x,a)$ for each $x \in X$
for $t = 1$ **to** ∞ **do**
  $v(x) \leftarrow (\boldsymbol{B}^\star \boldsymbol{v})(x) = \max_{a \in A} q(x,a)$ for each $x \in X$
choose π_v

Proof. Clearly, value iteration converges if $v^\star$ is the unique fixed-point of $\boldsymbol{B}^\star$. We already know from Bellman's theorem (10.33) that $v^\star$ is a fixed-point of $\boldsymbol{B}^\star$. It remains to show that it is indeed the unique fixed-point.

Analogously to our proof of the convergence of fixed-point iteration to the value function v^π, we show that $\boldsymbol{B}^\star$ is a contraction. Fix arbitrary $\boldsymbol{v}, \boldsymbol{v}' \in \mathbb{R}^n$, then

$$
\begin{aligned}
\left\| \boldsymbol{B}^\star \boldsymbol{v} - \boldsymbol{B}^\star \boldsymbol{v}' \right\|_\infty &= \max_{x \in X} \left| (\boldsymbol{B}^\star \boldsymbol{v})(x) - (\boldsymbol{B}^\star \boldsymbol{v}')(x) \right| && \text{using the definition of the } \ell_\infty \text{ norm (10.23)} \\
&= \max_{x \in X} \left| \max_{a \in A} q(x,a) - \max_{a \in A} q'(x,a) \right| && \text{using the definition of the Bellman update (10.37)} \\
&\leq \max_{x \in X} \max_{a \in A} \left| q(x,a) - q'(x,a) \right| && \text{using } |\max_x f(x) - \max_x g(x)| \leq \max_x |f(x) - g(x)| \\
&\leq \gamma \max_{x \in X} \max_{a \in A} \sum_{x' \in X} p(x' \mid x, a) \left| \boldsymbol{v}(x') - \boldsymbol{v}'(x') \right| && \text{using the definition of the Q-function (10.9) and } |\textstyle\sum_i a_i| \leq \sum_i |a_i| \\
&\leq \gamma \left\| \boldsymbol{v} - \boldsymbol{v}' \right\|_\infty && \text{using } \textstyle\sum_{x' \in X} p(x' \mid x, a) = 1 \text{ and } |\boldsymbol{v}(x') - \boldsymbol{v}'(x')| \leq \|\boldsymbol{v} - \boldsymbol{v}'\|_\infty
\end{aligned} \tag{10.39}
$$

where q and q' are the state-action value functions associated with $\boldsymbol{v}$ and $\boldsymbol{v}'$, respectively. By eq. (10.22), $\boldsymbol{B}^\star$ is a contraction and by Banach's fixed-point theorem $v^\star$ is its unique fixed-point. □

Remark 10.22: Value iteration as a dynamic program

Let us denote by v_t the value function estimate after the t-th iteration. Observe that $v_t(x)$ corresponds to the maximum expected reward when starting in state x and the "world ends" after t time steps. In particular, v_0 corresponds to the maximum immediate reward. This suggests a different perspective on value iteration (akin to dynamic programming) where in each iteration we extend the time horizon of our approximation by one time step.

Value iteration converges to an ϵ-optimal solution in a polynomial number of iterations. Unlike policy iteration, value iteration does not converge to an exact solution in general. Recalling the update rule of value iteration, its main benefit is that each iteration only requires a sum over all possible actions a in state x and a sum over all reachable

states x' from x. In sparse Markov decision processes,[6] an iteration of value iteration can be performed in (virtually) constant time.

[6] Sparsity refers to the interconnectivity of the state space. When only few states are reachable from any state, we call an MDP sparse.

Exercise 10.23: Linear convergence of policy iteration

Denote by π_t the policy obtained by policy iteration after t iterations. Use that the Bellman operator $\boldsymbol{B}^\star$ is a contraction with the unique fixed-point $\boldsymbol{v}^\star$ to show that

$$\|\boldsymbol{v}^{\pi_t} - \boldsymbol{v}^\star\|_\infty \leq \gamma^t \|\boldsymbol{v}^{\pi_0} - \boldsymbol{v}^\star\|_\infty \tag{10.40}$$

where $\boldsymbol{v}^\pi$ and $\boldsymbol{v}^\star$ are vector representations of the functions v^π and $v^\star$, respectively.

Hint: Recall from lemma 10.18 that $\boldsymbol{v}^{\pi_{t+1}} \geq \boldsymbol{B}^\star \boldsymbol{v}^{\pi_t} \geq \boldsymbol{v}^{\pi_t}$.

▷ *Solution*

Exercise 10.24: Reward modification

A key technique for solving sequential decision problems is the modification of reward functions that leaves the optimal policy unchanged while improving sample efficiency or convergence rates. This exercise looks at simple ways of modifying rewards and understanding how these modifications affect the optimal policy.

Consider two Markov decision processes $M \doteq (X, A, p, r)$ and $M' \doteq (X, A, p, r')$ where the reward function r is modified to obtain r', and the rewards are bounded and discounted by the discount factor $\gamma \in [0, 1)$. Let $\pi^\star_M$ be the optimal policy for M.

1. Suppose $r'(x) = \alpha r(x)$, where $\alpha > 0$. Show that the optimal policy $\pi^\star$ of M is also an optimal policy of M'.
2. Given a modification of the form $r'(x) = r(x) + c$, where $c > 0$ is a constant scalar, show that the optimal policy $\pi^\star_M$ can be different from $\pi^\star_{M'}$.
3. Another way of modifying the reward function is through *reward shaping* where one supplies additional rewards to the agent to guide the learning process. When one has no knowledge of the underlying transition dynamics p, a commonly used transformation is $r'(x, x') = r(x, x') + f(x, x')$ where f is a *potential-based* shaping function defined as

$$f(x, x') \doteq \gamma\phi(x') - \phi(x), \quad \phi : X \to \mathbb{R}. \tag{10.41}$$

   Show that the optimal policy remains unchanged under this definition of f.

▷ *Solution*

Readings

For a more extensive overview of finite Markov decision processes, refer to chapters 3 and 4 of "Reinforcement learning: An introduction" (Sutton and Barto, 2018) or chapter 17 of "Artificial intelligence: a modern approach" (Russell and Norvig, 2002).

10.4 *Partial Observability*

So far we have focused on the fully observable setting. That is, at any time, our agent knows its current state. We have seen that we can efficiently find the optimal policy (as long as the Markov decision process is finite).

We have already encountered the partially observable setting in chapter 3, where we discussed Bayesian filtering and Kalman filters. In this section, we consider how Markov decision processes can be extended to a partially observable setting where the agent can only access noisy observations Y_t of its state X_t.

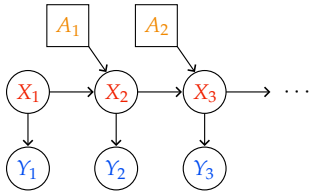


Figure 10.4: Directed graphical model of a partially observable Markov decision process with hidden states X_t, observables Y_t, and actions A_t.

Definition 10.25 (Partially observable Markov decision process, POMDP)**.** Similarly to a Markov decision process, a *partially observable Markov decision process* is specified by

- a set of *states* X,
- a set of *actions* A,
- *transition probabilities* $p(x' \mid x, a)$, and
- a *reward function* $r : X \times A \to \mathbb{R}$.

Additionally, it is specified by

- a set of *observations* Y, and
- *observation probabilities*

$$o(y \mid x) \doteq \mathbb{P}(Y_t = y \mid X_t = x). \tag{10.42}$$

Whereas MDPs are controlled Markov chains, POMDPs are controlled hidden Markov models.

Remark 10.26: Hidden Markov models

A hidden Markov model is a Markovian process with unobservable states X_t and observations Y_t that depend on X_t in a known way.

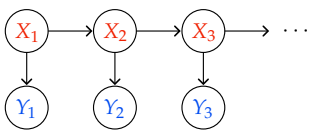


Figure 10.5: Directed graphical model of a hidden Markov model with hidden states X_t and observables Y_t.

Definition 10.27 (Hidden Markov model, HMM)**.** A *hidden Markov model* is specified by

- a set of *states* X,
- *transition probabilities* $p(x' \mid x) \doteq \mathbb{P}(X_{t+1} = x' \mid X_t = x)$ (also

called *motion model*), and
- a *sensor model* $o(y \mid x) \doteq \mathbb{P}(Y_t = y \mid X_t = x)$.

Following from its directed graphical model shown in fig. 10.5, its joint probability distribution factorizes into

$$\mathbb{P}(x_{1:t}, y_{1:t}) = \mathbb{P}(x_1) \cdot o(y_1 \mid x_1) \cdot \prod_{i=2}^{t} p(x_i \mid x_{i-1}) \cdot o(y_i \mid x_i). \tag{10.43}$$

Observe that a Kalman filter can be viewed as a hidden Markov model with conditional linear Gaussian motion and sensor models and a Gaussian prior on the initial state. In particular, the tasks of *filtering*, *smoothing*, and *predicting* which we discussed extensively in chapter 3 are also of interest for hidden Markov models.

A widely used application of hidden Markov models is to find the most likely sequence (also called *most likely explanation*) of hidden states $x_{1:t}$ given a series of observations $y_{1:t}$,[7] that is, to find

$$\arg\max_{x_{1:t}} \mathbb{P}(x_{1:t} \mid y_{1:t}). \tag{10.44}$$

This task can be solved in linear time by a simple backtracking algorithm known as the *Viterbi algorithm*.

[7] This is useful in many applications such as speech recognition, decoding data that was transmitted over a noisy channel, beat detection, and many more.

POMDPs are a very powerful model, but very hard to solve in general. POMDPs can be reduced to a Markov decision process with an enlarged state space. The key insight is to consider an MDP whose states are the *beliefs*,

$$b_t(x) \doteq \mathbb{P}(X_t = x \mid y_{1:t}, a_{1:t-1}), \tag{10.45}$$

about the current state in the POMDP. In other words, the states of the MDP are probability distributions over the states of the POMDP. We will make this more precise in the following.

Let us assume that our prior belief about the state of our agent is given by $b_0(x) \doteq \mathbb{P}(X_0 = x)$. Keeping track of how beliefs change over time is known as *Bayesian filtering*, which we already encountered in section 3.1. Given a prior belief b_t, an action taken a_t, and a new observation y_{t+1}, the belief state can be updated as follows,

$$\begin{aligned} b_{t+1}(x) &= \mathbb{P}(X_{t+1} = x \mid y_{1:t+1}, a_{1:t}) && \text{by the definition of beliefs (10.45)} \\ &= \frac{1}{Z}\mathbb{P}(y_{t+1} \mid X_{t+1} = x)\mathbb{P}(X_{t+1} = x \mid y_{1:t}, a_{1:t}) && \text{using Bayes' rule (1.59)} \\ &= \frac{1}{Z} o(y_{t+1} \mid x)\mathbb{P}(X_{t+1} = x \mid y_{1:t}, a_{1:t}) && \text{using the definition of observation probabilities (10.42)} \end{aligned}$$

$$= \frac{1}{Z} o(y_{t+1} \mid x) \sum_{x' \in X} p(x \mid x', a_t) \mathbb{P}\big(X_t = x' \mid y_{1:t}, a_{1:t-1}\big)$$ by conditioning on the previous state x', noting a_t does not influence X_t

$$= \frac{1}{Z} o(y_{t+1} \mid x) \sum_{x' \in X} p(x \mid x', a_t) b_t(x') \tag{10.46}$$ using the definition of beliefs (10.45)

where

$$Z \doteq \sum_{x \in X} o(y_{t+1} \mid x) \sum_{x' \in X} p(x \mid x', a_t) b_t(x'). \tag{10.47}$$

Thus, the updated belief state is a deterministic mapping from the previous belief state depending only on the (random) observation y_{t+1} and the taken action a_t. Note that this obeys a Markovian structure of transition probabilities with respect to the beliefs b_t.

The sequence of belief-states defines the sequence of random variables $(B_t)_{t \in \mathbb{N}_0}$,

$$B_t \doteq X_t \mid y_{1:t}, a_{1:t-1}, \tag{10.48}$$

where the (state-)space of all beliefs is the (infinite) space of all probability distributions over X,[8]

$$\mathcal{B} \doteq \Delta^X \doteq \Big\{ \boldsymbol{b} \in \mathbb{R}^{|X|} : \boldsymbol{b} \geq \mathbf{0}, \textstyle\sum_{i=1}^{|X|} \boldsymbol{b}(i) = 1 \Big\}. \tag{10.49}$$

[8] This definition naturally extends to continuous state spaces $\mathcal{X}$.

A Markov decision process, where every belief corresponds to a state is called a belief-state MDP.

Definition 10.28 (Belief-state Markov decision process)**.** Given a POMDP, the corresponding *belief-state Markov decision process* is a Markov decision process specified by

- the *belief space* $\mathcal{B} \doteq \Delta^X$ depending on the *hidden states* X,
- the set of *actions* A,
- *transition probabilities*

$$\tau(b' \mid b, a) \doteq \mathbb{P}\big(B_{t+1} = b' \mid B_t = b, A_t = a\big), \tag{10.50}$$

- and *rewards*

$$\rho(b, a) \doteq \mathbb{E}_{x \sim b}[r(x, a)] = \sum_{x \in X} b(x) r(x, a). \tag{10.51}$$

It remains to derive the transition probabilities τ in terms of the original POMDP. We have,

$$\begin{aligned} \tau(b_{t+1} \mid b_t, a_t) &= \mathbb{P}(b_{t+1} \mid b_t, a_t) \\ &= \sum_{y_{t+1} \in Y} \mathbb{P}(b_{t+1} \mid b_t, a_t, y_{t+1}) \mathbb{P}(y_{t+1} \mid b_t, a_t). \end{aligned} \tag{10.52}$$ by conditioning on $y_{t+1} \in Y$

Using the Markovian structure of the belief updates, we naturally set,

$$\mathbb{P}(b_{t+1} \mid b_t, a_t, y_{t+1}) \doteq \begin{cases} 1 & \text{if } b_{t+1} \text{ matches the belief update of} \\ & \text{eq. (10.45) given } b_t, a_t, \text{ and } y_{t+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{10.53}$$

The final missing piece is the likelihood of an observation y_{t+1} given the prior belief b_t and action a_t, which using our interpretation of beliefs corresponds to

$$\begin{aligned} \mathbb{P}(y_{t+1} \mid b_t, a_t) &= \mathbb{E}_{x \sim b_t}\Big[\mathbb{E}_{x' \mid x, a_t}\big[\mathbb{P}\big(y_{t+1} \mid X_{t+1} = x'\big)\big]\Big] \\ &= \mathbb{E}_{x \sim b_t}\Big[\mathbb{E}_{x' \mid x, a_t}\big[o(y_{t+1} \mid x')\big]\Big] && \text{using the definition of observation probabilities (10.42)} \\ &= \sum_{x \in X} b_t(x) \sum_{x' \in X} p(x' \mid x, a_t) \cdot o(y_{t+1} \mid x'). \end{aligned} \tag{10.54}$$

In principle, we can now apply arbitrary algorithms for planning in MDPs to POMDPs. Of course, the problem is that there are infinitely many beliefs, even for a finite state space X.[9] The belief-state MDP has therefore an infinitely large belief space $\mathcal{B}$. Even when only planning over finite horizons, exponentially many beliefs can be reached. So the belief space blows-up very quickly.

We will study MDPs with large state spaces (where transition dynamics and rewards are unknown) in chapters 12 and 13. Similar methods can also be used to approximately solve POMDPs.

A key idea in approximate solutions to POMDPs is that most belief states are never reached. A common approach is to discretize the belief space by sampling or by applying a dimensionality reduction. Examples are *point-based value iteration* (PBVI) and *point-based policy iteration* (PBPI) (Shani et al., 2013).

Readings

For a more extensive overview of POMDPs and algorithms for solving POMDPs refer to section 17.4 of "Artificial intelligence: a modern approach" (Russell and Norvig, 2002).

Even though we focus on the fully observed setting throughout this course, the partially observed setting can be reduced to the fully observed setting with very large state spaces. In the next chapter, we will consider learning and planning in unknown Markov decision processes (i.e., reinforcement learning) for small state spaces. The setting of small state and action spaces is also known as the *tabular setting*.

[9] You can think of an $|X|$-dimensional space. Here, all points whose coordinates sum to 1 correspond to probability distributions (i.e., beliefs) over the hidden states X. The convex hull of these points is also known as the $(|X|-1)$-dimensional *probability simplex* (cf. remark 1.10). Now, by definition of the $(|X|-1)$-dimensional probability simplex as a polytope in $|X|-1$ dimensions, we can conclude that its boundary consists of infinitely many points in $|X|$ dimensions. Noting that these points corresponded to the probability distributions on $|X|$, we conclude that there are infinitely many such distributions.

Then, in the final two chapters, we will consider approximate methods for large state and action spaces. In particular, in section 13.1, we will revisit the problem of probabilistic planning in known Markov decision processes, but with continuous state and action spaces.

11

Tabular Reinforcement Learning

11.1 The Reinforcement Learning Problem

Reinforcement learning is concerned with probabilistic planning in unknown environments. This extends our study of known environments in the previous chapter. Those environments are still modeled by Markov decision processes, but in reinforcement learning, we do not know the dynamics p and rewards r in advance. Hence, reinforcement learning is at the intersection of the theories of probabilistic planning (i.e., Markov decision processes) and learning (e.g., multi-armed bandits), which we covered extensively in the previous chapters.

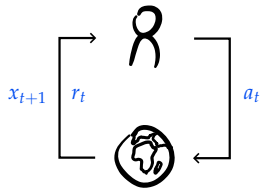


Figure 11.1: In reinforcement learning, an agent interacts with its environment in a sequence of rounds. After playing an action a_t, it observes rewards r_t and its new state x_{t+1}. The agent then uses this information to learn a model of the world.

We will continue to focus on the fully observed setting, where the agent knows its current state. As we have seen in the previous section, the partially observed setting corresponds to a fully observed setting with an enlarged state space. In this chapter, we will begin by considering reinforcement learning with small state and action spaces. This setting is often called the *tabular setting*, as the value functions can be computed exhaustively for all states and stored in a table.

Clearly, the agent needs to trade exploring and learning about the environment with exploiting its knowledge to maximize rewards. Thus, the exploration-exploitation dilemma, which was at the core of Bayesian optimization (see section 9.1), also plays a crucial role in reinforcement learning. In fact, Bayesian optimization can be viewed as reinforcement learning with a fixed state and a continuous action space: In each round, the agent plays an action, aiming to find the action that maximizes the reward. However, playing the same action multiple times yields the same reward, implying that we remain in a single state. In the context of Bayesian optimization, we used "regret" as performance metric: in the jargon of planning, minimizing regret corresponds to maximizing the cumulative reward.

Another key challenge of reinforcement learning is that the observed data is dependent on the played actions. This is in contrast to the setting of supervised learning that we have been considering in earlier chapters, where the data is sampled independently.

11.1.1 Trajectories

The data that the agent collects is modeled using so-called trajectories.

Definition 11.1 (Trajectory)**.** A *trajectory* τ is a (possibly infinite) sequence,

$$\tau \doteq (\tau_0, \tau_1, \tau_2, \dots), \tag{11.1}$$

of *transitions*,

$$\tau_i \doteq (x_i, a_i, r_i, x_{i+1}), \tag{11.2}$$

where $x_i \in X$ is the starting state, $a_i \in A$ is the played action, $r_i \in \mathbb{R}$ is the attained reward, and $x_{i+1} \in X$ is the ending state.

In the context of learning a dynamics and rewards model, x_i and a_i can be understood as inputs, and r_i and x_{i+1} can be understood as labels of a regression problem.

Crucially, the newly observed states x_{t+1} and the rewards r_t (across multiple transitions) are conditionally independent given the previous states x_t and actions a_t. This follows directly from the Markovian structure of the underlying Markov decision process.[1] Formally, we have,

$$X_{t+1} \perp X_{t'+1} \mid X_t, X_{t'}, A_t, A_{t'}, \tag{11.3a}$$

$$R_t \perp R_{t'} \mid X_t, X_{t'}, A_t, A_{t'}, \tag{11.3b}$$

for any $t, t' \in \mathbb{N}_0$. In particular, if $x_t = x_{t'}$ and $a_t = a_{t'}$, then x_{t+1} and $x_{t'+1}$ are independent samples according to the transition model $p(X_{t+1} \mid x_t, a_t)$. Analogously, if $x_t = x_{t'}$ and $a_t = a_{t'}$, then r_t and $r_{t'}$ are independent samples of the reward model $r(x_t, a_t)$. As we will see later in this chapter and especially in chapter 13, this independence property is crucial for being able to learn about the underlying Markov decision process. Notably, this implies that we can apply the law of large numbers (1.84) and Hoeffding's inequality (1.88) to our estimators of both quantities.

The collection of data is commonly classified into two settings. In the *episodic setting*, the agent performs a sequence of "training" rounds (called *episodes*). In the beginning of each episode, the agent is reset

[1] Recall the Markov property (6.6), which assumes that in the underlying Markov decision process (i.e., in our environment) the future state of an agent is independent of past states given the agent's current state. This is commonly called a Markovian structure. From this Markovian structure, we gather that repeated encounters of state-action pairs result in independent trials of the transition model and rewards.

to some initial state. In contrast, in the *continuous setting* (or non-episodic / online setting), the agent learns online. Especially, every action, every reward, and every state transition counts.

The episodic setting is more applicable to an agent playing a computer game. That is, the agent is performing in a simulated environment that is easy to reset. The continuous setting is akin to an agent that is deployed to the "real world". In principle, real-world agents can be trained in simulated environments before being deployed. However, this bears the risk of learning to exploit or rely on features of the simulated environment that are not present in the real environment. Sometimes, using a simulated environment for training is downright impossible, as the real environment is too complex.

11.1.2 On-policy and Off-policy Methods

Another important distinction in how data is collected, is the distinction between on-policy and off-policy methods. As the names suggest, *on-policy* methods are used when the agent has control over its own actions, in other words, the agent can freely choose to follow any policy. Being able to follow a policy is helpful, for example because it allows the agent to experiment with trading exploration and exploitation.

In contrast, *off-policy* methods can be used even when the agent cannot freely choose its actions. Off-policy methods are therefore able to make use of purely observational data. This might be data that was collected by another agent, a fixed policy, or during a previous episode. Off-policy methods are therefore more *sample-efficient* than on-policy methods. This is crucial, especially in settings where conducting experiments (i.e., collecting new data) is expensive.

11.2 Model-based Approaches

Approaches to reinforcement learning are largely categorized into two classes. *Model-based* approaches aim to learn the underlying Markov decision process. More concretely, they learn models of the dynamics p and rewards r. They then use these models to perform planning (i.e., policy optimization) in the underlying Markov decision process. In contrast, *model-free* approaches learn the value function directly. We begin by discussing model-based approaches to the tabular setting. In section 11.4, we cover model-free approaches.

11.2.1 Learning the Underlying Markov Decision Process

Recall that the underlying Markov decision process was specified by its dynamics $p(x' \mid x, a)$ that correspond to the probability of entering

state $x' \in X$ when playing action $a \in A$ from state $x \in X$, and its rewards $r(x, a)$ for playing action $a \in A$ in state $x \in X$. A natural first idea is to use maximum likelihood estimation to approximate these quantities.

We can think of each transition $x' \mid x, a$ as sampling from a categorical random variable of which we want to estimate the success probabilities for landing in each of the states. Therefore, as we have seen in example 1.49, the MLE of the dynamics model coincides with the sample mean,

$$\hat{p}(x' \mid x, a) = \frac{N(x' \mid x, a)}{N(a \mid x)} \tag{11.4}$$

where $N(x' \mid x, a)$ counts the number of transitions from state x to state x' when playing action a and $N(a \mid x)$ counts the number of transitions that start in state x and play action a (regardless of the next state). Similarly, for the rewards model, we obtain the following maximum likelihood estimate (i.e., sample mean),

$$\hat{r}(x, a) = \frac{1}{N(a \mid x)} \sum_{\substack{t=0 \\ x_t = x \\ a_t = a}}^{\infty} r_t. \tag{11.5}$$

It is immediate that both estimates are unbiased as both correspond to a sample mean.

Still, for the models of our environment to become accurate, our agent needs to visit *each* state-action pair (x, a) numerous times. Note that our estimators for dynamics and rewards are only well-defined when we visit the corresponding state-action pair at least once. However, in a stochastic environment, a single visit will likely not result in an accurate model. We can use Hoeffding's inequality (1.88) to gauge how accurate the estimates are after only a limited number of visits.

11.3 Balancing Exploration and Exploitation

The next natural question is how to use our current model of the environment to pick actions such that exploration and exploitation are traded effectively. This is what we will consider next.

Given the estimated MDP given by $\hat{p}$ and $\hat{r}$, we can compute the optimal policy using either policy iteration or value iteration. For example, using value iteration, we can compute the optimal state-action value function $Q^\star$ within the *estimated* MDP, and then employ the greedy policy

$$\pi(x) = \arg\max_{a \in A} Q^\star(x, a). \tag{11.6}$$

Recall from eq. (10.32) that this corresponds to always picking the best action under the *current* model (that is, π is the optimal policy). But since the model is inaccurate, while potentially quickly generating some reward, we will likely get stuck in a suboptimal state.

11.3.1 ϵ-greedy

Consider the other extreme: If we always pick a random action, we will eventually(!) estimate the dynamics and rewards correctly, yet we will do extremely poorly in terms of maximizing rewards along the way. To trade exploration and exploitation, a natural idea is to balance these two extremes.

Arguably, the simplest idea is the following: At each time step, throw a biased coin. If this coin lands heads, we pick an action uniformly at random among all actions. If the coin lands tails, we pick the best action under our current model. This algorithm is called *ϵ-greedy*, where the probability of a coin landing heads at time t is ϵ_t.

Algorithm 11.2: ϵ-greedy

```
for t = 0 to ∞ do
    sample u ∈ Unif([0, 1])
    if u ≤ ε_t then pick action uniformly at random among all actions
    else pick best action under the current model
```

The ϵ-greedy algorithm provides a general framework for addressing the exploration-exploitation dilemma. When the underlying MDP is learned using Monte Carlo estimation as we discussed in section 11.2.1, the resulting algorithm is known as *Monte Carlo control*. However, the same framework can also be used in the model-free setting where we pick the best action without estimating the full underlying MDP. We discuss this approach in greater detail in section 11.4.

Amazingly, this simple algorithm already works quite well. Nevertheless, it can clearly be improved. The key problem of ϵ-greedy is that it explores the state space in an uninformed manner. In other words, it explores ignoring all past experience. It thus does not eliminate clearly suboptimal actions. This is a problem, especially as we typically have many state-action pairs and recalling that we have to explore each such pair many times to learn an accurate model.

Remark 11.3: Asymptotic convergence

It can be shown that Monte Carlo control converges to an optimal policy (albeit slowly) almost surely when the learned policy is "greedy in the limit with infinite exploration".

Definition 11.4 (Greedy in the limit with infinite exploration, GLIE)**.** A sequence of policies π_t is said to be *greedy in the limit with infinite exploration* if

1. all state-action pairs are explored infinitely many times,[2]

$$\lim_{t\to\infty} N_t(x,a) = \infty \quad \text{and} \tag{11.7}$$

2. the policy converges to a greedy policy,

$$\lim_{t\to\infty} \pi_t(a \mid x) = \mathbb{1}\{a = \arg\max_{a'\in A} Q_t^\star(x,a')\} \tag{11.8}$$

where we denote by $N_t(x,a)$ the number of transitions from state x playing action a until time t, and $Q_t^\star$ is the optimal state-action value function in the estimated MDP at time t.

[2] That all state-action pairs are chosen is a fundamental requirement. There is no reason why any algorithm would converge to the true value function for *all* states when it only sees some state-action pairs finitely many times, or even not at all.

Note that ϵ-greedy is GLIE with probability 1 if the sequence $(\epsilon_t)_{t\in\mathbb{N}_0}$ satisfies the RM-conditions (1.109), e.g., if $\epsilon_t = 1/t$.

Theorem 11.5 (Convergence of Monte Carlo control)**.** *GLIE Monte Carlo control converges to an optimal policy with probability* 1*.*

Intuitively, the probability of exploration converges to zero, and hence, the policy will "eventually coincide" with the greedy policy. Moreover, the greedy policy will "eventually coincide" with the optimal policy due to an argument akin to the convergence of policy iteration,[3] and using that each state-action pair is visited infinitely often.

[3] see lemma 10.18 and theorem 10.19

11.3.2 Softmax Exploration

An alternative to using ϵ-greedy for trading between greedy exploitation and uniform exploration is the so-called *softmax exploration* or *Boltzmann exploration*. Given the agent is in state x, we pick action a with probability,

$$\pi_T(a \mid x) \propto \exp\left(\frac{1}{T} Q^\star(x,a)\right), \tag{11.9}$$

which is the Gibbs distribution with temperature parameter $T > 0$. Observe that for $T \to 0$, softmax exploration corresponds to greedily maximizing the Q-function (i.e., greedy exploitation), whereas for $T \to \infty$, softmax exploration explores uniformly at random. This can

outperform ϵ-greedy as the exploration is directed towards actions with larger estimated value.

11.3.3 Optimism

Recall from our discussion of multi-armed bandits in section 9.2.1 that a key principle in effectively trading exploration and exploitation is *optimism in the face of uncertainty*. Let us apply this principle to the reinforcement learning setting. The key idea is to assume that the dynamics and rewards model "work in our favor" until we have learned "good estimates" of the true dynamics and rewards.

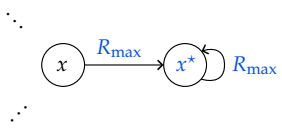


Figure 11.2: Illustration of the fairy-tale state of $R_{\max}$. If in doubt, the agent believes actions from the state x to lead to the fairy-tale state $x^\star$ with maximal rewards. This encourages the exploration of unknown states.

More formally, if $r(x, a)$ is unknown, we set $\hat{r}(x, a) = R_{\max}$, where $R_{\max}$ is the maximum reward our agent can attain during a single transition. Similarly, if $p(x' \mid x, a)$ is unknown, we set $\hat{p}(x^\star \mid x, a) = 1$, where $x^\star$ is a "fairy-tale state". The fairy-tale state corresponds to everything our agent could wish for, that is,

$$\hat{p}(x^\star \mid x^\star, a) = 1 \qquad \forall a \in A, \tag{11.10}$$

$$\hat{r}(x^\star, a) = R_{\max} \qquad \forall a \in A. \tag{11.11}$$

In practice, the decision of when to assume that the learned dynamics and reward models are "good enough" has to be tuned.

In using these optimistic estimates of p and r, we obtain an optimistic underlying Markov decision process that exhibits a bias towards exploration. In particular, the rewards attained in this MDP, are an upper bound of the true reward. The resulting algorithm is known as the $R_{\max}$ *algorithm*.

Algorithm 11.6: $R_{\max}$ algorithm

```
add the fairy-tale state x* to the Markov decision process
set r̂(x, a) = R_max for all x ∈ X and a ∈ A
set p̂(x* | x, a) = 1 for all x ∈ X and a ∈ A
compute the optimal policy π̂ for r̂ and p̂
for t = 0 to ∞ do
    execute policy π̂ (for some number of steps)
    for each visited state-action pair (x, a), update r̂(x, a)
    estimate transition probabilities p̂(x' | x, a)
    after observing "enough" transitions and rewards, recompute the
      optimal policy π̂ according the current model p̂ and r̂.
```

How many transitions are "enough"? We can use Hoeffding's inequality (1.88) to get a rough idea! The key here, is our observation from

eq. (11.3) that the transitions and rewards are conditionally independent given the state-action pairs since, as we have discussed in section 6.1.4 on the ergodic theorem, Hoeffding's inequality does not hold for dependent samples. In this case, Hoeffding's inequality tells us that for the absolute approximation error to be below ϵ with probability at least $1 - \delta$, we need

$$N(a \mid x) \geq \frac{R_{\max}^2}{2\epsilon^2} \log \frac{2}{\delta}. \tag{11.12}$$

see (1.89)

Lemma 11.7 (Exploration and exploitation of $R_{\max}$)**.** *Every T time steps, with high probability, $R_{\max}$ either*

- *obtains near-optimal reward; or*
- *visits at least one unknown state-action pair.*[4]

Here, T depends on the mixing time of the Markov chain induced by the optimal policy.

[4] Note that in the tabular setting, there are "only" polynomially many state-action pairs.

Theorem 11.8 (Convergence of $R_{\max}$, Brafman and Tennenholtz (2002))**.** *With probability at least $1 - \delta$, $R_{\max}$ reaches an ϵ-optimal policy in a number of steps that is polynomial in $|X|$, $|A|$, T, $1/\epsilon$, $1/\delta$, and $R_{\max}$.*

11.3.4 Challenges of Model-based Approaches

We have seen that the $R_{\max}$ algorithm performs remarkably well in the tabular setting. However, there are important computational limitations to the model-based approaches that we discussed so far.

First, observe that the (tabular) model-based approach requires us to store $\hat{p}(x' \mid x, a)$ and $\hat{r}(x, a)$ in a table. This table already has $\mathcal{O}(n^2 m)$ entries. Even though polynomial in the size of the state and action spaces, this quickly becomes unmanageable.

Second, the model-based approach requires us to "solve" the learned Markov decision processes to obtain the optimal policy (using policy or value iteration). As we continue to learn over time, we need to find the optimal policy many times. $R_{\max}$ recomputes the policy after each state-action pair is observed sufficiently often, so $\mathcal{O}(nm)$ times.

11.4 Model-free Approaches

In the previous section, we have seen that learning and remembering the model as well as planning within the estimated model can potentially be quite expensive in the model-based approach. We therefore turn to model-free methods that estimate the value function directly. Thus, they require neither remembering the full model nor planning (i.e., policy optimization) in the underlying Markov decision process.

We will, however, return to model-based methods in chapter 13 to see that promise lies in combining methods from model-based reinforcement learning with methods from model-free reinforcement learning.

A significant benefit to model-based reinforcement learning is that it is inherently off-policy. That is, any trajectory regardless of the policy used to obtain it can be used to improve the model of the underlying Markov decision process. In the model-free setting, this not necessarily true. By default, estimating the value function according to the data from a trajectory, will yield an estimate of the value function corresponding to the policy that was used to sample the data.

We will start by discussing on-policy methods and later see how the value function can be estimated off-policy.

11.4.1 On-policy Value Estimation

Let us suppose, our agent follows a fixed policy π. Then, the corresponding value function v^π is given as

$$\begin{aligned} v^\pi(x) &= r(x, \pi(x)) + \gamma \sum_{x' \in X} p(x' \mid x, \pi(x)) \cdot v^\pi(x') && \text{using the definition of the value function (10.7)} \\ &= \mathbb{E}_{R_0, X_1}[R_0 + \gamma v^\pi(X_1) \mid X_0 = x, A_0 = \pi(x)] && \text{interpreting the above expression as an expectation over the random quantities } R_0 \text{ and } X_1 \end{aligned} \tag{11.13}$$

Our first instinct might be to use a Monte Carlo estimate of this expectation. Due to the conditional independence of the transitions (11.3), Monte Carlo approximation does yield an unbiased estimate,

$$\approx r + \gamma v^\pi(x') \tag{11.14}$$

where the agent observed the transition (x, a, r, x'). Note that to estimate this expectation we use a single(!) sample,[5] unlike our previous applications of Monte Carlo sampling where we usually averaged over m samples. However, there is one significant problem in this approximation. Our approximation of v^π does in turn depend on the (unknown) true value of v^π!

[5] The idea is that we will use this approximation repeatedly as our agent collects new data to achieve the same effect as initially averaging over multiple samples.

The key idea is to use a bootstrapping estimate of the value function instead. That is, in place of the true value function v^π, we will use a "running estimate" V^π. In other words, whenever observing a new transition, we use our previous best estimate of v^π to obtain a new estimate V^π. We already encountered bootstrapping briefly in section 7.3.4 in the context of probabilistic ensembles in Bayesian deep learning. More generally, *bootstrapping* refers to approximating a true quantity (e.g., v^π) by using an empirical quantity (e.g., V^π), which itself is constructed using samples from the true quantity that is to be approximated.

Due to its use in estimating the value function, bootstrapping is a core concept to model-free reinforcement learning. Crucially, using a bootstrapping estimate generally results in biased estimates of the value function. Moreover, due to relying on a single sample, the estimates from eq. (11.14) tend to have very large variance.

The variance of the estimate is typically reduced by mixing new estimates of the value function with previous estimates using a learning rate α_t. This yields the *temporal-difference learning* algorithm.

Algorithm 11.9: Temporal-difference (TD) learning

initialize V^π arbitrarily (e.g., as $\mathbf{0}$)
for $t = 0$ **to** ∞ **do**
  follow policy π to obtain the transition (x, a, r, x')
  $V^\pi(x) \leftarrow (1 - \alpha_t) V^\pi(x) + \alpha_t (r + \gamma V^\pi(x'))$ // (11.15)

The update rule is sometimes written equivalently as

$$V^\pi(x) \leftarrow V^\pi(x) + \alpha_t (r + \gamma V^\pi(x') - V^\pi(x)). \tag{11.16}$$

Thus, the update to $V^\pi(x)$ is proportional to the learning rate and the difference between the previous estimate and the renewed estimate using the new observation.

Theorem 11.10 (Convergence of TD-learning, Jaakkola et al. (1993))**.** *If $(\alpha_t)_{t \in \mathbb{N}_0}$ satisfies the RM-conditions* (1.109) *and all state-action pairs are chosen infinitely often, then V^π converges to v^π with probability* 1*.*

Importantly, note that due to the Monte Carlo approximation of eq. (11.13) with respect to transitions attained by following policy π, TD-learning is fundamentally on-policy. That is, for the estimates V^π to converge to the true value function v^π, the transitions that are used for the estimation must follow policy π.

11.4.2 *SARSA: On-policy Control*

TD-learning merely estimates the value function of a fixed policy π. To find the optimal policy $\pi^\star$, we can use an analogue of policy iteration (see alg. 10.17). Here, it is more convenient to use an estimate of the state-action value function q^π which can be obtained analogously to the bootstrapping estimate of v^π (11.14),

$$\begin{aligned} q^\pi(x, a) &= r(x, a) + \gamma \sum_{x' \in X} p(x' \mid x, a) \sum_{a' \in A} \pi(a' \mid x') q^\pi(x', a') && \text{using Bellman's expectation equation (10.15)} \\ &= \mathbb{E}_{R_0, X_1, A_1}[R_0 + \gamma q^\pi(X_1, A_1) \mid X_0 = x, A_0 = a] && \text{interpreting the above expression as an expectation over } R_0, X_1 \text{ and } A_1 \end{aligned} \tag{11.17}$$

$$\approx r + \gamma q^\pi(x', a') \tag{11.18}$$

Monte Carlo approximation with a single sample

where the agent observed transitions (x, a, r, x') and (x', a', r', x'').

The update rule from TD-learning is therefore adapted to[6]

$$Q^\pi(x, a) \leftarrow (1 - \alpha_t) Q^\pi(x, a) + \alpha_t (r + \gamma Q^\pi(x', a')). \tag{11.19}$$

[6] Note that for deterministic policies π, $Q^\pi(x', a') = Q^\pi(x', \pi(x')) = V^\pi(x')$ if the transitions are obtained by following policy π.

This algorithm is known as *SARSA* (short for *state-action-reward-state-action*). Similar convergence guarantees to those of TD-learning can also be derived for SARSA.

Theorem 11.11 (Convergence of SARSA, Singh et al. (2000))**.** *If* $(\alpha_t)_{t \in \mathbb{N}_0}$ *satisfies the RM-conditions* (1.109) *and all state-action pairs are chosen infinitely often, then* Q^π *converges to* q^π *with probability* 1*.*

The policy iteration scheme to identify the optimal policy can be outlined as follows: In each iteration t, we estimate the value function q^{π_t} of policy π_t with the estimate Q^{π_t} obtained from SARSA. We then choose the greedy policy with respect to Q^{π_t} as the next policy π_{t+1}. However, due to the on-policy nature of SARSA, we cannot reuse any data between the iterations. Moreover, it turns out that in practice, when using only finitely many samples, this form of greedily optimizing Markov decision processes does not explore enough. At least partially, this can be compensated for by injecting noise when choosing the next action, e.g., by following an ϵ-greedy policy or using softmax exploration.

11.4.3 Off-policy Value Estimation

Consider the following slight adaptation of the derivation of SARSA (11.18),

$$\begin{aligned} q^\pi(x, a) &= r(x, a) + \gamma \sum_{x' \in X} p(x' \mid x, a) \sum_{a' \in A} \pi(a' \mid x') q^\pi(x', a') && \text{using Bellman's expectation equation (10.15)} \\ &= \mathbb{E}_{R_0, X_1}\left[R_0 + \gamma \sum_{a' \in A} \pi(a' \mid X_1) q^\pi(X_1, a') \mid X_0 = x, A_0 = a \right] && \text{interpreting the above expression as an expectation over } R_0 \text{ and } X_1 \\ &\approx r + \gamma \sum_{a' \in A} \pi(a' \mid x') q^\pi(x', a') && \text{Monte Carlo approximation with a single sample} \end{aligned} \tag{11.20, 11.21}$$

where the agent observed the transition (x, a, r, x'). This yields the update rule,

$$Q^\pi(x, a) \leftarrow (1 - \alpha_t) Q^\pi(x, a) + \alpha_t \left(r + \gamma \sum_{a' \in A} \pi(a' \mid x') Q^\pi(x', a') \right). \tag{11.22}$$

This adapted update rule *explicitly* chooses the subsequent action a' according to policy π whereas SARSA absorbs this choice into the Monte Carlo approximation. The algorithm has analogous convergence guarantees to those of SARSA.

Crucially, this algorithm is off-policy. That is, we can use transitions that were obtained according to *any* policy to estimate the value of a fixed policy π, which we may have never used! Perhaps this seems contradictory at first, but it is not. As noted, the key difference to the on-policy TD-learning and SARSA is that our estimate of the Q-function explicitly keeps track of the next-performed action. It does so for any action in any state. Moreover, note that the transitions that are due to the dynamics model and rewards are unaffected by the used policy. They merely depend on the originating state-action pair. We can therefore use the instances where other policies played action $\pi(x)$ in state x to estimate the performance of π.

11.4.4 Q-learning: Off-policy Control

It turns out that there is a way to estimate the value function of the optimal policy directly. Recall from Bellman's theorem (10.32) that the optimal policy $\pi^\star$ can be characterized in terms of the optimal state-action value function $q^\star$,

$$\pi^\star(x) = \arg\max_{a \in A} q^\star(x, a).$$

$\pi^\star$ corresponds to greedily maximizing the value function.

Analogously to our derivation of SARSA (11.18), only using Bellman's theorem (10.33) in place of Bellman's expectation equation (10.15), we obtain,

$$\begin{aligned} q^\star(x,a) &= r(x,a) + \gamma \sum_{x' \in X} p(x' \mid x, a) \max_{a' \in A} q^\star(x', a') && \text{using that the Q-function is a fixed-point of the Bellman update, see Bellman's theorem (10.33)} \\ &= \mathbb{E}_{R_0, X_1}\left[R_0 + \gamma \max_{a' \in A} q^\star(X_1, a') \mid X_0 = x, A_0 = a \right] && (11.23) \quad \text{interpreting the above expression as an expectation over } R_0 \text{ and } X_1 \\ &\approx r + \gamma \max_{a' \in A} q^\star(x', a') && (11.24) \quad \text{Monte Carlo approximation with a single sample} \end{aligned}$$

where the agent observed the transition (x, a, r, x'). Using a bootstrapping estimate $Q^\star$ for $q^\star$, we obtain a structurally similar algorithm to TD-learning and SARSA — only for estimating the optimal Q-function directly! This algorithm is known as *Q-learning*. Whereas we have seen that the optimal policy can be found using SARSA in a policy-iteration-like scheme, Q-learning is conceptually similar to value iteration.

Algorithm 11.12: Q-learning

initialize $Q^\star(x,a)$ arbitrarily (e.g., as $\mathbf{0}$)
for $t = 0$ **to** ∞ **do**
  observe the transition (x, a, r, x')
  $Q^\star(x,a) \leftarrow (1-\alpha_t)Q^\star(x,a) + \alpha_t(r + \gamma \max_{a' \in A} Q^\star(x', a'))$
  // (11.25)

Similarly to TD-learning, the update rule can also be expressed as

$$Q^\star(x,a) \leftarrow Q^\star(x,a) + \alpha_t \left(r + \gamma \max_{a' \in A} Q^\star(x', a') - Q^\star(x,a) \right). \quad (11.26)$$

Crucially, the Monte Carlo approximation of eq. (11.23) does not depend on the policy. Thus, Q-learning is an off-policy method.

Theorem 11.13 (Convergence of Q-learning, Jaakkola et al. (1993))**.** *If* $(\alpha_t)_{t \in \mathbb{N}_0}$ *satisfies the RM-conditions* (1.109) *and all state-action pairs are chosen infinitely often (that is, the sequence of policies used to obtain the transitions is GLIE), then* $Q^\star$ *converges to* $q^\star$ *with probability* 1.

It can be shown that with probability at least $1 - \delta$, Q-learning converges to an ϵ-optimal policy in a number of steps that is polynomial in $\log |X|$, $\log |A|$, $1/\epsilon$ and $\log 1/\delta$ (Even-Dar et al., 2003).

Exercise 11.14: Q-learning

Assume the following grid world, where from state A the agent can go to the right and down, and from state B to the left and down. From states G_1 and G_2 the only action is to exit. The agent receives a reward ($+10$ or $+1$) only when exiting.

Rewards

0	0
$+10$	$+1$

States

A	B
G_1	G_2

We assume the discount factor $\gamma = 1$ and that all actions are deterministic.

1. We observe the following two episodes:

Episode 1

x	a	x'	r
A	$\downarrow$	G_1	0
G_1	exit		10

Episode 2

x	a	x'	r
B	$\leftarrow$	A	0
A	$\downarrow$	G_1	0
G_1	exit		10

Assume $\alpha = 0.3$, and that Q-values of all non-terminal states

are initialized to 0.5. What are the Q-values

$$Q^\star(A, \downarrow), \quad Q^\star(G_1, \text{exit}), \quad Q^\star(G_2, \text{exit})$$

learned by executing Q-learning with the above episodes?

2. Will Q-learning converge to $q^\star$ for all state-action pairs (x, a) if we repeat episode 1 and episode 2 infinitely often? If not, design a sequence of episodes that leads to convergence.
3. How does the choice of initial Q-values influence the convergence of the Q-learning algorithm when episodes are obtained off-policy?
4. Determine $v^\star$ for all states.

▷ *Solution*

11.4.5 *Optimistic Q-learning*

The next natural question is how to effectively trade exploration and exploitation to both visit all state-action pairs many times, but also attain a high reward.

However, as we have seen in section 11.3, random "uninformed" exploration like ϵ-greedy and softmax exploration explores the state space very slowly. We therefore return to the principle of *optimism in the face of uncertainty*, which already led us to the $R_{\max}$ algorithm in the model-based setting. We will now additionally assume that the rewards are non-negative, that is, $0 \leq r(x, a) \leq R_{\max}$ $(\forall x \in X, a \in A)$. It turns out that a similar algorithm to $R_{\max}$ also exists for (model-free) Q-learning: it is called *optimistic Q-learning* and shown in alg. 11.15.

Algorithm 11.15: Optimistic Q-learning

initialize $Q^\star(x, a) = V_{\max} \prod_{t=1}^{T_{\text{init}}} (1 - \alpha_t)^{-1}$
for $t = 0$ **to** ∞ **do**
  pick action $a_t = \arg\max_{a \in A} Q^\star(x, a)$ and observe the transition (x, a, r, x')
  $Q^\star(x, a) \leftarrow (1 - \alpha_t) Q^\star(x, a) + \alpha_t (r + \gamma \max_{a' \in A} Q^\star(x', a'))$ // (11.27)

Here,

$$V_{\max} \doteq \frac{R_{\max}}{1 - \gamma} \geq \max_{x \in X, a \in A} q^\star(x, a)$$

is an upper bound on the discounted return and T_{init} is some initialization time. Intuitively, the initialization of $Q^\star$ corresponds to the

best-case long-term reward, assuming that all individual rewards are upper bounded by $R_{\max}$. This is shown by the following lemma.

Lemma 11.16. *Denote by $Q_t^\star$, the approximation of $q^\star$ attained in the t-th iteration of optimistic Q-learning. Then, for any state-action pair (x, a) and iteration t such that $N(a \mid x) \leq T_{\text{init}}$,*[7]

$$Q_t^\star(x, a) \geq V_{\max} \geq q^\star(x, a). \tag{11.28}$$

[7] $N(a \mid x)$ is the number of times action a is performed in state x.

Proof. We write $\beta_\tau \doteq \prod_{i=1}^{\tau}(1 - \alpha_i)$ and $\eta_{i,\tau} \doteq \alpha_i \prod_{j=i+1}^{\tau}(1 - \alpha_j)$. Using the update rule of optimistic Q-learning (11.27), we have

$$Q_t^\star(x, a) = \beta_{N(a|x)} Q_0^\star(x, a) + \sum_{i=1}^{N(a|x)} \eta_{i,N(a|x)} (r + \gamma \max_{a_i \in A} Q_{t_i}^\star(x_i, a_i)) \tag{11.29}$$

where x_i is the next state arrived at time t_i when action a is performed the i-th time in state x.

Using the assumption that the rewards are non-negative, from eq. (11.29) and $Q_0^\star(x, a) = {}^{V_{\max}}/_{\beta_{T_{\text{init}}}}$, we immediately have

$$\begin{aligned} Q_t^\star(x, a) &\geq \frac{\beta_{N(a|x)}}{\beta_{T_{\text{init}}}} V_{\max} \\ &\geq V_{\max}. \end{aligned}$$

□ using $N(a \mid x) \leq T_{\text{init}}$

Now, if T_{init} is chosen large enough, it can be shown that optimistic Q-learning converges quickly to an optimal policy.

Theorem 11.17 (Convergence of optimistic Q-learning, Even-Dar and Mansour (2001))**.** *With probability at least $1 - \delta$, optimistic Q-learning obtains an ϵ-optimal policy after a number of steps that is polynomial in $|X|$, $|A|$, ${}^1/_\epsilon$, $\log {}^1/_\delta$, and $R_{\max}$ where the initialization time T_{init} is upper bounded by a polynomial in the same coefficients.*

Note that for Q-learning, we still need to store $Q^\star(x, a)$ for any state-action pair in memory. Thus, Q-learning requires $\mathcal{O}(nm)$ memory. During each transition, we need to compute

$$\max_{a \in A} Q^\star(x', a')$$

once. If we run Q-learning for T iterations, this yields a time complexity of $\mathcal{O}(Tm)$. Crucially, for sparse Markov decision processes where, in most states, only few actions are permitted, each iteration of Q-learning can be performed in (virtually) constant time. This is a big improvement of the quadratic (in the number of states) performance of the model-based $R_{\max}$ algorithm.

11.4.6 Challenges of Model-free Approaches

We have seen that both the model-based $R_{\max}$ algorithm and the model-free Q-learning take time polynomial in the number of states $|X|$ and the number of actions $|A|$ to converge. While this is acceptable in small grid worlds, this is completely unacceptable for large state and action spaces.

Often, domains are continuous, for example when modeling beliefs about states in a partially observable environment. Also, in many structured domains (e.g., chess or multiagent planning), the size of the state and action space is exponential in the size of the input. In the final two chapters, we will therefore explore how model-free and model-based methods can be used (approximately) in such large domains.

Readings

For a more thorough treatment of tabular RL methods refer to

- chapter 22 of "Artificial intelligence: a modern approach" (Russell and Norvig, 2002) or
- chapter 6 of "Reinforcement learning: An introduction" (Sutton and Barto, 2018).

12

Model-free Reinforcement Learning

In the previous chapter, we have seen methods for tabular settings. Our goal now is to extend the model-free methods like TD-learning and Q-learning to large state-action spaces $\mathcal{X}$ and $\mathcal{A}$. We have seen that a crucial bottleneck of these methods is the parameterization of the value function. If we want to store the value function in a table, we need at least $\mathcal{O}(|\mathcal{X}|)$ space. If we learn the Q function, we even need $\mathcal{O}(|\mathcal{X}| \cdot |\mathcal{A}|)$ space. Also, for large state-action spaces, the time required to compute the value function for every state-action pair exactly will grow polynomially in the size of the state-action space. Hence, a natural idea is to learn *approximations* of these functions with a low-dimensional parameterization. Such approximations were the focus of the first few chapters and are, in fact, the key idea behind methods for reinforcement learning in large state-action spaces.

12.1 *Tabular Reinforcement Learning as Optimization*

To begin with, let us reinterpret the model-free methods from the previous section, TD-learning and Q-learning, as solving an optimization problem, where each iteration corresponds to a single gradient update. We will focus on TD-learning here, but the same interpretation applies to Q-learning. Recall the update rule of TD-learning (11.15),

$$V^\pi(x) \leftarrow (1-\alpha_t)V^\pi(x) + \alpha_t(r + \gamma V^\pi(x')).$$

Note that this looks just like the update rule of an optimization algorithm! We can parameterize our estimates V^π with parameters $\boldsymbol{\theta}$ that are updated according to the gradient of some loss function, assuming fixed bootstrapping estimates. In particular, in the tabular setting (i.e., over a finite domain), we can parameterize the value function exactly by learning a separate parameter for each state,

$$\boldsymbol{\theta} \doteq [\boldsymbol{\theta}(1), \dots, \boldsymbol{\theta}(n)], \quad V^\pi(x;\boldsymbol{\theta}) \doteq \boldsymbol{\theta}(x). \tag{12.1}$$

To re-derive the above update rule as a gradient update, let us consider the following loss function,

$$\bar{\ell}(\boldsymbol{\theta}; x, r) \doteq \frac{1}{2}(v^\pi(x) - \boldsymbol{\theta}(x))^2 \tag{12.2}$$

$$= \frac{1}{2}\Big(r + \gamma \mathbb{E}_{x'|x,\pi(x)}\big[v^\pi(x')\big] - \boldsymbol{\theta}(x)\Big)^2 \tag{12.3}$$

using Bellman's expectation equation (10.11)

Note that this loss corresponds to a standard squared loss of the difference between the parameter $\boldsymbol{\theta}(x)$ and the label $v^\pi(x)$ we want to learn.

We can now find the gradient of this loss. Elementary calculations yield,

$$\nabla_{\boldsymbol{\theta}(x)} \bar{\ell}(\boldsymbol{\theta}; x, r) = \boldsymbol{\theta}(x) - \Big(r + \gamma \mathbb{E}_{x'|x,\pi(x)}\big[v^\pi(x')\big]\Big). \tag{12.4}$$

Now, we cannot compute this derivative because we cannot compute the expectation. Firstly, the expectation is over the true value function which is unknown to us. Secondly, the expectation is over the transition model which we are trying to avoid in model-free methods.

To resolve the first issue, analogously to TD-learning, instead of learning the true value function v^π which is unknown, we learn the bootstrapping estimate V^π. Recall that the core principle behind bootstrapping as discussed in section 11.4.1 is that this bootstrapping estimate V^π is *treated* as if it were independent of the current estimate of the value function $\boldsymbol{\theta}$. To emphasize this, we write $V^\pi(x; \boldsymbol{\theta}^{\text{old}}) \approx v^\pi(x)$ where $\boldsymbol{\theta}^{\text{old}} = \boldsymbol{\theta}$ but $\boldsymbol{\theta}^{\text{old}}$ is treated as a constant with respect to $\boldsymbol{\theta}$.[1]

[1] That is, the bootstrapping estimate $V^\pi(x; \boldsymbol{\theta}^{\text{old}})$ is assumed to be constant with respect to $\boldsymbol{\theta}(x)$ in the same way that $v^\pi(x)$ is constant with respect to $\boldsymbol{\theta}(x)$.

If we were not using the bootstrapping estimate, the following derivation of the gradient of the loss would not be this simple.

To resolve the second issue, analogously to the introduction of TD-learning in the previous chapter, we will use a Monte Carlo estimate using a single sample. Recall that this is only possible because the transitions are conditionally independent given the state-action pair.

Remark 12.1: Sample (in)efficiency of model-free methods

Taking these two shortcuts are two of the main reasons why model-free methods such as TD-learning and Q-learning are usually *sample inefficient*. This is because using a bootstrapping estimate leads to "(initially) incorrect" and "unstable" targets of the optimization problem,[2] and Monte Carlo estimation with a single sample leads to a large variance. Recall that the theoretical guarantees for model-free methods in the tabular setting therefore required that all state-action pairs are visited infinitely often.

[2] We explore this in some more capacity in section 12.2.1 where we cover heuristic approaches to alleviate this problem to some degree.

Using the aforementioned shortcuts, let us define the loss ℓ after ob-

serving the single transition (x, a, r, x'),

$$\ell(\boldsymbol{\theta}; x, r, x') \doteq \frac{1}{2}\Big(r + \gamma \boldsymbol{\theta}^{\text{old}}(x') - \boldsymbol{\theta}(x)\Big)^2. \tag{12.5}$$

We define the gradient of this loss with respect to $\boldsymbol{\theta}(x)$ as

$$\begin{aligned}\delta_{\text{TD}} &\doteq \boldsymbol{\nabla}_{\boldsymbol{\theta}(x)}\, \ell(\boldsymbol{\theta}; x, r, x') \\ &= \boldsymbol{\theta}(x) - \Big(r + \gamma \boldsymbol{\theta}^{\text{old}}(x')\Big).\end{aligned} \tag{12.6}$$

This error term is also called *temporal-difference (TD) error*. The temporal difference error compares the previous estimate of the value function to the bootstrapping estimate of the value function. We know from the law of large numbers (1.84) that Monte Carlo averages are unbiased.[3] We therefore have,

$$\mathbb{E}_{x'|x,\pi(x)}[\delta_{\text{TD}}] = \boldsymbol{\nabla}_{\boldsymbol{\theta}(x)}\, \overline{\ell}(\boldsymbol{\theta}; x, r). \tag{12.7}$$

[3] Crucially, the samples are unbiased with respect to the approximate label in terms of the bootstrapping estimate only. Due to bootstrapping the value function, the estimates are not unbiased with respect to the true value function. Moreover, the variance of each individual estimation of the gradient is large, as we only consider a single transition.

Naturally, we can use these unbiased gradient estimates with respect to the loss $\overline{\ell}$ to perform stochastic gradient descent. This yields the update rule,

$$V^\pi(x; \boldsymbol{\theta}) = \boldsymbol{\theta}(x) \leftarrow \boldsymbol{\theta}(x) - \alpha_t \delta_{\text{TD}} \tag{12.8}$$

using stochastic gradient descent with learning rate α_t, see alg. 1.60

$$= (1 - \alpha_t)\boldsymbol{\theta}(x) + \alpha_t\Big(r + \gamma \boldsymbol{\theta}^{\text{old}}(x')\Big)$$

using the definition of the temporal difference error (12.6)

$$= (1 - \alpha_t) V^\pi(x; \boldsymbol{\theta}) + \alpha_t\Big(r + \gamma V^\pi(x'; \boldsymbol{\theta}^{\text{old}})\Big). \tag{12.9}$$

substituting $V^\pi(x; \boldsymbol{\theta})$ for $\boldsymbol{\theta}(x)$

Observe that this gradient update coincides with the update rule of TD-learning (11.15). Therefore, TD-learning is essentially performing stochastic gradient descent using the TD-error as an unbiased gradient estimate.[4] Crucially, TD-learning performs stochastic gradient descent with respect to the bootstrapping estimate of the value function V^π and not the true value function v^π! Stochastic gradient descent with a bootstrapping estimate is also called *stochastic semi-gradient descent*. Importantly, the optimization target $r + \gamma V^\pi(x'; \boldsymbol{\theta}^{\text{old}})$ from the loss ℓ is now *moving* between iterations which introduces some practical challenges we will discuss in section 12.2.1. We have seen in the previous chapter that using a bootstrapping estimate still guarantees (asymptotic) convergence to the true value function.

[4] An alternative interpretation is that TD-learning performs gradient descent with respect to the loss ℓ.

12.2 Value Function Approximation

To scale to large state spaces, it is natural to approximate the value function using a parameterized model, $V(\boldsymbol{x}; \boldsymbol{\theta})$ or $Q(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta})$. You may think of this as a regression problem where we map state(-action) pairs to a real number. Recall from the previous section that this is a strict

generalization of the tabular setting, as we could use a separate parameter to learn the value function for each individual state-action pair. Our goal for large state-action spaces is to exploit the smoothness properties[5] of the value function to condense the representation.

[5] That is, the value function takes similar values in "similar" states.

A straightforward approach is to use a linear function approximation with the hand-designed feature map $\boldsymbol{\phi}$,

$$Q(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}) \doteq \boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{a}). \tag{12.10}$$

A common alternative is to use a deep neural network to learn these features instead. Doing so is also known as *deep reinforcement learning*.[6]

[6] Note that often non-Bayesian neural networks (i.e., point estimates of the weights) are used. In the final chapter, chapter 13, we will explore the benefits of using Bayesian neural networks.

We will now apply the derivation from the previous section directly to Q-learning. For Q-learning, after observing the transition $(\boldsymbol{x}, \boldsymbol{a}, r, \boldsymbol{x}')$, the loss function is given as

$$\ell(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{a}, r, \boldsymbol{x}') \doteq \frac{1}{2}\left(r + \gamma \max_{\boldsymbol{a}' \in \mathcal{A}} Q^\star(\boldsymbol{x}', \boldsymbol{a}'; \boldsymbol{\theta}^{\text{old}}) - Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta})\right)^2. \tag{12.11}$$

Here, we simply use Bellman's optimality equation (10.33) to estimate $q^\star(\boldsymbol{x}, \boldsymbol{a})$, instead of the estimation of $v^\pi(\boldsymbol{x})$ using Bellman's expectation equation for TD-learning. The difference between the current approximation and the optimization target,

$$\delta_{\text{B}} \doteq r + \gamma \max_{\boldsymbol{a}' \in \mathcal{A}} Q^\star(\boldsymbol{x}', \boldsymbol{a}'; \boldsymbol{\theta}^{\text{old}}) - Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}), \tag{12.12}$$

is called the *Bellman error*. Analogously to TD-learning,[7] we obtain the gradient update,

[7] compare to eq. (12.8)

$$\begin{aligned}
\boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} - \alpha_t \boldsymbol{\nabla}_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{a}, r, \boldsymbol{x}') && (12.13)\\
&= \boldsymbol{\theta} - \alpha_t \boldsymbol{\nabla}_{\boldsymbol{\theta}} \frac{1}{2}\left(r + \gamma \max_{\boldsymbol{a}' \in \mathcal{A}} Q^\star(\boldsymbol{x}', \boldsymbol{a}'; \boldsymbol{\theta}^{\text{old}}) - Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta})\right)^2 && \text{using the definition of } \ell \text{ (12.11)}\\
&= \boldsymbol{\theta} + \alpha_t \delta_{\text{B}} \boldsymbol{\nabla}_{\boldsymbol{\theta}} Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}). && (12.14) \quad \text{using the chain rule}
\end{aligned}$$

When using a neural network to learn $Q^\star$, we can use automatic differentiation to obtain unbiased gradient estimates. In the case of linear function approximation, we can compute the gradient exactly,

$$\begin{aligned}
&= \boldsymbol{\theta} + \alpha_t \delta_{\text{B}} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{a}) && \text{using the linear approximation of } Q^\star \text{ (12.10)}\\
&= \boldsymbol{\theta} + \alpha_t \delta_{\text{B}} \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{a}). && (12.15)
\end{aligned}$$

In the tabular setting, this algorithm is identical to Q-learning and, in particular, converges to the true Q-function $q^\star$.[8] There are few such results in the approximate setting. Usage in practice indicates that using an approximation of the value function "should be fine" when a "rich-enough" class of functions is used.

[8] see theorem 11.13

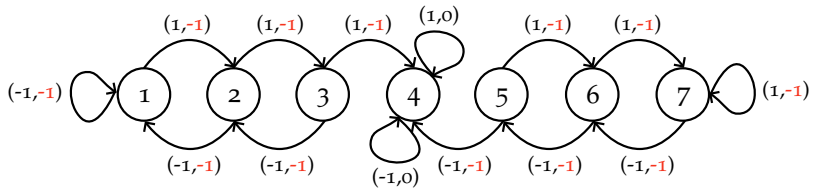


Figure 12.1: MDP studied in exercise 12.2. Each arrow marks a (deterministic) transition and is labeled with (action, reward).

Exercise 12.2: Q-learning and function approximation

Consider the MDP of fig. 12.1 and set $\gamma = 1$.

1. Using Bellman's theorem, prove that $v^\star(x) = -|x-4|$ is the optimal value function.
2. Suppose we observe the following episode:

x	a	x'	r
3	-1	2	-1
2	1	3	-1
3	1	4	-1
4	1	4	0

   We initialize all Q-values to 0. Compute the updated Q-values using Q-learning with learning rate $\alpha = 1/2$.
3. We will now approximate the Q-function with a linear function. We let

$$Q(x, a; \boldsymbol{w}) \doteq x w_0 + a w_1 + w_2$$

   where $\boldsymbol{w} = [w_0\ w_1\ w_2]^\top \in \mathbb{R}^3$.
   Suppose we have $\boldsymbol{w}^{\text{old}} = [1\ -1\ -2]^\top$ and $\boldsymbol{w} = [-1\ 1\ 1]^\top$, and we observe the transition $\tau = (2, -1, -1, 1)$. Use the learning rate $\alpha = 1/4$ to compute $\boldsymbol{\nabla}_{\boldsymbol{w}}\, \ell(\boldsymbol{w}; \tau)$ and the updated weights $\boldsymbol{w}' = \boldsymbol{w} - \alpha \boldsymbol{\nabla}_{\boldsymbol{w}}\, \ell(\boldsymbol{w}; \tau)$.

▷ *Solution*

12.2.1 *Heuristics*

The vanilla stochastic semi-gradient descent is very slow. In this subsection, we will discuss some "tricks of the trait" to improve its performance.

Stabilizing Optimization Targets There are mainly two problems. The first problem is that, as mentioned previously, the bootstrapping estimate changes after each iteration. As we are trying to learn an approximate value function that depends on the bootstrapping estimate, this means that the optimization target is "moving" between iterations. In practice, moving targets lead to stability issues. The first family of

techniques we discuss here aim to "stabilize" the optimization targets.

One such technique is called *neural fitted Q-iteration* or *deep Q-networks* (DQN) (Mnih et al., 2015). DQN updates the neural network used for the approximate bootstrapping estimate infrequently to maintain a constant optimization target across multiple episodes. How this is implemented exactly varies. One approach is to clone the neural network and maintain one changing neural network ("online network") for the most recent estimate of the Q-function which is parameterized by $\boldsymbol{\theta}$, and one fixed neural network ("target network") used as the target which is parameterized by $\boldsymbol{\theta}^{\text{old}}$ and which is updated infrequently.

This can be implemented by maintaining a data set $\mathcal{D}$ of observed transitions (the so-called *replay buffer*) and then "every once in a while" (e.g., once $|\mathcal{D}|$ is large enough) solving a regression problem, where the labels are determined by the target network. This yields a loss term where the target is fixed across all transitions in the replay buffer $\mathcal{D}$,

$$\ell_{\text{DQN}}(\boldsymbol{\theta}; \mathcal{D}) \doteq \frac{1}{2} \sum_{(\boldsymbol{x},\boldsymbol{a},r,\boldsymbol{x}') \in \mathcal{D}} \left(r + \gamma \max_{\boldsymbol{a}' \in \mathcal{A}} Q^\star(\boldsymbol{x}', \boldsymbol{a}'; \boldsymbol{\theta}^{\text{old}}) - Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}) \right)^2. \tag{12.16}$$

compare to the Q-learning loss (12.11)

The loss can also be interpreted (in an online sense) as performing regular Q-learning with the modification that the target network $\boldsymbol{\theta}^{\text{old}}$ is not updated to $\boldsymbol{\theta}$ after every observed transition, but instead only after observing $|\mathcal{D}|$-many transitions. This technique is known as *experience replay*. Another approach is *Polyak averaging* where the target network is gradually "nudged" by the neural network used to estimate the Q-function.

Maximization Bias Now, observe that the estimates $Q^\star$ are noisy estimates of $q^\star$ and consider the term,

$$\max_{\boldsymbol{a}' \in \mathcal{A}} q^\star(\boldsymbol{x}', \boldsymbol{a}') \approx \max_{\boldsymbol{a}' \in \mathcal{A}} Q^\star(\boldsymbol{x}', \boldsymbol{a}'; \boldsymbol{\theta}^{\text{old}}),$$

from the loss function (12.16). This term maximizes a *noisy* estimate of $q^\star$, which leads to a *biased* estimate of $\max q^\star$ as can be seen in fig. 12.2. The fact that the update rules of Q-learning and DQN are affected by inaccuracies (i.e., noise in the estimates) of the learned Q-function is known as the "*maximization bias*".

Double DQN (DDQN) is an algorithm that addresses this maximization bias (Van Hasselt et al., 2016). Instead of picking the optimal action with respect to the old network, it picks the optimal action with respect

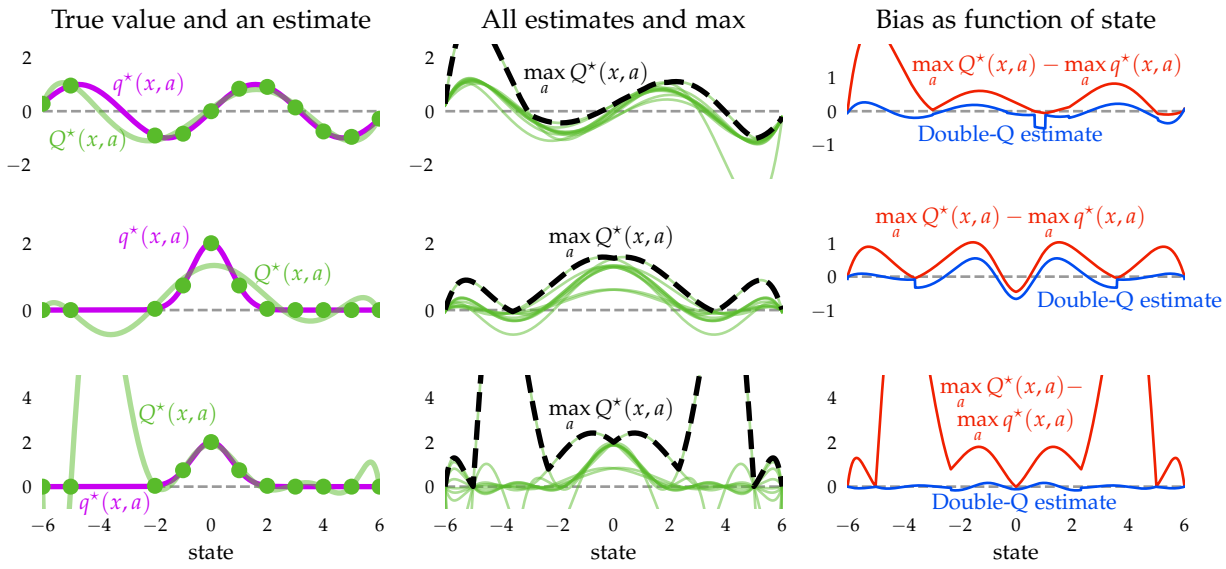


Figure 12.2: Illustration of overestimation during learning. In each state (x-axis), there are 10 actions. The left column shows the true values $v^\star(x)$ (purple line). All true action values are defined by $q^\star(x,a) \doteq v^\star(x)$. The green line shows estimated values $Q^\star(x,a)$ for one action as a function of state, fitted to the true value at several sampled states (green dots). The middle column plots show all the estimated values (green), and the maximum of these values (dashed black). The maximum is higher than the true value (purple, left plot) almost everywhere. The right column plots show the difference in red. The blue line in the right plots is the estimate used by Double Q-learning with a second set of samples for each state. The blue line is much closer to zero, indicating less bias. The three rows correspond to different true functions (left, purple) or capacities of the fitted function (left, green). Taken from "Deep reinforcement learning with double q-learning" (Van Hasselt et al., 2016).

to the new network,

$$\ell_{\text{DDQN}}(\boldsymbol{\theta};\mathcal{D}) \doteq \frac{1}{2}\sum_{(\boldsymbol{x},\boldsymbol{a},r,\boldsymbol{x}')\in\mathcal{D}}\Big(r+\gamma Q^\star(\boldsymbol{x}',\boldsymbol{a}^\star(\boldsymbol{x}';\boldsymbol{\theta});\boldsymbol{\theta}^{\text{old}}) - Q^\star(\boldsymbol{x},\boldsymbol{a};\boldsymbol{\theta})\Big)^2 \tag{12.17}$$

$$\text{where}\quad \boldsymbol{a}^\star(\boldsymbol{x}';\boldsymbol{\theta}) \doteq \arg\max_{\boldsymbol{a}'\in\mathcal{A}} Q^\star(\boldsymbol{x}',\boldsymbol{a}';\boldsymbol{\theta}). \tag{12.18}$$

Intuitively, this change ensures that the evaluation of the target network is consistent with the updated Q-function, which makes the algorithm more robust to noise.

Similarly to DQN, this can also be interpreted as the online update,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_t\Big(r+\gamma Q^\star(\boldsymbol{x}',\boldsymbol{a}^\star(\boldsymbol{x}';\boldsymbol{\theta});\boldsymbol{\theta}^{\text{old}}) - Q^\star(\boldsymbol{x},\boldsymbol{a};\boldsymbol{\theta})\Big)\nabla_{\boldsymbol{\theta}} Q^\star(\boldsymbol{x},\boldsymbol{a};\boldsymbol{\theta}) \tag{12.19}$$

after observing a single transition $(\boldsymbol{x},\boldsymbol{a},r,\boldsymbol{x}')$ where while differentiating, $\boldsymbol{a}^\star(\boldsymbol{x}';\boldsymbol{\theta})$ is treated as constant with respect to $\boldsymbol{\theta}$. $\boldsymbol{\theta}^{\text{old}}$ is then updated to $\boldsymbol{\theta}$ after observing $|\mathcal{D}|$-many transitions.

Readings

For more details on value function approximation, refer to chapter 9 of "Reinforcement learning: An introduction" (Sutton and Barto, 2018).

Optional Readings

- Mnih, Kavukcuoglu, Silver, Rusu, Veness, Bellemare, Graves, Riedmiller, Fidjeland, Ostrovski, et al. (2015).
  Human-level control through deep reinforcement learning.
- Van Hasselt, Guez, and Silver (2016).
  Deep reinforcement learning with double q-learning.

12.3 Policy Approximation

Q-learning defines a policy implicitly by

$$\pi^\star(\boldsymbol{x}) \doteq \arg\max_{\boldsymbol{a} \in \mathcal{A}} Q^\star(\boldsymbol{x}, \boldsymbol{a}). \tag{12.20}$$

Q-learning also maximizes over the set of all actions in its update step while learning the Q-function. This is intractable for large and, in particular, continuous action spaces. A natural idea to escape this limitation is to immediately learn an approximate parameterized policy,

$$\pi^\star(\boldsymbol{x}) \approx \boldsymbol{\pi}(\boldsymbol{x}; \boldsymbol{\varphi}) \doteq \boldsymbol{\pi}_{\boldsymbol{\varphi}}(\boldsymbol{x}). \tag{12.21}$$

Methods that find an approximate policy are also called *policy search methods* or *policy gradient methods*.

Whereas with Q-learning, exploration can be encouraged by using an ϵ-greedy policy, softmax exploration, or an optimistic initialization, we will see that policy gradient methods fundamentally rely on randomized policies for exploration.

Remark 12.3: Notation

We refer to deterministic policies $\boldsymbol{\pi}$ in bold, as they can be interpreted as vector-valued functions from $\mathcal{X}$ to $\mathcal{A}$. We still refer to randomized policies by π, as for each state $\boldsymbol{x} \in \mathcal{X}$ they are represented as a PDF over actions $\mathcal{A}$.

In particular, we denote by $\pi_{\boldsymbol{\varphi}}(\boldsymbol{a} \mid \boldsymbol{x})$ the probability of playing action $\boldsymbol{a}$ when in state $\boldsymbol{x}$ according to $\pi_{\boldsymbol{\varphi}}$.

12.3.1 Estimating Policy Values

We will begin by attributing a "value" to a policy. Recall the definition of the discounted payoff G_t from time t, which we are aiming to maximize,

$$G_t = \sum_{m=0}^{\infty} \gamma^m R_{t+m}. \qquad \text{see eq. (10.5)}$$

We define $G_{t:T}$ to be the *bounded discounted payoff* until time T,

$$G_{t:T} \doteq \sum_{m=0}^{T-1-t} \gamma^m R_{t+m}. \tag{12.22}$$

Based on these two random variables, we define the policy value function.

Definition 12.4 (Policy value function)**.** The *policy value function*,

$$j(\pi) \doteq \mathbb{E}_\pi[G_0] = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t\right], \tag{12.23}$$

measures the expected discounted payoff of policy π. We also define the bounded variant,

$$j_T(\pi) \doteq \mathbb{E}_\pi[G_{0:T}] = \mathbb{E}_\pi\left[\sum_{t=0}^{T-1} \gamma^t R_t\right]. \tag{12.24}$$

For simplicity, we will abbreviate $j(\boldsymbol{\varphi}) \doteq j(\pi_{\boldsymbol{\varphi}})$.

Naturally, we want to maximize $j(\boldsymbol{\varphi})$. That is, we want to solve

$$\boldsymbol{\varphi}^\star \doteq \arg\max_{\boldsymbol{\varphi}} j(\boldsymbol{\varphi}) \tag{12.25}$$

which is a non-convex optimization problem. Let us see how $j(\boldsymbol{\varphi})$ can be evaluated to understand the optimization problem better. We will again use a Monte Carlo estimate. Recall that a fixed $\boldsymbol{\varphi}$ induces a unique Markov chain, which can be simulated. In the episodic setting, each episode (also called *rollout*) of length T yields an independently sampled trajectory,

$$\tau^{(i)} \doteq ((\boldsymbol{x}_0^{(i)}, \boldsymbol{a}_0^{(i)}, r_0^{(i)}, \boldsymbol{x}_1^{(i)}), (\boldsymbol{x}_1^{(i)}, \boldsymbol{a}_1^{(i)}, r_1^{(i)}, \boldsymbol{x}_2^{(i)}), \dots) \tag{12.26}$$

Simulating m rollouts yields the samples,

$$\tau^{(1)}, \dots, \tau^{(m)} \sim \Pi_{\boldsymbol{\varphi}}, \tag{12.27}$$

where $\Pi_{\boldsymbol{\varphi}}$ is the distribution of all trajectories (i.e., rollouts) of the Markov chain induced by policy $\pi_{\boldsymbol{\varphi}}$. We denote the (bounded) discounted payoff of the i-th rollout by

$$g_{0:T}^{(i)} \doteq \sum_{t=0}^{T-1} \gamma^t r_t^{(i)} \tag{12.28}$$

where $r_t^{(i)}$ is the reward at time t of the i-th rollout. Using a Monte Carlo approximation, we can then estimate $j_T(\boldsymbol{\varphi})$. Moreover, due to the exponential discounting of future rewards, it is reasonable to approximate the policy value function using bounded trajectories,

$$j(\boldsymbol{\varphi}) \approx j_T(\boldsymbol{\varphi}) \approx \frac{1}{m} \sum_{i=1}^{m} g_{0:T}^{(i)}. \tag{12.29}$$

12.3.2 Policy Gradient

Let us now formally define the distribution over trajectories $\Pi_{\boldsymbol{\varphi}}$ that we introduced in the previous section. We can specify the probability of a specific trajectory τ under a policy $\pi_{\boldsymbol{\varphi}}$ by

$$\Pi_{\boldsymbol{\varphi}}(\tau) = p(\boldsymbol{x}_0) \prod_{t=0}^{T-1} \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t) p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{a}_t). \tag{12.30}$$

For optimizing $j(\boldsymbol{\varphi})$ we need to obtain unbiased estimates of its gradient,

$$\boldsymbol{\nabla}_{\boldsymbol{\varphi}} j(\boldsymbol{\varphi}) \approx \boldsymbol{\nabla}_{\boldsymbol{\varphi}} j_T(\boldsymbol{\varphi}) = \boldsymbol{\nabla}_{\boldsymbol{\varphi}} \mathbb{E}_{\tau \sim \Pi_{\boldsymbol{\varphi}}}[G_0]. \tag{12.31}$$

Note that the expectation integrates over the measure $\Pi_{\boldsymbol{\varphi}}$, which depends on the parameter $\boldsymbol{\varphi}$. Thus, we cannot move the gradient operator inside the expectation as we have done previously in remark 1.22. This should remind you of the reparameterization trick (see eq. (5.77)) that we used to solve a similar gradient in the context of variational inference. In this context, however, we cannot apply the reparameterization trick.[9] Fortunately, there is another way of estimating this gradient.

[9] This is because the distribution $\Pi_{\boldsymbol{\varphi}}$ is generally not reparameterizable. We will, however, see that reparameterization gradients are also useful in reinforcement learning. See, e.g., sections 12.4.5 and 13.1.2.

Theorem 12.5 (Score gradient estimator)**.** *Under some regularity assumptions, we have*

$$\boldsymbol{\nabla}_{\boldsymbol{\varphi}} \mathbb{E}_{\tau \sim \Pi_{\boldsymbol{\varphi}}}[G_0] = \mathbb{E}_{\tau \sim \Pi_{\boldsymbol{\varphi}}}\left[G_0 \boldsymbol{\nabla}_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau)\right]. \tag{12.32}$$

This estimator of the gradient is called the score gradient estimator*.*

Proof. To begin with, let us look at the so-called *score function* of the distribution $\Pi_{\boldsymbol{\varphi}}$,

$$\boldsymbol{\nabla}_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau). \tag{12.33}$$

Using the chain rule, the score function can be expressed as

$$\boldsymbol{\nabla}_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau) = \frac{\boldsymbol{\nabla}_{\boldsymbol{\varphi}} \Pi_{\boldsymbol{\varphi}}(\tau)}{\Pi_{\boldsymbol{\varphi}}(\tau)} \tag{12.34}$$

and by rearranging the terms, we obtain

$$\boldsymbol{\nabla}_{\boldsymbol{\varphi}} \Pi_{\boldsymbol{\varphi}}(\tau) = \Pi_{\boldsymbol{\varphi}}(\tau) \boldsymbol{\nabla}_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau). \tag{12.35}$$

This is called the *score function trick*.[10]

[10] We have already applied this "trick" in exercise 6.31.

Now, assuming that state and action spaces are continuous, we obtain

$$\boldsymbol{\nabla}_{\boldsymbol{\varphi}} \mathbb{E}_{\tau \sim \Pi_{\boldsymbol{\varphi}}}[G_0] = \boldsymbol{\nabla}_{\boldsymbol{\varphi}} \int \Pi_{\boldsymbol{\varphi}}(\tau) \cdot G_0 \, d\tau$$

using the definition of expectation (1.23b)

$$= \int \nabla_{\boldsymbol{\varphi}} \Pi_{\boldsymbol{\varphi}}(\tau) \cdot G_0 \, d\tau$$ using the regularity assumptions to swap gradient and integral

$$= \int G_0 \cdot \Pi_{\boldsymbol{\varphi}}(\tau) \nabla_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau) \, d\tau$$ using the score function trick (12.35)

$$= \mathbb{E}_{\tau \sim \Pi_{\boldsymbol{\varphi}}} \big[G_0 \nabla_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau) \big]. \qquad \square$$ interpreting the integral as an expectation over $\Pi_{\boldsymbol{\varphi}}$

Intuitively, maximizing $j(\boldsymbol{\varphi})$ increases the probability of policies with high returns and decreases the probability of policies with low returns.

To use the score gradient estimator for estimating the gradient, we need to compute $\nabla_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau)$.

$$\begin{aligned}
&\nabla_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau) \\
&= \nabla_{\boldsymbol{\varphi}} \left(\log p(\boldsymbol{x}_0) + \sum_{t=0}^{T-1} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t) + \sum_{t=0}^{T-1} \log p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{a}_t) \right) && \text{using the definition of the distribution over trajectories } \Pi_{\boldsymbol{\varphi}} \\
&= \nabla_{\boldsymbol{\varphi}} \log p(\boldsymbol{x}_0) + \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t) + \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\varphi}} \log p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{a}_t) \\
&= \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t). && (12.36) \quad \text{using that the first and third term are independent of } \boldsymbol{\varphi}
\end{aligned}$$

When using a neural network for the parameterization of the policy π, we can use automatic differentiation to compute the gradients.

Exercise 12.6: Eligibility vector

The vector $\nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t)$ is commonly called *eligibility vector*. In the following, we assume that the action space $\mathcal{A}$ is finite and denote it by A.

If we parameterize $\pi_{\boldsymbol{\varphi}}$ as a softmax distribution

$$\pi_{\boldsymbol{\varphi}}(a \mid \boldsymbol{x}) \doteq \frac{\exp(h(\boldsymbol{x}, a, \boldsymbol{\varphi}))}{\sum_{b \in A} \exp(h(\boldsymbol{x}, b, \boldsymbol{\varphi}))} \tag{12.37}$$

with linear preferences $h(\boldsymbol{x}, a, \boldsymbol{\varphi}) \doteq \boldsymbol{\varphi}^\top \boldsymbol{\phi}(\boldsymbol{x}, a)$ where $\boldsymbol{\phi}(\boldsymbol{x}, a)$ is some feature vector, what is the form of the eligibility vector?

▷ *Solution*

The expectation of the score gradient estimator (12.32) can be approximated using Monte Carlo sampling,

$$\nabla_{\boldsymbol{\varphi}} j_T(\boldsymbol{\varphi}) \approx \frac{1}{m} \sum_{i=1}^{m} g_{0:T}^{(i)} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t^{(i)} \mid \boldsymbol{x}_t^{(i)}). \tag{12.38}$$

However, typically the variance of these estimates is very large. Using so-called *baselines* can reduce the variance dramatically.

Lemma 12.7 (Score gradients with baselines)**.** *We have,*

$$\mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\big[G_0 \nabla_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau)\big] = \mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\big[(G_0 - b)\nabla_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau)\big]. \quad (12.39)$$

Here, $b \in \mathbb{R}$ is called a baseline*.*

Proof. For the term to the right, we have due to linearity of expectation (1.24),

$$\begin{aligned}\mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\big[(G_0 - b)\nabla_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau)\big] = \mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\big[G_0 \nabla_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau)\big] \\ - \mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\big[b \cdot \nabla_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau)\big].\end{aligned}$$

Thus, it remains to show that the second term is zero,

$$\begin{aligned}\mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\big[b \cdot \nabla_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau)\big] &= b \cdot \int \Pi_{\boldsymbol{\varphi}}(\tau) \nabla_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau)\, d\tau && \text{using the definition of expectation (1.23b)} \\ &= b \cdot \int \Pi_{\boldsymbol{\varphi}}(\tau) \frac{\nabla_{\boldsymbol{\varphi}} \Pi_{\boldsymbol{\varphi}}(\tau)}{\Pi_{\boldsymbol{\varphi}}(\tau)}\, d\tau && \text{substituting the score function (12.33), ``undoing the score function trick''} \\ &= b \cdot \int \nabla_{\boldsymbol{\varphi}} \Pi_{\boldsymbol{\varphi}}(\tau)\, d\tau && \Pi_{\boldsymbol{\varphi}}(\tau) \text{ cancels} \\ &= b \cdot \nabla_{\boldsymbol{\varphi}} \int \Pi_{\boldsymbol{\varphi}}(\tau)\, d\tau \\ &= b \cdot \nabla_{\boldsymbol{\varphi}} 1 = 0. && \text{integrating a PDF over its domain is 1 and the derivative of a constant is 0}\end{aligned}$$

□

Exercise 12.8: Variance of score gradients with baselines

In this exercise, we will see a sufficient condition for baselines to reduce the variance of score gradient estimators.

1. Suppose for a random vector $\mathbf{X}$, we want to estimate $\mathbb{E}[f(\mathbf{X})]$ for some function f. Assume that you are given a function g and also its expectation $\mathbb{E}[g(\mathbf{X})]$. Instead of estimating $\mathbb{E}[f(\mathbf{X})]$ directly, we will instead estimate $\mathbb{E}[f(\mathbf{X}) - g(\mathbf{X})]$ as we know from linearity of expectation (1.24) that

   $$\mathbb{E}[f(\mathbf{X})] = \mathbb{E}[f(\mathbf{X}) - g(\mathbf{X})] + \mathbb{E}[g(\mathbf{X})].$$

   Prove that if $\frac{1}{2}\mathrm{Var}[g(\mathbf{X})] \leq \mathrm{Cov}[f(\mathbf{X}), g(\mathbf{X})]$, then

   $$\mathrm{Var}[f(\mathbf{X}) - g(\mathbf{X})] \leq \mathrm{Var}[f(\mathbf{X})]. \quad (12.40)$$

2. Consider estimating $\nabla_{\boldsymbol{\varphi}} j(\boldsymbol{\varphi})$. Prove that if $b^2 \leq 2b \cdot r(\boldsymbol{x}, \boldsymbol{a})$ for every state $\boldsymbol{x} \in \mathcal{X}$ and action $\boldsymbol{a} \in \mathcal{A}$, then

   $$\mathrm{Var}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\big[(G_0 - b)\nabla_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau)\big] \leq \mathrm{Var}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\big[G_0 \nabla_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau)\big]. \quad (12.41)$$

▷ *Solution*

Exercise 12.9: Score gradients with state-dependent baselines

One can even show, that we can subtract arbitrary baselines depending on *previous* states.

For a sequence of *state-dependent baselines* $\{b(\tau_{0:t-1})\}_{t=1}^{T}$ where

$$\tau_{0:t-1} \doteq (\tau_0, \tau_1, \ldots, \tau_{t-1}),$$

show that

$$\begin{aligned}&\mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\left[G_0 \nabla_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau)\right] \\ &= \mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\left[\sum_{t=0}^{T-1} (G_0 - b(\tau_{0:t-1})) \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t)\right]\end{aligned} \tag{12.42}$$

where we write $b(\tau_{0:-1}) = 0$.

▷ *Solution*

Example 12.10: Downstream returns

A commonly used baseline is

$$b(\tau_{0:t-1}) \doteq \sum_{m=0}^{t-1} \gamma^m r_m. \tag{12.43}$$

This baseline subtracts the returns of all actions before time t. Intuitively, using this baseline, the score gradient only considers downstream returns. Recall from eq. (12.22) that we defined $G_{t:T}$ as the bounded discounted payoff from time t. It is also commonly called the (bounded) *downstream return* (or *reward to go*) beginning at time t.

For a fixed trajectory τ that is bounded at time T, we have

$$G_0 - b(\tau_{0:t-1}) = \gamma^t G_{t:T}, \tag{12.44}$$

yielding the gradient estimator,

$$\begin{aligned}\nabla_{\boldsymbol{\varphi}} j(\boldsymbol{\varphi}) \approx \nabla_{\boldsymbol{\varphi}} j_T(\boldsymbol{\varphi}) &= \mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\left[G_0 \nabla_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau)\right] && \text{using the score gradient estimator (12.32)} \\ &= \mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\left[\sum_{t=0}^{T-1} \gamma^t G_{t:T} \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t)\right]. && \text{using a state-dependent baseline (12.42)}\end{aligned} \tag{12.45}$$

Performing stochastic gradient descent with the score gradient estimator and downstream returns is known as the *REINFORCE algorithm* (Williams, 1992) which is shown in alg. 12.11.

Algorithm 12.11: REINFORCE algorithm

initialize policy weights $\boldsymbol{\varphi}$
repeat
  generate an episode (i.e., rollout) to obtain trajectory τ
  for $t = 0$ **to** $T - 1$ **do**
    set $g_{t:T}$ to the downstream return from time t
    $\boldsymbol{\varphi} \leftarrow \boldsymbol{\varphi} + \eta \gamma^t g_{t:T} \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t)$ // (12.46)
until converged

The variance of REINFORCE can be reduced further. A common technique is to subtract a term b_t from the downstream returns,

$$\nabla_{\boldsymbol{\varphi}} j(\boldsymbol{\varphi}) = \mathbb{E}_{\tau \sim \Pi_{\boldsymbol{\varphi}}} \left[\sum_{t=0}^{T} \gamma^t (G_{t:T} - b_t) \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t) \right]. \tag{12.47}$$

For example, we can subtract the mean reward to go,

$$b_t \doteq \frac{1}{T} \sum_{t=0}^{T-1} G_{t:T}. \tag{12.48}$$

Exercise 12.12: Policy gradients with downstream returns

Suppose we are training an agent to solve a computer game. There are only two possible actions, specifically:

1. *do nothing*; and
2. *move*.

Each episode lasts for four ($T = 3$) time steps. The policy π_θ is completely determined by the parameter $\theta \in [0, 1]$. Here, for simplicity, we have assumed that the policy is independent of the current state. The probability of moving (action 2) is equal to θ and the probability of doing nothing (action 1) is $1 - \theta$.

Initially, $\theta = 0.5$. One episode is played with this initial policy and the results are

$$\text{actions} = (1, 0, 1, 0), \quad \text{rewards} = (1, 0, 1, 1).$$

Compute the policy gradient estimate with downstream returns, discount factor $\gamma = 1/2$, and the provided *single* sample $\tau \sim \Pi_\theta$.

▷ *Solution*

The main advantage of policy gradient methods such as REINFORCE is that they can be used in continuous action spaces. However, REINFORCE is not guaranteed to find an optimal policy. Even when

operating in very small domains, REINFORCE can get stuck in local optima.

Typically, policy gradient methods are slow due to the large variance in the score gradient estimates. Because of this, they need to take small steps and require many rollouts of a Markov chain. Moreover, we cannot reuse data from previous rollouts, as policy gradient methods are fundamentally on-policy.[11]

[11] This is because the score gradient estimator is used to obtain gradients of the policy value function with respect to the *current* policy.

Next, we will combine value approximation techniques like Q-learning and policy gradient methods, leading to an often more practical family of methods called actor-critic methods.

Readings

For more details on policy gradient methods, refer to chapter 13 (up to section 13.4) of "Reinforcement learning: An introduction" (Sutton and Barto, 2018).

12.4 Actor-Critic Methods

Actor-Critic methods reduce the variance of policy gradient estimates by using ideas from value function approximation. They use function approximation both to approximate value functions *and* to approximate policies. The goal for these algorithms is to scale to reinforcement learning problems, where we both have large state spaces and large action spaces.

12.4.1 Advantage Function

A key concept of actor-critic methods is the advantage function.

Definition 12.13 (Advantage function)**.** Given a policy π, the *advantage function*,

$$a^\pi(\boldsymbol{x}, \boldsymbol{a}) \doteq q^\pi(\boldsymbol{x}, \boldsymbol{a}) - v^\pi(\boldsymbol{x}) \tag{12.49}$$

$$= q^\pi(\boldsymbol{x}, \boldsymbol{a}) - \mathbb{E}_{\boldsymbol{a}' \sim \pi(\boldsymbol{x})}\left[q^\pi(\boldsymbol{x}, \boldsymbol{a}')\right], \tag{12.50}$$

using eq. (10.14)

measures the advantage of picking action $\boldsymbol{a} \in \mathcal{A}$ when in state $\boldsymbol{x} \in \mathcal{X}$ over simply following policy π.

It follows immediately from eq. (12.50) that for any policy π and state $\boldsymbol{x} \in \mathcal{X}$, there exists an action $\boldsymbol{a} \in \mathcal{A}$ such that $a^\pi(\boldsymbol{x}, \boldsymbol{a})$ is non-negative,

$$\max_{\boldsymbol{a} \in \mathcal{A}} a^\pi(\boldsymbol{x}, \boldsymbol{a}) \geq 0. \tag{12.51}$$

Moreover, it follows directly from Bellman's theorem (10.32) that

$$\pi \text{ is optimal} \iff \forall \boldsymbol{x} \in \mathcal{X}, \boldsymbol{a} \in \mathcal{A} : a^\pi(\boldsymbol{x}, \boldsymbol{a}) \leq 0. \tag{12.52}$$

In other words, quite intuitively, π is optimal if and only if there is no action that has an advantage in any state over the action that is played by π.

Finally, we can re-define the greedy policy π_q with respect to the state-action value function q as

$$\pi_q(\boldsymbol{x}) \doteq \arg\max_{\boldsymbol{a} \in \mathcal{A}} a(\boldsymbol{x}, \boldsymbol{a}) \tag{12.53}$$

since

$$\arg\max_{\boldsymbol{a} \in \mathcal{A}} a(\boldsymbol{x}, \boldsymbol{a}) = \arg\max_{\boldsymbol{a} \in \mathcal{A}} q(\boldsymbol{x}, \boldsymbol{a}) - v(\boldsymbol{x}) = \arg\max_{\boldsymbol{a} \in \mathcal{A}} q(\boldsymbol{x}, \boldsymbol{a}),$$

as $v(\boldsymbol{x})$ is independent of $\boldsymbol{a}$. This coincides with our initial definition of greedy policies in eq. (10.29). Intuitively, the advantage function is a shifted version of the state-action value function q that is relative to 0. Using this quantity rather than q often has numerical advantages.

12.4.2 *Policy Gradient Theorem*

Recall the score gradient estimator (12.45) that we had introduced in the previous section,

$$\boldsymbol{\nabla}_{\boldsymbol{\varphi}} j_T(\boldsymbol{\varphi}) = \mathbb{E}_{\tau \sim \Pi_{\boldsymbol{\varphi}}} \left[\sum_{t=0}^{T-1} \gamma^t G_{t:T} \boldsymbol{\nabla}_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t) \right].$$

Previously, we have approximated the policy value function $j(\boldsymbol{\varphi})$ by the bounded policy value function $j_T(\boldsymbol{\varphi})$. We said that this approximation was "reasonable" due to the diminishing returns. Essentially, we have "cut off the tails" of the policy value function. Let us now reinterpret score gradients while taking into account the tails of $j(\boldsymbol{\varphi})$.

$$\begin{aligned} \boldsymbol{\nabla}_{\boldsymbol{\varphi}} j(\boldsymbol{\varphi}) &= \lim_{T \to \infty} \boldsymbol{\nabla}_{\boldsymbol{\varphi}} j_T(\boldsymbol{\varphi}) \qquad (12.54) \\ &= \sum_{t=0}^{\infty} \mathbb{E}_{\tau \sim \Pi_{\boldsymbol{\varphi}}} \left[\gamma^t G_t \boldsymbol{\nabla}_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t) \right]. \end{aligned}$$

substituting the score gradient estimator with downstream returns (12.45) and using linearity of expectation (1.24)

Observe that because the expectations only consider downstream returns, we can disregard all data from the trajectory prior to time t. Let us define

$$\tau_{t:\infty} \doteq ((\boldsymbol{x}_t, \boldsymbol{a}_t, r_t, \boldsymbol{x}_{t+1}), (\boldsymbol{x}_{t+1}, \boldsymbol{a}_{t+1}, r_{t+1}, \boldsymbol{x}_{t+2}), \dots), \tag{12.55}$$

as the trajectory from time step t. Then,

$$= \sum_{t=0}^{\infty} \mathbb{E}_{\tau_{t:\infty} \sim \Pi_{\boldsymbol{\varphi}}} \left[\gamma^t G_t \boldsymbol{\nabla}_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t) \right].$$

We now condition on $\boldsymbol{x}_t$ and $\boldsymbol{a}_t$,

$$= \sum_{t=0}^{\infty} \mathbb{E}_{\boldsymbol{x}_t, \boldsymbol{a}_t}\big[\gamma^t \mathbb{E}_{r_t, \tau_{t+1:\infty}}[G_t \mid \boldsymbol{x}_t, \boldsymbol{a}_t] \, \boldsymbol{\nabla}_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t)\big].$$

using that $\pi_{\boldsymbol{\varphi}}(a_t \mid x_t)$ is a constant given x_t and a_t

Observe that averaging over the trajectories $\mathbb{E}_{\tau \sim \Pi_{\boldsymbol{\varphi}}}[\cdot]$ that are sampled according to policy $\pi_{\boldsymbol{\varphi}}$ is equivalent to our shorthand notation $\mathbb{E}_{\pi_{\boldsymbol{\varphi}}}[\cdot]$ from eq. (10.6),

$$= \sum_{t=0}^{\infty} \mathbb{E}_{\boldsymbol{x}_t, \boldsymbol{a}_t}\Big[\gamma^t \mathbb{E}_{\pi_{\boldsymbol{\varphi}}}[G_t \mid \boldsymbol{x}_t, \boldsymbol{a}_t] \, \boldsymbol{\nabla}_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t)\Big]$$

$$= \sum_{t=0}^{\infty} \mathbb{E}_{\boldsymbol{x}_t, \boldsymbol{a}_t}\big[\gamma^t q^{\pi_{\boldsymbol{\varphi}}}(\boldsymbol{x}_t, \boldsymbol{a}_t) \boldsymbol{\nabla}_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t)\big]. \tag{12.56}$$

using the definition of the Q-function (10.8)

It turns out that $\mathbb{E}_{\boldsymbol{x}_t, \boldsymbol{a}_t}[q^{\pi_{\boldsymbol{\varphi}}}(\boldsymbol{x}_t, \boldsymbol{a}_t)]$ exhibits much less variance than our previous estimator $\mathbb{E}_{\boldsymbol{x}_t, \boldsymbol{a}_t} \mathbb{E}_{\pi_{\boldsymbol{\varphi}}}[G_t \mid \boldsymbol{x}_t, \boldsymbol{a}_t]$. Equation (12.56) is known as the *policy gradient theorem*.

Often, the policy gradient theorem is stated in a slightly rephrased form in terms of the *discounted state occupancy measure*,[12]

$$\rho_{\boldsymbol{\varphi}}^{\infty}(\boldsymbol{x}) \doteq (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}_{\pi_{\boldsymbol{\varphi}}}(\mathbf{X}_t = \boldsymbol{x}). \tag{12.57}$$

[12] Depending on the reward setting, there exist various variations of the policy gradient theorem. We derived the variant for infinite-horizon discounted payoffs. "Reinforcement learning: An introduction" (Sutton and Barto, 2018) derive the variant for undiscounted average rewards.

The factor $(1-\gamma)$ ensures that $\rho_{\boldsymbol{\varphi}}^{\infty}$ is a probability density as

$$\int \rho_{\boldsymbol{\varphi}}^{\infty}(\boldsymbol{x}) \, d\boldsymbol{x} = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \int \mathbb{P}_{\pi_{\boldsymbol{\varphi}}}(\mathbf{X}_t = \boldsymbol{x}) \, d\boldsymbol{x} = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t = 1.$$

Intuitively, $\rho_{\boldsymbol{\varphi}}^{\infty}(\boldsymbol{x})$ measures how often we visit state $\boldsymbol{x}$ when following policy $\pi_{\boldsymbol{\varphi}}$. It can be thought of as a "discounted frequency".

Theorem 12.14 (Policy gradient theorem in terms of $\rho_{\boldsymbol{\varphi}}^{\infty}$)**.** *Policy gradients can be represented in terms of the Q-function,*

$$\boldsymbol{\nabla}_{\boldsymbol{\varphi}} j(\boldsymbol{\varphi}) \propto \mathbb{E}_{\boldsymbol{x} \sim \rho_{\boldsymbol{\varphi}}^{\infty}} \mathbb{E}_{\boldsymbol{a} \sim \pi_{\boldsymbol{\varphi}}(\cdot \mid \boldsymbol{x})}\big[q^{\pi_{\boldsymbol{\varphi}}}(\boldsymbol{x}, \boldsymbol{a}) \boldsymbol{\nabla}_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a} \mid \boldsymbol{x})\big]. \tag{12.58}$$

Proof. The right hand side of eq. (12.56) can be expressed as

$$\sum_{t=0}^{\infty} \int \mathbb{P}_{\pi_{\boldsymbol{\varphi}}}(\mathbf{X}_t = \boldsymbol{x}) \cdot \mathbb{E}_{\boldsymbol{a}_t \sim \pi_{\boldsymbol{\varphi}}(\cdot \mid \boldsymbol{x})}\big[\gamma^t q^{\pi_{\boldsymbol{\varphi}}}(\boldsymbol{x}, \boldsymbol{a}_t) \boldsymbol{\nabla}_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x})\big] \, d\boldsymbol{x}$$

$$= \frac{1}{1-\gamma} \int \rho_{\boldsymbol{\varphi}}^{\infty}(\boldsymbol{x}) \cdot \mathbb{E}_{\boldsymbol{a} \sim \pi_{\boldsymbol{\varphi}}(\cdot \mid \boldsymbol{x})}\big[q^{\pi_{\boldsymbol{\varphi}}}(\boldsymbol{x}, \boldsymbol{a}) \boldsymbol{\nabla}_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a} \mid \boldsymbol{x})\big] \, d\boldsymbol{x}$$

where we swapped the order of sum and integral and reorganized terms. □

Matching our intuition, according to the policy gradient theorem, maximizing $j(\boldsymbol{\varphi})$ corresponds to increasing the probability of actions with a large value and decreasing the probability of actions with a small value, taking into account how often the resulting policy visits certain states.

Exercise 12.15: Policy gradient with an exponential family

1. Suppose, we can choose between two actions $a \in \{0, 1\}$ in each state. A natural stochastic policy is induced by a Bernoulli distribution,
$$a \sim \text{Bern}(\sigma(f_{\boldsymbol{\varphi}}(\boldsymbol{x}))), \tag{12.59}$$
where σ is the logistic function from eq. (5.10). First, write down the expression for $\pi_{\boldsymbol{\varphi}}(a \mid \boldsymbol{x})$. Then, derive the expression for $\nabla_{\boldsymbol{\varphi}} j(\boldsymbol{\varphi})$ in terms of $q^{\pi_{\boldsymbol{\varphi}}}$, $\sigma(f_{\boldsymbol{\varphi}}(\boldsymbol{x}))$, and $\nabla_{\boldsymbol{\varphi}} f_{\boldsymbol{\varphi}}(\boldsymbol{x})$ using the policy gradient theorem.
2. The Bernoulli distribution is part of a family of distributions that allows for a much simpler derivation of the gradient than was necessary in (1). A univariate *exponential family* is a family of distributions whose PDF (or PMF) can be expressed in canonical form as
$$\pi_{\boldsymbol{\varphi}}(a \mid \boldsymbol{x}) = h(a) \exp(a f_{\boldsymbol{\varphi}}(\boldsymbol{x}) - A(f_{\boldsymbol{\varphi}}(\boldsymbol{x}))) \tag{12.60}$$
where h, f, and A are known functions.[13] Derive the expression of the policy gradient $\nabla_{\boldsymbol{\varphi}} j(\boldsymbol{\varphi})$ for such a distribution.
3. Can you relate the results of the previous two exercises (1) and (2)? What are h and A in case of the Bernoulli distribution?
4. The Gaussian distribution with unit variance $\mathcal{N}(f_{\boldsymbol{\varphi}}(\boldsymbol{x}), 1)$ is of the same canonical form with
$$A(f_{\boldsymbol{\varphi}}(\boldsymbol{x})) = \frac{f_{\boldsymbol{\varphi}}(\boldsymbol{x})^2}{2}. \tag{12.61}$$
Determine the policy gradient $\nabla_{\boldsymbol{\varphi}} j(\boldsymbol{\varphi})$.
5. For a Gaussian policy, can we instead apply the reparameterization trick (5.80) that we have seen in the context of variational inference? If yes, how? If not, why?

▷ *Solution*

[13] Observe that this form is equivalent to the form introduced in eq. (5.57) where $f_{\boldsymbol{\varphi}}(\boldsymbol{x})$ is the natural parameter, and we let $A(f) = \log Z(f)$.

Observe that we cannot use the policy gradient to calculate the gradient exactly, as we do not know $q^{\pi_{\boldsymbol{\varphi}}}$. Instead, we will use bootstrapping estimates $Q^{\pi_{\boldsymbol{\varphi}}}$ of $q^{\pi_{\boldsymbol{\varphi}}}$.

12.4.3 *On-policy Actor-Critics*

Actor-Critic methods consist of two components:

- a parameterized policy, $\pi(\boldsymbol{a} \mid \boldsymbol{x}; \boldsymbol{\varphi}) \doteq \pi_{\boldsymbol{\varphi}}$, which is called *actor*; and (12.62)
- a value function approximation, $q^{\pi_{\boldsymbol{\varphi}}}(\boldsymbol{x}, \boldsymbol{a}) \approx Q^{\pi_{\boldsymbol{\varphi}}}(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta})$, which is called *critic*. In the following, we will abbreviate $Q^{\pi_{\boldsymbol{\varphi}}}$ by Q. (12.63)

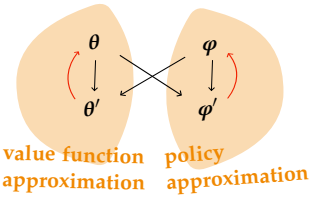


Figure 12.3: Illustration of one iteration of actor-critic methods. The dependencies between the actors and critics are shown as arrows. Methods differ in the exact order in which actor and critic are updated.

In deep reinforcement learning, neural networks are used to parameterize both actor and critic. Therefore, in principle, the actor-critic framework allows scaling to both large state spaces and large action spaces. We begin by discussing on-policy actor-critics.

One approach in the online setting (i.e., non-episodic setting), is to simply use SARSA for learning the critic. To learn the actor, we use stochastic gradient descent with gradients obtained using single samples from

$$\nabla_{\boldsymbol{\varphi}} J(\boldsymbol{\varphi}) \approx \sum_{t=0}^{\infty} \mathbb{E}_{(\boldsymbol{x}_t, \boldsymbol{a}_t) \sim \pi_{\boldsymbol{\varphi}}} \left[\gamma^t Q(\boldsymbol{x}_t, \boldsymbol{a}_t; \boldsymbol{\theta}) \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t) \right] \tag{12.64}$$

see eq. (12.56)

where Q is a bootstrapping estimate of $q^{\pi_{\boldsymbol{\varphi}}}$. This algorithm is known as *online actor-critic* or *Q actor-critic* and shown in alg. 12.16.

Algorithm 12.16: Online actor-critic

initialize parameters $\boldsymbol{\varphi}$ and $\boldsymbol{\theta}$
repeat
  use $\pi_{\boldsymbol{\varphi}}$ to obtain transition $(\boldsymbol{x}, \boldsymbol{a}, r, \boldsymbol{x}')$
  $\delta = r + \gamma Q(\boldsymbol{x}', \pi_{\boldsymbol{\varphi}}(\boldsymbol{x}'); \boldsymbol{\theta}) - Q(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta})$
  // actor update
  $\boldsymbol{\varphi} \leftarrow \boldsymbol{\varphi} + \eta \gamma^t Q(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}) \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a} \mid \boldsymbol{x})$ // (12.65)
  // critic update
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \delta \nabla_{\boldsymbol{\theta}} Q(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta})$ // (12.66)
until converged

Comparing to the derivation for TD-learning from eq. (12.8), we observe that eq. (12.66) corresponds to the SARSA update rule.[14] Due to the use of SARSA for learning the critic, this algorithm is fundamentally on-policy.

[14] The gradient with respect to $\boldsymbol{\theta}_Q$ appears analogously to our derivation of approximate Q-learning (12.15).

Crucially, by neglecting the dependence of the bootstrapping estimate Q on the policy parameters $\boldsymbol{\varphi}$, we introduce bias in the gradient estimates. In other words, using the bootstrapping estimate Q means that the resulting gradient direction might not be a valid ascent direction. In particular, the actor is not guaranteed to improve. Still, it turns out that under strong so-called "compatibility conditions" that are rarely satisfied in practice, a valid ascent direction can be guaranteed.

Reducing Variance To further reduce the variance of the gradient estimates, it turns out that a similar approach to the baselines we discussed in the previous section on policy gradient methods is useful.

A common approach is to subtract the state value function from estimates of the Q-function,

$$\boldsymbol{\varphi} \leftarrow \boldsymbol{\varphi} + \eta_t \gamma^t (Q(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}) - V(\boldsymbol{x}; \boldsymbol{\theta})) \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a} \mid \boldsymbol{x}) \tag{12.67}$$

$$= \boldsymbol{\varphi} + \eta_t \gamma^t A(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}) \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a} \mid \boldsymbol{x}) \tag{12.68}$$

using the definition of the advantage function (12.49)

where $A(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta})$ is a bootstrapped estimate of the advantage function $a^{\pi_{\boldsymbol{\varphi}}}$. This algorithm is known as *advantage actor-critic* (A2C) (Mnih et al., 2016). Recall that the Q-function is an absolute quantity, whereas the advantage function is a relative quantity, where the sign is informative for the gradient direction. Intuitively, an absolute value is harder to estimate than the sign. Actor-Critic methods are therefore often implemented with respect to the advantage function rather than the Q-function.

Taking a step back, observe that policy gradient methods such as REINFORCE generally have *high variance* in their gradient estimates. However, due to using Monte Carlo estimates of G_t, the gradient estimates are *unbiased*. In contrast, using a bootstrapped Q-function to obtain gradient estimates yields estimates with a *smaller variance*, but those estimates are *biased*. We are faced with a *bias-variance tradeoff*. A natural approach is therefore to blend both gradient estimates to allow for effectively trading bias and variance. This leads to algorithms such as *generalized advantage estimation* (GAE) (Schulman et al., 2015b).

Exploration Similarly to REINFORCE, actor-critic methods typically rely on randomization in the policy to encourage exploration, the idea being that if the policy is stochastic, then the agent will visit a diverse set of states. The inherent stochasticity of the policy is, however, often insufficient. A common problem is that the policy quickly "collapses" to a deterministic policy since the objective function is greedily exploitative. A common workaround is to use an ϵ-greedy policy (cf. section 11.3.1) or to explicitly encourage the policy to exhibit uncertainty by adding an entropy term to the objective function (more on this in section 12.5). However, note that for on-policy methods, changing the policy also changes the value function learned by the critic.

Improving sample efficiency Actor-Critic methods typically suffer from low sample efficiency. When additionally using an on-policy method, actor-critics often need an extremely large number of interactions before learning a near-optimal policy, because they cannot reuse past data. Allowing to reuse past data is a major advantage of off-policy methods like Q-learning.

One well-known variant that slightly improves the sample efficiency is *trust-region policy optimization* (TRPO) (Schulman et al., 2015a). TRPO

uses multiple iterations, where in each iteration a fixed critic is used to optimize the policy.[15] During iteration k, we have,

$$\boldsymbol{\varphi}_{k+1} \leftarrow \arg\max_{\boldsymbol{\varphi}} J(\boldsymbol{\varphi}) \quad \text{subject to } \mathbb{E}_{\boldsymbol{x}\sim\rho^{\infty}_{\boldsymbol{\varphi}_k}} \mathrm{KL}\big(\pi_{\boldsymbol{\varphi}_k}(\cdot \mid \boldsymbol{x}) \| \pi_{\boldsymbol{\varphi}}(\cdot \mid \boldsymbol{x})\big) \leq \delta \tag{12.69}$$

where

$$J(\boldsymbol{\varphi}) \doteq \mathbb{E}_{\boldsymbol{x}\sim\rho^{\infty}_{\boldsymbol{\varphi}_k}, \boldsymbol{a}\sim\pi_{\boldsymbol{\varphi}_k}(\cdot|\boldsymbol{x})} \left[\frac{\pi_{\boldsymbol{\varphi}}(\boldsymbol{a} \mid \boldsymbol{x})}{\pi_{\boldsymbol{\varphi}_k}(\boldsymbol{a} \mid \boldsymbol{x})} A^{\pi_{\boldsymbol{\varphi}_k}}(\boldsymbol{x}, \boldsymbol{a}) \right]. \tag{12.70}$$

Notably, J is an expectation with respect to the *previous* policy $\pi_{\boldsymbol{\varphi}_k}$ and the previous critic $A^{\pi_{\boldsymbol{\varphi}_k}}$. TRPO is an example of *importance weighting* where the importance weights,

$$\frac{\pi_{\boldsymbol{\varphi}}(\boldsymbol{a} \mid \boldsymbol{x})}{\pi_{\boldsymbol{\varphi}_k}(\boldsymbol{a} \mid \boldsymbol{x})},$$

are used to correct for using the previous policy. To be able to assume that the fixed critic is a good approximation within a certain "trust region" (i.e., one iteration), we impose the constraint

$$\mathbb{E}_{\boldsymbol{x}\sim\rho^{\infty}_{\boldsymbol{\varphi}_k}} \mathrm{KL}\big(\pi_{\boldsymbol{\varphi}_k}(\cdot \mid \boldsymbol{x}) \| \pi_{\boldsymbol{\varphi}}(\cdot \mid \boldsymbol{x})\big) \leq \delta$$

to optimize only in the "neighborhood" of the current policy. This constraint is also necessary for the importance weights not to blow up.

Taking the expectation with respect to the previous policy $\pi_{\boldsymbol{\varphi}_k}$ means that we can reuse data from rollouts within the same iteration. That is, TRPO allows reusing past data as long as it can still be "trusted". This makes TRPO "somewhat" off-policy. Fundamentally, though, TRPO is still an on-policy method.

Proximal policy optimization (PPO) is a heuristic variant of TRPO that often works well in practice (Schulman et al., 2017; Wang et al., 2020). It is based on controlling the importance weights directly rather than imposing a constraint on the KL-divergence. PPO is used, for example, to train large-scale language models such as GPT (Stiennon et al., 2020; OpenAI, 2023).

12.4.4 *Off-policy Actor-Critics*

In many applications, sample efficiency is crucial. Either because requiring too many interactions is computationally prohibitive or because obtaining sufficiently many samples for learning a near-optimal policy is simply impossible. We therefore now want to look at a separate family of actor-critic methods, which are off-policy, and hence, allow for the reuse of past data. These algorithms use the reparameterization gradient estimates, which we encountered before in the context of variational inference,[16] instead of score gradient estimators.

[15] Intuitively, each iteration performs a collection of gradient ascent steps.

[16] see section 5.5.1

The on-policy methods that we discussed in the previous section can be understood as performing a variant of *policy iteration*, where we use an estimate of the state-action value function of the current policy and then try to improve that policy by acting greedily with respect to this estimate. They mostly vary in how improving the policy is traded with improving the estimate of its value. Fundamentally, these methods rely on policy evaluation.[17]

[17] Policy evaluation is at the core of policy iteration. See alg. 10.17 for the definition of policy iteration and section 10.2 for a summary of policy evaluation in the context of Markov decision processes.

The techniques that we will introduce in this section are much more closely related to value iteration, essentially making use of Bellman's optimality principle to learn the optimal value function directly which characterizes the optimal policy.

To begin with, let us assume that the policy π is deterministic. We will later lift this restriction in section 12.4.5. Recall that our initial motivation to consider policy gradient methods and then actor-critic methods was the intractability of the DQN loss

$$\ell_{\text{DQN}}(\boldsymbol{\theta}; \mathcal{D}) = \frac{1}{2} \sum_{(\boldsymbol{x}, \boldsymbol{a}, r, \boldsymbol{x}') \in \mathcal{D}} \left(r + \gamma \max_{\boldsymbol{a}' \in \mathcal{A}} Q^\star(\boldsymbol{x}', \boldsymbol{a}'; \boldsymbol{\theta}^{\text{old}}) - Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}) \right)^2 \quad \text{see eq. (12.16)}$$

when the action space $\mathcal{A}$ is large. What if we simply replace the exact maximum over actions by a parameterized policy?

$$\ell_{\text{DQN}}(\boldsymbol{\theta}; \mathcal{D}) \approx \frac{1}{2} \sum_{(\boldsymbol{x}, \boldsymbol{a}, r, \boldsymbol{x}') \in \mathcal{D}} \left(r + \gamma Q^\star(\boldsymbol{x}', \pi_{\boldsymbol{\varphi}}(\boldsymbol{x}'); \boldsymbol{\theta}^{\text{old}}) - Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}) \right)^2. \tag{12.71}$$

We want to train our parameterized policy to learn the maximization over action, that is, to approximate the greedy policy[18]

[18] Here, we already apply the improvement of DDQN to use the most-recent estimate of the Q-function for action selection (see eq. (12.17)).

$$\pi_{\boldsymbol{\varphi}}(\boldsymbol{x}) \approx \pi^\star(\boldsymbol{x}) = \arg\max_{\boldsymbol{a} \in \mathcal{A}} Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}). \tag{12.72}$$

The key idea is that if we use a "rich-enough" parameterization of policies, selecting the greedy policy with respect to $Q^\star$ is equivalent to

$$\boldsymbol{\varphi}^\star = \arg\max_{\boldsymbol{\varphi}} \mathbb{E}_{\boldsymbol{x} \sim \mu} \left[Q^\star(\boldsymbol{x}, \pi_{\boldsymbol{\varphi}}(\boldsymbol{x}); \boldsymbol{\theta}) \right] \tag{12.73}$$

where $\mu(\boldsymbol{x}) > 0$ is an *exploration distribution* over states with full support.[19] We refer to this expectation by

[19] We require full support to ensure that all states are explored.

$$J_\mu(\boldsymbol{\varphi}; \boldsymbol{\theta}) \doteq \mathbb{E}_{x \sim \mu} \left[Q^\star(\boldsymbol{x}, \pi_{\boldsymbol{\varphi}}(\boldsymbol{x}); \boldsymbol{\theta}) \right]. \tag{12.74}$$

Commonly, the exploration distribution μ is taken to be the distribution that samples states uniformly from a replay buffer. Note that we can easily obtain unbiased gradient estimates of J_μ with respect to $\boldsymbol{\varphi}$:

$$\nabla_{\boldsymbol{\varphi}} J_\mu(\boldsymbol{\varphi}; \boldsymbol{\theta}) = \mathbb{E}_{x \sim \mu} \left[\nabla_{\boldsymbol{\varphi}} Q^\star(\boldsymbol{x}, \pi_{\boldsymbol{\varphi}}(\boldsymbol{x}); \boldsymbol{\theta}) \right]. \tag{12.75}$$

as we have seen in remark 1.22

Analogously to on-policy actor-critics (see eq. (12.64)), we use a bootstrapping estimate of $Q^\star$. That is, we neglect the dependence of the critic $Q^\star$ on the actor $\pi_{\boldsymbol{\varphi}}$, and in particular, the policy parameters $\boldsymbol{\varphi}$. We have seen that bootstrapping works with Q-learning, so there is reason to hope that it will work in this context too. This then allows us to use the chain rule to compute the gradient,

$$\nabla_{\boldsymbol{\varphi}} Q^\star(\boldsymbol{x}, \pi_{\boldsymbol{\varphi}}(\boldsymbol{x}); \boldsymbol{\theta}) = \boldsymbol{D}_{\boldsymbol{\varphi}} \pi_{\boldsymbol{\varphi}}(\boldsymbol{x}) \cdot \nabla_{\boldsymbol{a}} Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta})\big|_{\boldsymbol{a}=\pi_{\boldsymbol{\varphi}}(\boldsymbol{x})}. \tag{12.76}$$

This corresponds to evaluating the bootstrapping estimate of the Q-function at $\pi_{\boldsymbol{\varphi}}(\boldsymbol{x})$ and obtaining a gradient estimate of the policy estimate (e.g., through automatic differentiation). Note that as $\pi_{\boldsymbol{\varphi}}$ is vector-valued, $\boldsymbol{D}_{\boldsymbol{\varphi}} \pi_{\boldsymbol{\varphi}}(\boldsymbol{x})$ is the Jacobian of $\pi_{\boldsymbol{\varphi}}$ evaluated at $\boldsymbol{x}$.

Exploration Now that we have estimates of the gradient of our optimization target J_μ, it is natural to ask how we should select actions (based on $\pi_{\boldsymbol{\varphi}}$) to trade exploration and exploitation. As we have seen, policy gradient techniques rely on the randomness in the policy to explore, but here we consider deterministic policies. As our method is off-policy, a simple idea in continuous action spaces is to add Gaussian noise to the action selected by $\pi_{\boldsymbol{\varphi}}$ — also known as *Gaussian noise "dithering"*.[20] This corresponds to an algorithm called *deep deterministic policy gradients* (Lillicrap et al., 2015) shown in alg. 12.17. This algorithm is essentially equivalent to Q-learning with function approximation (e.g., DQN),[21] with the only exception that we replace the maximization over actions with the learned policy $\pi_{\boldsymbol{\varphi}}$.

[20] Intuitively, this adds "additional randomness" to the policy $\pi_{\boldsymbol{\varphi}}$.

[21] see eq. (12.16)

Twin delayed DDPG (TD3) is an extension of DDPG that uses two separate critic networks for predicting the maximum action and evaluating the policy (Fujimoto et al., 2018). This addresses the maximization bias akin to Double-DQN. TD3 also applies delayed updates to the actor network, which increases stability.

12.4.5 Off-Policy Actor Critics with Randomized Policies

We have seen that randomized policies naturally encourage exploration. With deterministic actor-critic methods like DDPG, we had to inject Gaussian noise to enforce sufficient exploration. A natural question is therefore whether we can also handle randomized policies in this framework of off-policy actor-critics.

The key idea is to replace the squared loss of the critic,

$$\ell_{\mathrm{DQN}}(\boldsymbol{\theta}; \mathcal{D}) \approx \frac{1}{2}\Big(r + \gamma Q^\star(\boldsymbol{x}', \boldsymbol{\pi}(\boldsymbol{x}'; \boldsymbol{\varphi}); \boldsymbol{\theta}^{\mathrm{old}}) - Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta})\Big)^2,$$

refer to the squared loss of Q-learning (12.11)

Algorithm 12.17: Deep deterministic policy gradients, DDPG

initialize $\boldsymbol{\varphi}, \boldsymbol{\theta}$, a (possibly non-empty) replay buffer $\mathcal{D} = \emptyset$
set $\boldsymbol{\varphi}^{\text{old}} = \boldsymbol{\varphi}$ and $\boldsymbol{\theta}^{\text{old}} = \boldsymbol{\theta}$
for $t = 0$ **to** ∞ **do**
    observe state $\boldsymbol{x}$, pick action $\boldsymbol{a} = \boldsymbol{\pi}_{\boldsymbol{\varphi}}(\boldsymbol{x}) + \boldsymbol{\epsilon}$ for $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \lambda \boldsymbol{I},)$
    execute $\boldsymbol{a}$, observe r and $\boldsymbol{x}'$
    add $(\boldsymbol{x}, \boldsymbol{a}, r, \boldsymbol{x}')$ to the replay buffer $\mathcal{D}$
    if `collected "enough" data` **then**
        `// policy improvement step`
        for `some iterations` **do**
            sample a mini-batch B of $\mathcal{D}$
            for each transition in B, compute the label $y = r + \gamma Q^\star(\boldsymbol{x}', \boldsymbol{\pi}(\boldsymbol{x}'; \boldsymbol{\varphi}^{\text{old}}); \boldsymbol{\theta}^{\text{old}})$
            `// critic update`
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \boldsymbol{\nabla}_{\boldsymbol{\theta}} \frac{1}{B} \sum_{(\boldsymbol{x}, \boldsymbol{a}, r, \boldsymbol{x}', y) \in B} (y - Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}))^2$
            `// actor update`
            $\boldsymbol{\varphi} \leftarrow \boldsymbol{\varphi} + \eta \boldsymbol{\nabla}_{\boldsymbol{\varphi}} \frac{1}{B} \sum_{(\boldsymbol{x}, \boldsymbol{a}, r, \boldsymbol{x}', y) \in B} Q^\star(\boldsymbol{x}, \boldsymbol{\pi}(\boldsymbol{x}; \boldsymbol{\varphi}); \boldsymbol{\theta})$
            $\boldsymbol{\theta}^{\text{old}} \leftarrow (1 - \rho) \boldsymbol{\theta}^{\text{old}} + \rho \boldsymbol{\theta}$
            $\boldsymbol{\varphi}^{\text{old}} \leftarrow (1 - \rho) \boldsymbol{\varphi}^{\text{old}} + \rho \boldsymbol{\varphi}$

which only considers the fixed action $\pi(\boldsymbol{x}'; \boldsymbol{\varphi}^{\text{old}})$ with an expected squared loss,

$$\ell_{\text{DQN}}(\boldsymbol{\theta}; \mathcal{D}) \approx \mathbb{E}_{\boldsymbol{a}' \sim \pi(\boldsymbol{x}'; \boldsymbol{\varphi})} \left[\frac{1}{2} \Big(r + \gamma Q^\star(\boldsymbol{x}', \boldsymbol{a}'; \boldsymbol{\theta}^{\text{old}}) - Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}) \Big)^2 \right], \tag{12.77}$$

which considers a distribution over actions.

It turns out that we can still compute gradients of this expectation.

$$\begin{aligned} &\nabla_{\boldsymbol{\theta}} \, \mathbb{E}_{\boldsymbol{a}' \sim \pi(\boldsymbol{x}'; \boldsymbol{\varphi})} \left[\frac{1}{2} \Big(r + \gamma Q^\star(\boldsymbol{x}', \boldsymbol{a}'; \boldsymbol{\theta}^{\text{old}}) - Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}) \Big)^2 \right] \\ &= \mathbb{E}_{\boldsymbol{a}' \sim \pi(\boldsymbol{x}'; \boldsymbol{\varphi})} \left[\nabla_{\boldsymbol{\theta}} \, \frac{1}{2} \Big(r + \gamma Q^\star(\boldsymbol{x}', \boldsymbol{a}'; \boldsymbol{\theta}^{\text{old}}) - Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}) \Big)^2 \right]. \end{aligned}$$

as we have seen in remark 1.22

Similarly to our definition of the Bellman error (12.12), we define by

$$\delta_{\text{B}}(\boldsymbol{a}') \doteq r + \gamma Q^\star(\boldsymbol{x}', \boldsymbol{a}'; \boldsymbol{\theta}^{\text{old}}) - Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}), \tag{12.78}$$

the *Bellman error* for a fixed action $\boldsymbol{a}'$. Using the chain rule, we obtain

$$= \mathbb{E}_{\boldsymbol{a}' \sim \pi(\boldsymbol{x}'; \boldsymbol{\varphi})} \big[\delta_{\text{B}}(\boldsymbol{a}') \nabla_{\boldsymbol{\theta}} \, Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}) \big]. \tag{12.79}$$

Note that this is identical to the gradient in DQN (12.14), except that now we have an expectation over actions. As we have done many times already, we can use automatic differentiation to obtain gradient estimates of $\nabla_{\boldsymbol{\theta}} \, Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta})$. This provides us with a method of obtaining unbiased gradient estimates for the critic.

We also need to reconsider the actor update. When using a randomized policy, the objective function changes to

$$J_\mu(\boldsymbol{\varphi}; \boldsymbol{\theta}) \doteq \mathbb{E}_{\boldsymbol{x} \sim \mu} \mathbb{E}_{\boldsymbol{a} \sim \pi(\boldsymbol{x}; \boldsymbol{\varphi})} [Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta})].$$

of which we can obtain gradients via

$$\nabla_{\boldsymbol{\varphi}} \, J_\mu(\boldsymbol{\varphi}; \boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x} \sim \mu} \nabla_{\boldsymbol{\varphi}} \, \mathbb{E}_{\boldsymbol{a} \sim \pi(\boldsymbol{x}; \boldsymbol{\varphi})} [Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta})]. \tag{12.80}$$

see remark 1.22

Note that the inner expectation is with respect to a measure that depends on the parameters $\boldsymbol{\varphi}$, which we are trying to optimize. We therefore cannot move the gradient operator inside the expectation. This is a problem that we have already encountered several times. In the previous section on policy gradients, we used the score gradient estimator.[22] Earlier, in chapter 5 on variational inference we have already seen reparameterization gradients.[23] Here, if our policy is reparameterizable, we can use the *reparameterization trick* from theorem 5.31!

[22] see eq. (12.32)

[23] see (5.78)

Example 12.18: Reparameterization gradients for Gaussians

Suppose we use a Gaussian parameterization of policies,

$$\pi(\boldsymbol{x};\boldsymbol{\varphi}) \doteq \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x};\boldsymbol{\varphi}), \boldsymbol{\Sigma}(\boldsymbol{x};\boldsymbol{\varphi})).$$

Then, using conditional linear Gaussians, our action $\boldsymbol{a}$ is given by

$$\boldsymbol{a} = \boldsymbol{g}(\boldsymbol{\epsilon};\boldsymbol{x},\boldsymbol{\varphi}) \doteq \boldsymbol{\Sigma}^{1/2}(\boldsymbol{x};\boldsymbol{\varphi})\boldsymbol{\epsilon} + \boldsymbol{\mu}(\boldsymbol{x};\boldsymbol{\varphi}), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \tag{12.81}$$

where $\boldsymbol{\Sigma}^{1/2}(\boldsymbol{x};\boldsymbol{\varphi})$ is the square root of $\boldsymbol{\Sigma}(\boldsymbol{x};\boldsymbol{\varphi})$. This coincides with our earlier application of the reparameterization trick to Gaussians in example 5.32.

As we have seen, not only Gaussians are reparameterizable. In general, we called a distribution (in this context, a policy) reparameterizable iff $\boldsymbol{a} \sim \pi(\boldsymbol{x};\boldsymbol{\varphi})$ is such that $\boldsymbol{a} = \boldsymbol{g}(\boldsymbol{\epsilon};\boldsymbol{x},\boldsymbol{\varphi})$, where $\boldsymbol{\epsilon} \sim \phi$ is an independent random variable.

Then, we have,

$$\begin{aligned}
&\nabla_{\boldsymbol{\varphi}} \mathbb{E}_{\boldsymbol{a}\sim\pi(\boldsymbol{x};\boldsymbol{\varphi})}[Q^\star(\boldsymbol{x},\boldsymbol{a};\boldsymbol{\theta})] \\
&= \mathbb{E}_{\boldsymbol{\epsilon}\sim\phi}\left[\nabla_{\boldsymbol{\varphi}} Q^\star(\boldsymbol{x},\boldsymbol{g}(\boldsymbol{\epsilon};\boldsymbol{x},\boldsymbol{\varphi});\boldsymbol{\theta})\right] && (12.82) \quad \text{using the reparameterization trick (5.78)} \\
&= \mathbb{E}_{\boldsymbol{\epsilon}\sim\phi}\left[\nabla_{\boldsymbol{a}} Q^\star(\boldsymbol{x},\boldsymbol{a};\boldsymbol{\theta})\big|_{\boldsymbol{a}=\boldsymbol{g}(\boldsymbol{\epsilon};\boldsymbol{x},\boldsymbol{\varphi})} \cdot \boldsymbol{D}_{\boldsymbol{\varphi}}\boldsymbol{g}(\boldsymbol{\epsilon};\boldsymbol{x},\boldsymbol{\varphi})\right]. && (12.83) \quad \text{using the chain rule analogously to eq. (12.76)}
\end{aligned}$$

In this way, we can obtain unbiased gradient estimates for reparameterizable policies. This general technique does not only apply to continuous action spaces. For discrete action spaces, there is the analogous so-called *Gumbel-max trick*, which we will not discuss in greater detail here.

The algorithm that uses eq. (12.79) to obtain gradients for the critic and reparameterization gradients for the actor is called *stochastic value gradients* (SVG) (Heess et al., 2015).

Readings

For more details on actor-critic methods, refer to chapter 13 (from section 13.5) of "Reinforcement learning: An introduction" (Sutton and Barto, 2018).

12.5 Maximum Entropy Reinforcement Learning

In practice, algorithms like SVG often do not explore enough. A key issue with relying on randomized policies for exploration is that they might collapse to deterministic policies. That is, the algorithm might

quickly reach a local optimum, where all mass is placed on a single action.

A simple trick that encourages a little bit of extra exploration is to regularize the randomized policies "away" from putting all mass on a single action. In other words, we want to encourage the policies to exhibit some uncertainty. A natural measure of uncertainty is entropy, which we have already seen several times.[24] This approach is known as *entropy regularization* or *maximum entropy reinforcement learning* (MERL). Canonically, entropy regularization is applied to finite-horizon rewards (cf. remark 10.5), yielding the optimization problem of maximizing

[24] see section 5.4

$$j_\lambda(\boldsymbol{\varphi}) \doteq j(\boldsymbol{\varphi}) + \lambda \mathrm{H}\big[\Pi_{\boldsymbol{\varphi}}\big] \tag{12.84}$$

$$= \sum_{t=1}^{T} \mathbb{E}_{(\boldsymbol{x}_t, \boldsymbol{a}_t) \sim \Pi_{\boldsymbol{\varphi}}} \big[r(\boldsymbol{x}_t, \boldsymbol{a}_t) + \lambda \mathrm{H}\big[\pi_{\boldsymbol{\varphi}}(\cdot \mid \boldsymbol{x}_t)\big] \big], \tag{12.85}$$

where we have a preference for entropy in the actor distribution to encourage exploration which is regulated by the temperature parameter λ. As $\lambda \to 0$, we recover the "standard" reinforcement learning objective (here for finite-horizon rewards):

$$j(\boldsymbol{\varphi}) = \sum_{t=1}^{T} \mathbb{E}_{(\boldsymbol{x}_t, \boldsymbol{a}_t) \sim \Pi_{\boldsymbol{\varphi}}} [r(\boldsymbol{x}_t, \boldsymbol{a}_t)]. \tag{12.86}$$

Here, for notational convenience, we begin the sum with $t = 1$ rather than $t = 0$.

12.5.1 *Entropy Regularization as Probabilistic Inference*

The entropy-regularized objective from eq. (12.85) leads us to a remarkable interpretation of reinforcement learning and, more generally, decision-making under uncertainty as solving an inference problem akin to variational inference. The framing of "control as inference" will lead us to contemporary algorithms for reinforcement learning as well as paint a path for decision-making under uncertainty beyond stationary MDPs.

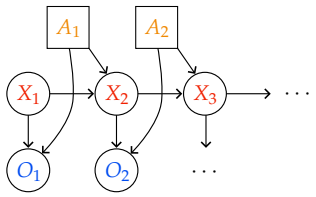


Figure 12.4: Directed graphical model of the underlying hidden Markov model with hidden states X_t, optimality variables O_t, and actions A_t.

Let us denote by $\Pi_\star$ the distribution over trajectories τ under the optimal policy $\pi^\star$. By framing the problem of optimal control as an inference problem in a hidden Markov model with hidden "optimality variables" $O_t \in \{0, 1\}$ indicating whether the played action $\boldsymbol{a}_t$ was optimal we can derive $\Pi_\star$ analytically. That is to say, when $O_t = 1$ and $O_{t+1:T} \equiv 1$ the policy from time t onwards was optimal. To simplify the notation, we will denote the event $O_t = 1$ by $\mathcal{O}_t$.

We consider the HMM defined by the Gibbs distribution

$$p(\mathcal{O}_t \mid \boldsymbol{x}_t, \boldsymbol{a}_t) \propto \exp\left(\frac{1}{\lambda} r(\boldsymbol{x}_t, \boldsymbol{a}_t)\right), \quad \text{with } \lambda > 0 \tag{12.87}$$

which is a natural choice as we have seen in exercise 6.25 that the Gibbs distribution maximizes entropy subject to $\mathbb{E}[O_t \cdot r(\boldsymbol{x}_t, \boldsymbol{a}_t) \mid \boldsymbol{x}_t, \boldsymbol{a}_t] < \infty$.

The distribution over trajectories conditioned on optimality of actions (i.e., conditioned on $\mathcal{O}_{1:T}$) is given by

$$\Pi_\star(\tau) \doteq p(\tau \mid \mathcal{O}_{1:T}) = p(\boldsymbol{x}_1) \prod_{t=1}^{T} p(\boldsymbol{a}_t \mid \boldsymbol{x}_t, \mathcal{O}_t) p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{a}_t). \tag{12.88}$$

Using eq. (12.30). We assume here that the dynamics and initial state distribution are "fixed", that is, we assume $p(\boldsymbol{x}_1 \mid \mathcal{O}_{1:T}) = p(\boldsymbol{x}_1)$ and $p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{a}_t, \mathcal{O}_{1:T}) = p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{a}_t)$.

It remains to determine $p(\boldsymbol{a}_t \mid \boldsymbol{x}_t, \mathcal{O}_t)$ which corresponds to the optimal policy $\pi^\star(\boldsymbol{a}_t \mid \boldsymbol{x}_t)$. It is generally useful to think of the situation where the prior policy $p(\boldsymbol{a}_t \mid \boldsymbol{x}_t)$ is uniform on $\mathcal{A}$,[25] in which case by Bayes' rule (1.59), $p(\boldsymbol{a}_t \mid \boldsymbol{x}_t, \mathcal{O}_t) \propto p(\mathcal{O}_t \mid \boldsymbol{x}_t, \boldsymbol{a}_t)$, so

$$\Pi_\star(\tau) \propto \left[p(\boldsymbol{x}_1) \prod_{t=1}^{T} p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{a}_t)\right] \exp\left(\frac{1}{\lambda} \sum_{t=1}^{T} r(\boldsymbol{x}_t, \boldsymbol{a}_t)\right). \tag{12.89}$$

[25] This is not a restriction as any non-uniform prior can be pushed into eq. (12.87).

Recall that our fundamental goal is to approximate $\Pi_\star$ with a distribution over trajectories $\Pi_{\boldsymbol{\varphi}}$ under the parameterized policy $\pi_{\boldsymbol{\varphi}}$. It is therefore a natural idea to minimize $\mathrm{KL}\big(\Pi_{\boldsymbol{\varphi}} \| \Pi_\star\big)$:[26]

[26] Observe that we cannot easily minimize forward-KL as we cannot sample from $\Pi_\star$. In the context of RL, it can be argued that the mode-seeking behavior of reverse-KL is preferable over the moment-matching behavior of forward-KL (Levine, 2018).

$$\begin{aligned}
&\arg\min_{\boldsymbol{\varphi}} \mathrm{KL}\big(\Pi_{\boldsymbol{\varphi}} \| \Pi_\star\big) \\
&= \arg\min_{\boldsymbol{\varphi}} \mathrm{H}\big[\Pi_{\boldsymbol{\varphi}} \| \Pi_\star\big] - \mathrm{H}\big[\Pi_{\boldsymbol{\varphi}}\big] && \text{using the definition of KL-divergence (5.44)} \\
&= \arg\max_{\boldsymbol{\varphi}} \mathbb{E}_{\tau \sim \Pi_{\boldsymbol{\varphi}}}\big[\log \Pi_\star(\tau) - \log \Pi_{\boldsymbol{\varphi}}(\tau)\big] && \text{using the definition of cross-entropy (5.40) and entropy (5.32)} \\
&= \arg\max_{\boldsymbol{\varphi}} \mathbb{E}_{\tau \sim \Pi_{\boldsymbol{\varphi}}}\left[\sum_{t=1}^{T} r(\boldsymbol{x}_t, \boldsymbol{a}_t) - \lambda \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t)\right] && \text{using eqs. (12.30) and (12.89) and simplifying} \\
&= \arg\max_{\boldsymbol{\varphi}} \sum_{t=1}^{T} \mathbb{E}_{(\boldsymbol{x}_t, \boldsymbol{a}_t) \sim \Pi_{\boldsymbol{\varphi}}}\big[r(\boldsymbol{x}_t, \boldsymbol{a}_t) + \lambda \mathrm{H}\big[\pi_{\boldsymbol{\varphi}}(\cdot \mid \boldsymbol{x}_t)\big]\big]. && \text{using the definition of entropy (5.32) and linearity of expectation (1.24)}
\end{aligned} \tag{12.90}$$

That is, entropy regularization is equivalent to minimizing the KL-divergence from $\Pi_\star$ to $\Pi_{\boldsymbol{\varphi}}$. This highlights a very natural tradeoff between exploration and exploitation, wherein $\mathrm{H}\big[\Pi_{\boldsymbol{\varphi}} \| \Pi_\star\big]$ encourages exploitation and $\mathrm{H}\big[\Pi_{\boldsymbol{\varphi}}\big]$ encourages exploration.

It can be shown that a "softmax" version of the Bellman optimality equation (10.35) can be obtained for eq. (12.90):

$$q^\star(\boldsymbol{x}, \boldsymbol{a}) = \frac{1}{\lambda} r(\boldsymbol{x}, \boldsymbol{a}) + \mathbb{E}_{\boldsymbol{x}' \sim \boldsymbol{x}, \boldsymbol{a}}\left[\log \int_{\mathcal{A}} \exp(q^\star(\boldsymbol{x}', \boldsymbol{a}'))\, d\boldsymbol{a}'\right] \tag{12.91}$$

and the optimal policy has the form $\pi^\star(\boldsymbol{a} \mid \boldsymbol{x}) \propto \exp(q^\star(\boldsymbol{x}, \boldsymbol{a}))$. Here, $q^\star$ is called a *soft value function*. The second term of eq. (12.91) quantifies downstream rewards. In comparison to the "standard" Bellman optimality equation (10.35), the soft value function is less greedy which tends to encourage robustness.

Exercise 12.19: Soft value function

In this exercise, we derive the soft value function from eq. (12.91).

1. We let $\beta(\boldsymbol{a}_t \mid \boldsymbol{x}_t) \doteq p(\boldsymbol{a}_t \mid \boldsymbol{x}_t, \mathcal{O}_t)$, $Z(\boldsymbol{x}) \doteq \int_{\mathcal{A}} \beta(\boldsymbol{a} \mid \boldsymbol{x}) \, d\boldsymbol{a}$, and denote by $\hat{\pi}(\cdot \mid \boldsymbol{x})$ the policy $\beta(\cdot \mid \boldsymbol{x}) / Z(\boldsymbol{x})$. Show that

$$\begin{aligned} &\mathrm{KL}\big(\Pi_{\boldsymbol{\varphi}} \| \Pi_\star\big) \\ &\quad = \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{x}_t \sim \Pi_{\boldsymbol{\varphi}}} \big[\mathrm{KL}\big(\pi_{\boldsymbol{\varphi}}(\cdot \mid \boldsymbol{x}_t) \| \hat{\pi}(\cdot \mid \boldsymbol{x}_t)\big) - \log Z(\boldsymbol{x}_t)\big]. \end{aligned} \tag{12.92}$$

2. Show that if the space of policies parameterized by $\boldsymbol{\varphi}$ is sufficiently expressive, $\pi^\star(\boldsymbol{a} \mid \boldsymbol{x}) \propto \exp(q^\star(\boldsymbol{x}, \boldsymbol{a}))$ solves eq. (12.90).

▷ *Solution*

Analogously to Q-learning, the soft value function $q^\star$ can be approximated via a bootstrapped "critic" $Q^\star$ which is called *soft Q-learning* (Levine, 2018). Note that computing the optimal policy requires computing an integral over the action space, which is typically intractable for continuous action spaces. As discussed in sections 12.3 and 12.4 and analogously to actor-critic methods such as DDPG and SVG, we can learn a parameterized policy (i.e., an "actor") $\pi_{\boldsymbol{\varphi}}$ to approximate the optimal policy $\pi^\star$. The resulting algorithm, *soft actor critic* (SAC) (Haarnoja et al., 2018a,b), is widely used. Due to its off-policy nature, it is also relatively sample efficient.

We can also express this optimization in terms of an evidence lower bound. The evidence lower bound for the observations $\mathcal{O}_{1:T}$ is

$$L(\Pi_{\boldsymbol{\varphi}}, \Pi_\star; \mathcal{O}_{1:T}) = \mathbb{E}_{\tau \sim \Pi_{\boldsymbol{\varphi}}} \big[\log p(\mathcal{O}_{1:T} \mid \tau) + \log \Pi(\tau) - \log \Pi_{\boldsymbol{\varphi}}(\tau)\big] \tag{12.93}$$

using the definition of the ELBO (5.63c)

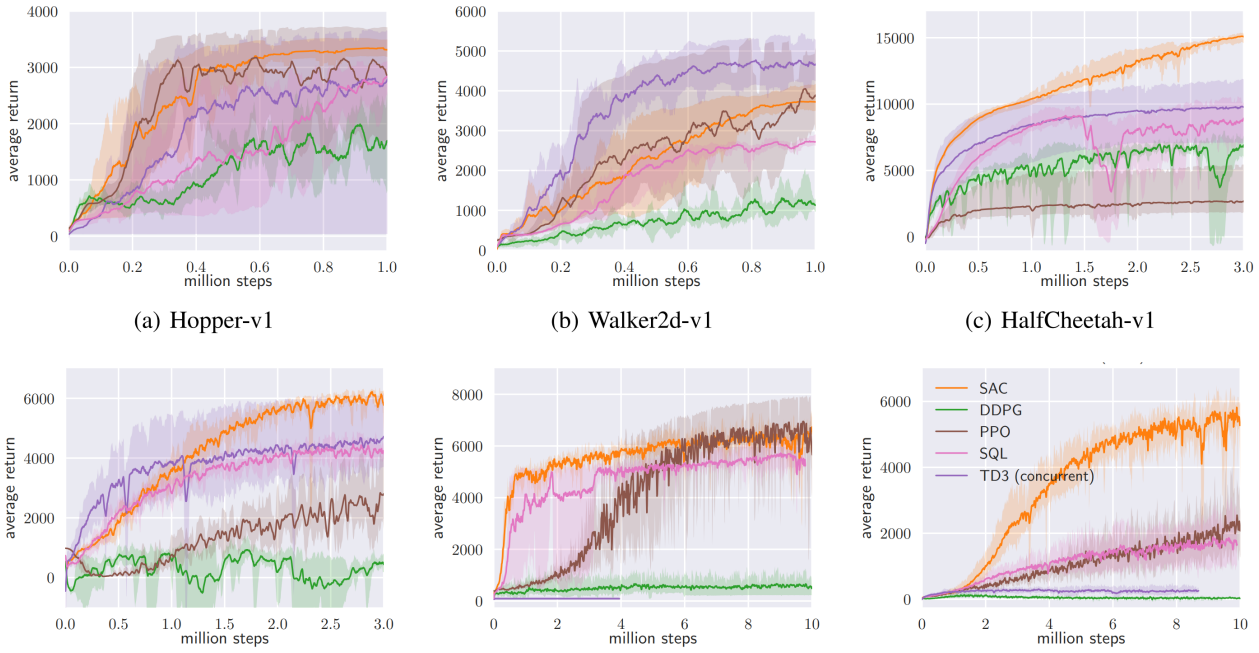
where Π denotes the distribution over trajectories (12.30) under the prior policy $p(\boldsymbol{a}_t \mid \boldsymbol{x}_t)$. This is commonly written as the variational free energy

$$-L(\Pi_{\boldsymbol{\varphi}}, \Pi_\star; \mathcal{O}_{1:T}) = \mathbb{E}_{\tau \sim \Pi_{\boldsymbol{\varphi}}}[\mathrm{S}[p(\mathcal{O}_{1:T} \mid \tau)]] + \mathrm{KL}\big(\Pi_{\boldsymbol{\varphi}} \| \Pi\big) \tag{12.94}$$

$$= \underbrace{\mathrm{S}[p(\mathcal{O}_{1:T})]}_{\text{"extrinsic" value}} + \underbrace{\mathrm{KL}\big(\Pi_{\boldsymbol{\varphi}} \| \Pi_\star\big)}_{\text{"epistemic" value}}. \tag{12.95}$$

using that $\Pi_\star(\tau) = p(\tau \mid \mathcal{O}_{1:T})$

which we already encountered in eq. (5.68) in the context of variational



(a) Hopper-v1 (b) Walker2d-v1 (c) HalfCheetah-v1

(d) Ant-v1 (e) Humanoid-v1 (f) Humanoid (rllab)

Figure 12.5: Comparison of training curves of a selection of on-policy and off-policy policy gradient methods. Taken from "Soft actor-critic algorithms and applications" (Haarnoja et al., 2018b).

inference. To summarize, we have seen that

$$\arg\max_{\boldsymbol{\varphi}} j_\lambda(\boldsymbol{\varphi}) = \arg\min_{\boldsymbol{\varphi}} \mathrm{KL}\big(\Pi_{\boldsymbol{\varphi}} \| \Pi_\star\big) = \arg\max_{\boldsymbol{\varphi}} L(\Pi_{\boldsymbol{\varphi}}, \Pi_\star; \mathcal{O}_{1:T}).$$

Recall that the free energy $-L(\Pi_{\boldsymbol{\varphi}}, \Pi_\star; \mathcal{O}_{1:T})$ is a variational upper bound to the surprise about observations $\mathrm{S}[\Pi_\star(\mathcal{O}_{1:T})]$ when following an optimal policy.[27] For example, undesirable states incur a low reward while desirable states yield a high reward, and thus, if we expect optimality of actions (i.e., $\mathcal{O}_{1:T}$) paths leading to such states have high and low surprise, respectively.[28]

An agent which acts to minimize free energy with $\mathcal{O}_{1:T}$ can be thought of as hallucinating to perform optimally, and acting to minimize the surprise about having played suboptimal actions. Think, for example, about the robotics task of moving an arm to a new position. Intuitively, minimizing free energy solves this task by "hallucinating that the arm is at the goal position", and then minimizing the surprise with respect to this perturbed world model. In this way, MERL can be understood as identifying paths of least surprise akin to the *free energy principle*.

[27] see remark 5.27

[28] This is because, a path leading to a low reward state will include sub-optimal actions.

Optional Readings

For more details, read "Reinforcement learning and control as probabilistic inference: Tutorial and review" (Levine, 2018).

Remark 12.20: Towards active inference

Maximum entropy reinforcement learning makes one fundamental assumption, namely, that the "biased" distribution about observations specifying the underlying HMM is

$$p(\mathcal{O}_t \mid \boldsymbol{x}_t, \boldsymbol{a}_t) \propto \exp\left(\frac{1}{\lambda} r(\boldsymbol{x}_t, \boldsymbol{a}_t)\right).$$

This assumption is how the task of optimal control (in an unknown MDP) with a certain reward function is framed as an inference problem. One could conceive other HMMs. For example, Fellows et al. (2019) propose an HMM defined in terms of the current value function of the agent:

$$p(\mathcal{O}_t \mid \boldsymbol{x}_t, \boldsymbol{a}_t) \propto \exp\left(\frac{1}{\lambda} Q(\boldsymbol{x}_t, \boldsymbol{a}_t; \boldsymbol{\theta})\right).$$

Crucially, the choice of $p(\mathcal{O}_t \mid \boldsymbol{x}_t, \boldsymbol{a}_t)$ is the only place where the reward enters the inference problem, and one can conceive of settings where a "stationary" (i.e., time-independent) reward is not assumed to exist. The general approach to decision-making as probabilistic inference presented in this section (but for possibly reward-independent HMMs) is known as *active inference* (Friston et al., 2015; Millidge et al., 2020, 2021).

12.5.2 Summary

In this chapter, we studied central ideas in actor-critic methods. We have seen two main approaches to use policy-gradient methods. We began, in section 12.3, by introducing the REINFORCE algorithm which uses policy gradients and Monte Carlo estimation, but suffered from large variance in the gradient estimates of the policy value function. We have then seen a number of actor-critic methods such as A2C and GAE behaving similarly to policy iteration that exhibit less variance, but are very sample inefficient due to their on-policy nature. TRPO improves the sample efficiency slightly, but not fundamentally.

We then discussed a second family of policy gradient techniques that generalize Q-learning and are akin to value iteration. For reparameterizable policies, this led us to algorithms such as DDPG, TD3, SVG,

and SAC that are widely used and work quite well in practice. Importantly, they are significantly more sample efficient than on-policy policy gradient methods, which often results in much faster learning of a near-optimal policy.

Optional Readings

- **A3C:** Mnih, Badia, Mirza, Graves, Lillicrap, Harley, Silver, and Kavukcuoglu (2016).
  Asynchronous methods for deep reinforcement learning.
- **GAE:** Schulman, Moritz, Levine, Jordan, and Abbeel (2015b).
  High-dimensional continuous control using generalized advantage estimation.
- **TRPO:** Schulman, Levine, Abbeel, Jordan, and Moritz (2015a).
  Trust region policy optimization.
- **PPO:** Schulman, Wolski, Dhariwal, Radford, and Klimov (2017).
  Proximal policy optimization algorithms.
- **DDPG:** Lillicrap, Hunt, Pritzel, Heess, Erez, Tassa, Silver, and Wierstra (2015).
  Continuous control with deep reinforcement learning.
- **TD3:** Fujimoto, Hoof, and Meger (2018).
  Addressing function approximation error in actor-critic methods.
- **SVG:** Heess, Wayne, Silver, Lillicrap, Erez, and Tassa (2015).
  Learning continuous control policies by stochastic value gradients.
- **SAC:** Haarnoja, Zhou, Abbeel, and Levine (2018a).
  Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.

13

Model-based Reinforcement Learning

In this final chapter, we will revisit the model-based approach to reinforcement learning. We will see some advantages it offers over model-free methods. In particular, we will use the machinery of Bayesian learning, which we developed in the first chapters, to quantify uncertainty about our model and use this uncertainty for planning, exploration, and reliable decision-making.

To recap, in chapter 11, we began by discussing model-based reinforcement learning which attempts to learn the underlying Markov decision process and then use it for planning. We have seen that in the tabular setting, computing and storing the entire model is computationally expensive. This led us to consider the family of model-free approaches, which learn the value function directly, and as such can be considered more economical in the amount of data that they store.

In chapter 12, we saw that using function approximation, we were able to scale model-free methods to very large state and action spaces. We will now explore similar ideas in the model-based framework. Namely, we will use function approximation to condense the representation of our model of the environment. More concretely, we will learn an approximate dynamics model $f \approx p$ and approximate rewards r.

There are a few ways in which the model-based approach is advantageous. First, if we have an accurate model of the environment, we can use it for planning — ideally also for interacting safely with the environment. However, in practice, we will rarely have such a model. In fact, the accuracy of our model will depend on the past experience of our agent and the region of the state-action space. Understanding the uncertainty in our model of the environment is crucial for planning. In particular, quantifying uncertainty is necessary to drive safe(!) exploration and avoid undesired states.

Moreover, modeling uncertainty in our model of the environment can be extremely useful in deciding where to explore. Learning a model can therefore help to dramatically reduce the sample complexity over model-free techniques. Often times this is crucial when developing agents for real-world use as in such settings, exploration is costly and potentially dangerous.

Algorithm 13.1 describes the general approach to model-based reinforcement learning.

Algorithm 13.1: Model-based reinforcement learning (outline)

```
start with an initial policy π and no (or some) initial data D
for several episodes do
    roll out policy π to collect data
    learn a model of the dynamics f and rewards r from data
    plan a new policy π based on the estimated models
```

We face three main challenges in model-based reinforcement learning. First, given a fixed model, we need to perform planning to decide on which actions to play. Second, we need to learn models f and r accurately and efficiently. Third, we need to effectively trade exploration and exploitation. We will discuss these three topics in the following.

13.1 Planning

There exists a large literature on planning in various settings. These settings can be mainly characterized as

- discrete or continuous action spaces;
- fully- or partially observable state spaces;
- constrained or unconstrained; and
- linear or non-linear dynamics.

In chapter 10, we have already seen algorithms such as policy iteration and value iteration, which can be used to solve planning exactly in tabular settings. In the following, we will now focus on the setting of continuous state and action spaces, fully observable state spaces, no constraints, and non-linear dynamics.

13.1.1 Deterministic Dynamics

To begin with, let us assume that our dynamics model is deterministic and known. That is, given a state-action pair, we know the subsequent state,

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{a}_t). \tag{13.1}$$

We continue to focus on the setting of infinite-horizon discounted returns (10.5), which we have been considering throughout our discussion of reinforcement learning. This yields the objective,

$$\max_{\boldsymbol{a}_{0:\infty}} \sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{x}_t, \boldsymbol{a}_t) \quad \text{such that } \boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{a}_t). \tag{13.2}$$

Now, because we are optimizing over an infinite time horizon, we cannot solve this optimization problem directly. This problem is studied ubiquitously in the area of *optimal control*. We will discuss one central idea from optimal control that is widely used in model-based reinforcement learning, and will later return to using this idea for learning parametric policies in section 13.1.3.

Planning over Finite Horizons The key idea of a classical algorithm from optimal control called *receding horizon control* (RHC) or *model predictive control* (MPC) is to iteratively plan over finite horizons. That is, in each round, we plan over a finite time horizon H and carry out the first action.

Algorithm 13.2: Model predictive control, MPC

1 **for** $t = 0$ **to** ∞ **do**
2 observe $\boldsymbol{x}_t$
3 plan over a finite horizon H,

$$\max_{\boldsymbol{a}_{t:t+H-1}} \sum_{\tau=t}^{t+H-1} \gamma^{\tau-t} r(\boldsymbol{x}_\tau, \boldsymbol{a}_\tau) \quad \text{such that } \boldsymbol{x}_{\tau+1} = \boldsymbol{f}(\boldsymbol{x}_\tau, \boldsymbol{a}_\tau) \tag{13.3}$$

4 carry out action $\boldsymbol{a}_t$

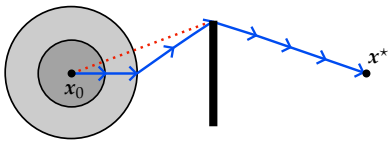


Figure 13.1: Illustration of model predictive control in a deterministic transition model. The agent starts in position $\boldsymbol{x}_0$ and wants to reach $\boldsymbol{x}^\star$ despite the black obstacle. We use the reward function $r(\boldsymbol{x}) = -\|\boldsymbol{x} - \boldsymbol{x}^\star\|$. The gray concentric circles represent the length of a single step. We plan with a time horizon of $H = 2$. Initially, the agent does not "see" the black obstacle, and therefore moves straight towards the goal. As soon as the agent sees the obstacle, the optimal trajectory is "replanned". The dotted red line corresponds to the optimal trajectory, the agent's steps are shown in blue.

Observe that the state $\boldsymbol{x}_\tau$ can be interpreted as a deterministic function $\boldsymbol{x}_\tau(\boldsymbol{a}_{t:\tau-1})$, which depends on all actions from time t to time $\tau - 1$ and the state $\boldsymbol{x}_t$. To solve the optimization problem of a single iteration, we therefore need to maximize,

$$J_H(\boldsymbol{a}_{t:t+H-1}) \doteq \sum_{\tau=t}^{t+H-1} \gamma^{\tau-t} r(\boldsymbol{x}_\tau(\boldsymbol{a}_{t:\tau-1}), \boldsymbol{a}_\tau). \tag{13.4}$$

This optimization problem is in general non-convex. If the actions are continuous and the dynamics and reward models are differentiable, we can nevertheless obtain analytic gradients of J_H. This can be done using the chain rule and "backpropagating" through time, analogously to backpropagation in neural networks.[1] Especially for large horizons H, this optimization problem becomes difficult to solve exactly due to local optima and vanishing/exploding gradients.

[1] see section 7.1.4

Tree Search Often, heuristic global optimization methods (also called "search methods") are used to optimize J_H. An example are *random shooting methods*, which find the optimal choice among a set of random proposals. Of course, obtaining a "good" set of randomly proposed action sequences is crucial. A naïve way of generating the proposals is to pick them uniformly at random. This strategy, however, usually does not perform very well as it corresponds to suggesting random walks of the state space. Alternatives are to sample from a Gaussian distribution or using the *cross-entropy method* which gradually adapts the sampling distribution by reweighing samples according to the rewards they produce.

Algorithm 13.3: Random shooting methods

1 generate m sets of random samples, $\boldsymbol{a}_{t:t+H-1}^{(i)}$

2 pick the sequence of actions $\boldsymbol{a}_{t:t+H-1}^{(i^\star)}$ where

$$i^\star = \arg\max_{i \in [m]} J_H(\boldsymbol{a}_{t:t+H-1}^{(i)}) \tag{13.5}$$

The evolution of the state can be visualized as a tree where — if dynamics are deterministic — the branching is fully determined by the played actions. For this reason, classical tree search algorithms can be employed, such as *alpha-beta pruning* or *Monte Carlo tree search*, which was used for example by "AlphaZero" (Silver et al., 2016, 2017) and can be viewed as an advanced variant of a shooting method.

Finite-Horizon Planning with Sparse Rewards A common problem of finite-horizon methods is that in the setting of sparse rewards, there is often no signal that can be followed. You can think of an agent that operates in a kitchen and tries to find a box of candy. Yet, to get to this box, it needs to perform a large number of actions. In particular, if this number of actions is larger than the horizon H, then the local optimization problem of MPC will not take into account the reward for choosing this long sequence of actions. Thus, the box of candy will likely never be found by our agent.

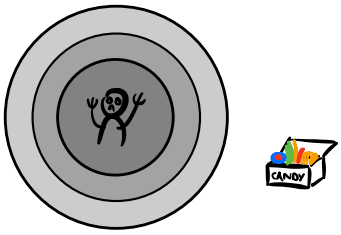


Figure 13.2: Illustration of finite-horizon planning with sparse rewards. When the finite time horizon does not suffice to "reach" a reward, the agent has no signal to follow.

A solution to this problem is to amend a long-term value estimate to the finite-horizon sum. The idea is to not only consider the rewards attained *while* following the actions $\boldsymbol{a}_{t:t_H-1}$, but to also consider the value of the final state $\boldsymbol{x}_{t+H}$, which estimates the discounted sum of

future rewards.

$$J_H(\boldsymbol{a}_{t:t+H-1}) \doteq \underbrace{\sum_{\tau=t}^{t+H-1} \gamma^{\tau-t} r(\boldsymbol{x}_\tau(\boldsymbol{a}_{t:\tau-1}), \boldsymbol{a}_\tau)}_{\text{short-term}} + \underbrace{\gamma^H V(\boldsymbol{x}_{t+H})}_{\text{long-term}}. \tag{13.6}$$

Intuitively, $\gamma^H V(\boldsymbol{x}_{t+H})$ is estimating the tail of the infinite sum.

Remark 13.4: Planning generalizes model-free methods!

Observe that for $H = 1$, when we use the value function estimate associated with this MPC controller, maximizing J_H coincides with using the greedy policy π_V. That is,

$$\boldsymbol{a}_t = \arg\max_{\boldsymbol{a} \in \mathcal{A}} J_1(\boldsymbol{a}) = \pi_V(\boldsymbol{x}_t). \tag{13.7}$$

Thus, by looking ahead for a single time step, we recover the approaches from the model-free setting in this model-based setting! Essentially, if we do not plan long-term and only consider the value estimate, the model-based setting reduces to the model-free setting. However, in the model-based setting, we are now able to use our model of the transition dynamics to anticipate the downstream effects of picking a particular action $\boldsymbol{a}_t$. This is one of the fundamental reasons for why model-based approaches are typically severely more sample efficient than model-free methods.

To obtain the value estimates, we can use the approaches we discussed in detail in section 11.4, such as TD-learning for on-policy value estimates and Q-learning for off-policy value estimates. For large state-action spaces, we can use their approximate variants, which we discussed in section 12.1 and section 12.2. To improve value estimates, we can obtain "artificial" data by rolling out policies within our model. This is a key advantage over model-free methods as, once a sufficiently accurate model has been learned, data for value estimation can be generated efficiently in simulation.

13.1.2 Stochastic Dynamics

How can we extend this approach to planning to a stochastic transition model? A natural extension of model predictive control is to do what is called *stochastic average approximation* (SAA) or *trajectory sampling* (Chua et al., 2018). Like in MPC, we still optimize over a deterministic sequence of actions, but now we will average over all resulting trajectories.

Algorithm 13.5: Trajectory sampling

for $t = 0$ **to** ∞ **do**

observe $\boldsymbol{x}_t$

optimize expected performance over a finite horizon H,

$$\max_{\boldsymbol{a}_{t:t+H-1}} \mathbb{E}_{\boldsymbol{x}_{t+1:t+H}}\left[\sum_{\tau=t}^{t+H-1} \gamma^{\tau-t} r_\tau + \gamma^H V(\boldsymbol{x}_{t+H}) \,\middle|\, \boldsymbol{a}_{t:t+H-1}, f\right] \tag{13.8}$$

carry out action $\boldsymbol{a}_t$

Intuitively, trajectory sampling optimizes over a much simpler object — namely a deterministic sequence of actions of length H — than finding a policy, which corresponds to finding an optimal decision tree mapping states to actions. Of course, using trajectory sampling (from an arbitrary starting state) implies such a policy. However, trajectory sampling never computes this policy explicitly, and rather, in each step, only plans over a finite horizon.

Figure 13.3: Illustration of trajectory sampling. High-reward states are shown in brighter colors. The agent iteratively plans a finite number of time steps into the future and picks the best initial action.

Computing the expectation exactly is typically not possible as this involves solving a high-dimensional integral of non-linear functions. Instead, a common approach is to use Monte Carlo estimates of this expectation. This approach is known as *Monte Carlo trajectory sampling*. The key issue with using sampling based estimation is that the sampled trajectory (i.e., sampled sequence of states) we obtain, depends on the actions we pick. In other words, the measure we average over depends on the decision variables — the actions. This is a problem that we have seen several times already! It naturally suggests using the reparameterization trick.[2]

[2] see theorem 5.31 and eq. (12.82)

Previously, we have used the reparameterization trick to reparameterize variational distributions (see section 5.5.1) and to reparameterize policies (see section 12.4.5). It turns out that we can use the exact same approach for reparameterizing the transition model. We say that a (stochastic) transition model f is *reparameterizable* iff $\boldsymbol{x}_{t+1} \sim f(\boldsymbol{x}_t, \boldsymbol{a}_t)$ is such that $\boldsymbol{x}_{t+1} = \boldsymbol{g}(\boldsymbol{\epsilon}; \boldsymbol{x}_t, \boldsymbol{a}_t)$, where $\boldsymbol{\epsilon} \sim \phi$ is a random variable that is independent of $\boldsymbol{x}_t$ and $\boldsymbol{a}_t$. We have already seen in example 12.18 (in the context of stochastic policies) how a Gaussian transition model can be reparameterized.

In this case, $\boldsymbol{x}_\tau$ is determined recursively by $\boldsymbol{a}_{t:\tau-1}$ and $\boldsymbol{\epsilon}_{t:\tau-1}$,

$$\begin{aligned} \boldsymbol{x}_\tau &= \boldsymbol{x}_\tau(\boldsymbol{\epsilon}_{t:\tau-1}; \boldsymbol{a}_{t:\tau-1}) \\ &\doteq \boldsymbol{g}(\boldsymbol{\epsilon}_{\tau-1}; \boldsymbol{g}(\dots; (\boldsymbol{\epsilon}_{t+1}; \boldsymbol{g}(\boldsymbol{\epsilon}_t; \boldsymbol{x}_t, \boldsymbol{a}_t), \boldsymbol{a}_{t+1}), \dots), \boldsymbol{a}_{\tau-1}). \end{aligned} \tag{13.9}$$

This allows us to obtain unbiased estimates of J_H using Monte Carlo approximation,

$$J_H(\boldsymbol{a}_{t:t+H-1}) \approx \frac{1}{m}\sum_{i=1}^{m}\sum_{\tau=t}^{t+H-1}\Big(\gamma^{\tau-t} r(\boldsymbol{x}_\tau(\boldsymbol{\epsilon}^{(i)}_{t:\tau-1}; \boldsymbol{a}_{t:\tau-1}), \boldsymbol{a}_\tau) + \gamma^H V(\boldsymbol{x}_{t+H})\Big) \tag{13.10}$$

where $\boldsymbol{\epsilon}^{(i)}_{t:t+H-1} \overset{\text{iid}}{\sim} \phi$ are independent samples. To optimize this approximation we can again compute analytic gradients or use shooting methods as we have discussed in section 13.1.1 for deterministic dynamics.

13.1.3 *Parametric Policies*

When using algorithms such as model predictive control for planning, planning needs to be done online before each time we take an action. This is called *closed-loop control* and can be expensive. Especially when the time horizon is large, or we encounter similar states many times (leading to "repeated optimization problems"), it can be beneficial to "store" the planning decision in a (deterministic) policy,

$$\boldsymbol{a}_t = \boldsymbol{\pi}(\boldsymbol{x}_t; \boldsymbol{\varphi}) \doteq \boldsymbol{\pi}_{\boldsymbol{\varphi}}(\boldsymbol{x}_t). \tag{13.11}$$

This policy can then be trained offline and evaluated cheaply online, which is known as *open-loop control*.

This is akin to a problem that we have discussed in detail in the previous chapter when extending Q-learning to large action spaces. There, this led us to discuss policy gradient and actor-critic methods. Recall that in Q-learning, we seek to follow the greedy policy,

$$\boldsymbol{\pi}^\star(\boldsymbol{x}) = \arg\max_{\boldsymbol{a}\in\mathcal{A}} Q^\star(\boldsymbol{x}, \boldsymbol{a}; \boldsymbol{\theta}), \qquad \text{see eq. (12.20)}$$

and therefore had to solve an optimization problem over all actions. We accelerated this by learning an approximate policy that "mimicked" this optimization,

$$\boldsymbol{\varphi}^\star = \arg\max_{\boldsymbol{\varphi}} \underbrace{\mathbb{E}_{\boldsymbol{x}\sim\mu}\big[Q^\star(\boldsymbol{x}, \boldsymbol{\pi}_{\boldsymbol{\varphi}}(\boldsymbol{x}); \boldsymbol{\theta})\big]}_{J_\mu(\boldsymbol{\varphi};\boldsymbol{\theta})} \qquad \text{see eq. (12.73)}$$

where $\mu(\boldsymbol{x}) > 0$ was some exploration distribution that has full support and thus leads to the exploration of all states. The key idea was that if we use a differentiable approximation Q and a differentiable parameterization of policies, which is "rich enough", then both optimizations are equivalent, and we can use the chain rule to obtain gradient estimates of the second expression. We then used this to derive the *deep deterministic policy gradients* (DDPG) and *stochastic value*

gradients (SVG) algorithms. It turns out that there is a very natural analogue to DDPG/SVG for model-based reinforcement learning.

Instead of maximizing the Q-function directly, we use finite-horizon planning to estimate the immediate value of the policy within the next H time steps and simply use the Q-function to approximate the terminal value (i.e., the tails of the infinite sum). Then, our objective becomes,

$$J_{\mu,H}(\boldsymbol{\varphi};\boldsymbol{\theta}) \doteq \mathbb{E}_{\boldsymbol{x}_0\sim\mu,\boldsymbol{x}_{1:H}|\pi_{\boldsymbol{\varphi}},f}\left[\sum_{\tau=0}^{H-1}\gamma^\tau r_\tau + \gamma^H Q^\star(\boldsymbol{x}_H, \pi_{\boldsymbol{\varphi}}(\boldsymbol{x}_H);\boldsymbol{\theta})\right] \tag{13.12}$$

This approach naturally extends to randomized policies using reparameterization gradients, which we have discussed in section 12.4.5. Analogously to remark 13.4, for $H = 0$, this coincides with the DDPG objective! For larger time horizons, the look-ahead takes into account the transition model for planning next time steps. This tends to help dramatically in improving policies *much more rapidly* between episodes. Instead of just gradually improving policies a little by slightly adapting the policy to the learned value function estimates (as in model-free RL), we use the model to anticipate the consequences of actions multiple time steps ahead. This is at the heart of model-based reinforcement learning.

Essentially, we are using methods such as Q-learning and DDPG/SVG as subroutines within the framework of model predictive control to do much bigger steps in policy improvement than to slightly improve the next picked action.

Remark 13.6: Planning as inference

We have been using *Monte Carlo rollouts* to estimate the expectation of eq. (13.12). That is, we have been using a Monte Carlo approximation (e.g., a sample mean) using samples obtained by "rolling out" the induced Markov chain of a fixed policy.

It would certainly be preferable to compute the expectation exactly, however, this is generally not possible as this involves solving a high dimensional integral. Recall that we faced the same problem when studying inference in the first half of the course. In both problems, we need to approximate high-dimensional integrals (i.e., expectations). This suggests a deep connection between the problems of planning and inference. It is therefore not surprising that many techniques for approximate inference that we have seen earlier can also be applied to planning.

- *Monte Carlo approximation* which we have been focusing on during our discussion of planning is a very simple inference algorithm — approximating an expectation by sampling from the distribution that is averaged over. This allowed us to obtain unbiased gradient estimates (which may have high variance).
- An alternative approach is *moment matching* (cf. section 5.4.6). Instead of approximating the expectation, here, we approximate the distribution over trajectories using a tractable distribution (e.g., a Gaussian) and "matching" their moments. This then allows us to analytically compute gradients of the expectation in eq. (13.12). A prominent example of this approach is *probabilistic inference for learning control* (PILCO).
- In section 12.5, we have used *variational inference* for planning, and seen that it coincides with entropy regularization as implemented by the *soft actor critic* (SAC) algorithm.

13.2 Learning

Thus far, we have considered known environments. That is, we assumed that the transition model f and the rewards r are known. In reinforcement learning, f and r are (of course!) not known. Instead, we have to estimate them from data. This will also be crucial in our later discussion of exploration in section 13.3 where we explore methods of driving data collection to learn what we need to learn about the world.

First, let us revisit one of our key observations when we first introduced the reinforcement learning problem. Namely, that the observed transitions $\boldsymbol{x}'$ and rewards r are conditionally independent given the state-action pairs $(\boldsymbol{x}, \boldsymbol{a})$.[3] This is due to the Markovian structure of the underlying Markov decision process.

[3] see eq. (11.3)

This is the key observation that allows us to treat the estimation of the dynamics and rewards as a simple regression problem (or a density estimation problem when the quantities are stochastic rather than deterministic). More concretely, we can estimate the dynamics and rewards off-policy using the standard supervised learning techniques we discussed in earlier chapters, from a replay buffer

$$\mathcal{D} = \{(\underbrace{\boldsymbol{x}_t, \boldsymbol{a}_t}_{\text{"input"}}, \underbrace{r_t, \boldsymbol{x}_{t+1}}_{\text{"label"}})\}_t. \tag{13.13}$$

Here, $\boldsymbol{x}_t$ and $\boldsymbol{a}_t$ are the "inputs", and r_t and $\boldsymbol{x}_{t+1}$ are the "labels" of the regression problem. Due to the conditional independence of the labels given the inputs, we have independent label noise (i.e., "independent

training data") which is the basic assumption that we have been making throughout our discussion of techniques for probabilistic machine learning in part I.

The key difference to supervised learning is that the set of inputs depends on how we act. That is, the current inputs arise from previous policies, and the inputs which we will observe in the future will depend on the model (and policy) obtained from the current data: we have feedback loops! We will come back to this aspect of reinforcement learning in the next section on exploration. For now, recall we only assume that we have used an arbitrary policy to collect some data, which we then stored in a replay buffer, and which we now want to use to learn the "best-possible" model of our environment.

13.2.1 Bayesian Learning

In the following, we will discuss how we can use the techniques from Bayesian learning, which we have seen in part I, to learn the dynamics and reward models. Thereby, we will focus on learning the transition model f as learning the reward model r is completely analogous. For learning deterministic dynamics or rewards, we can use for example Gaussian processes (cf. chapter 4) or deep neural networks (cf. chapter 7). We will now focus on the setting where the dynamics are stochastic, that is,

$$\boldsymbol{x}_{t+1} \sim f(\boldsymbol{x}_t, \boldsymbol{a}_t; \boldsymbol{\psi}). \tag{13.14}$$

Example 13.7: Conditional Gaussian dynamics

We could, for example, use a conditional Gaussian for the transition model,

$$\boldsymbol{x}_{t+1} \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}_t, \boldsymbol{a}_t; \boldsymbol{\psi}), \boldsymbol{\Sigma}(\boldsymbol{x}_t, \boldsymbol{a}_t; \boldsymbol{\psi})). \tag{13.15}$$

As we have seen in eq. (1.134), we can rewrite the covariance matrix $\boldsymbol{\Sigma}$ as a product of a lower-triangular matrix $\mathcal{L}$ and its transpose using the Cholesky decomposition $\boldsymbol{\Sigma} = \mathcal{L}\mathcal{L}^\top$ of $\boldsymbol{\Sigma}$. This allows us to represent the model by only $n(n+1)/2$ parameters. Moreover, we have learned that Gaussians are reparameterizable, which we have seen to be useful for planning.

Note that this model reduces to a deterministic model if the covariance is zero. So the stochastic transition models encompass all deterministic models. Moreover, in many applications (such as robotics), it is often useful to use stochastic models to attribute slight inaccuracies and measurement noise to a small uncertainty in the model.

A first approach might be to obtain a point estimate for f, either through maximum likelihood estimation (which we have seen to overfit easily) or through maximum a posteriori estimation. If, for example, f is represented as a deep neural network, we have already seen how to find the MAP estimate of its weights in section 7.2.1.

Remark 13.8: The key pitfall of point estimates

However, using point estimates leads to a *key pitfall* of model-based reinforcement learning. Using a point estimate of the model for planning — even if this point estimate is very accurate — often performs *very poorly*. The reason is that planning is very good at overfitting (i.e., exploiting) small errors in the transition model. Moreover, the errors in the model estimate compound over time when using a longer time horizon H. The key to remedy this pitfall lies in being robust to misestimated models. This naturally suggests quantifying the uncertainty in our model estimate and taking it into account during planning.[4] In the following section, we will rediscover that estimates of epistemic uncertainty are also extremely useful for driving (safe) exploration — something that we have already encountered in our discussion of Bayesian optimization.

[4] Quantifying the uncertainty of an estimate is a problem that we have spent the first few chapters exploring. Notably, refer to

- section 2.3 for a description of epistemic and aleatoric uncertainty;
- chapter 4 for our use of uncertainty estimates in the context of Gaussian processes;
- chapter 7 for our use of uncertainty estimates in the context of Bayesian deep learning; and
- chapters 8 and 9 for our use of epistemic uncertainty estimates to drive exploration.

In the following, we will differentiate between the epistemic uncertainty and the aleatoric uncertainty. Recall from section 2.3 that epistemic uncertainty corresponds to our uncertainty about the model, $p(f \mid \mathcal{D})$, while aleatoric uncertainty corresponds to the uncertainty of the transitions in the underlying Markov decision process (which can be thought of as "irreducible" noise), $p(\boldsymbol{x}_{t+1} \mid f, \boldsymbol{x}_t, \boldsymbol{a}_t)$.

Intuitively, Bayesian learning of dynamics models corresponds to learning a distribution over possible models f and r given prior *beliefs*, where f and r characterize the underlying Markov decision process. This goes to show another benefit of the model-based over the model-free approach to reinforcement learning. Namely, that it is much easier to encode prior knowledge (as a Bayesian prior) about the transition and rewards model.

Example 13.9: Inference with conditional Gaussian dynamics

Let us revisit inference with our conditional Gaussian dynamics model from eq. (13.15),

$$\boldsymbol{x}_{t+1} \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}_t, \boldsymbol{a}_t; \boldsymbol{\psi}), \boldsymbol{\Sigma}(\boldsymbol{x}_t, \boldsymbol{a}_t; \boldsymbol{\psi})).$$

Recall that in the setting of Bayesian neural networks, most ap-

proximate inference techniques represented the approximate posterior using some form of a mixture of Gaussians,[5]

$$p(\boldsymbol{x}_{t+1} \mid \mathcal{D}, \boldsymbol{x}_t, \boldsymbol{a}_t) \approx \frac{1}{m} \sum_{i=1}^{m} \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}_t, \boldsymbol{a}_t; \boldsymbol{\psi}^{(i)}), \boldsymbol{\Sigma}(\boldsymbol{x}_t, \boldsymbol{a}_t; \boldsymbol{\psi}^{(i)})). \tag{13.16}$$

Hereby, the epistemic uncertainty is represented by the variance between mixture components, and the aleatoric uncertainty by the average variance within the components.[6]

[5] see
- eq. (7.18) for variational inference;
- eq. (7.21a) for Marko chain Monte Carlo;
- eq. (7.28) for dropout regularization; and
- eq. (7.29) for probabilistic ensembles.

[6] see section 7.3.1

In supervised learning, we often conflated the notions of epistemic and aleatoric uncertainty. In the context of planning, there is an important consequence of the decomposition into epistemic and aleatoric uncertainty. Recall that the epistemic uncertainty corresponds to a distribution over Markov decision processes f, whereas the aleatoric uncertainty corresponds to the randomness in the transitions within one such MDP f. Crucially, this randomness in the transitions must be consistent within a single MDP! That is, once we selected a single MDP for planning, we should disregard the epistemic uncertainty and solely focus on the randomness of the transitions. Then, to take into account epistemic uncertainty, we should average our plan across the different realizations of f. This yields the following Monte Carlo estimate of our reward J_H,

$$J_H(\boldsymbol{a}_{t:t+H-1}) \approx \frac{1}{m} \sum_{i=1}^{m} J_H(\boldsymbol{a}_{t:t+H-1}; f^{(i)}) \qquad \text{where} \tag{13.17}$$

$$J_H(\boldsymbol{a}_{t:t+H-1}; f) \doteq \sum_{\tau=t}^{t+H-1} \gamma^{\tau-t} r(\boldsymbol{x}_\tau(\boldsymbol{\epsilon}^{(i)}_{t:\tau-1}; \boldsymbol{a}_{t:\tau-1}, f), \boldsymbol{a}_\tau) + \gamma^H V(\boldsymbol{x}_{t+H}). \tag{13.18}$$

Here, $f^{(i)} \overset{\text{iid}}{\sim} p(f \mid \mathcal{D})$ are independent samples of the transition model, and $\boldsymbol{\epsilon}^{(i)}_{t:t+H-1} \overset{\text{iid}}{\sim} \phi$ parameterizes the dynamics analogously to eq. (13.9):

$$\begin{aligned} &\boldsymbol{x}_\tau(\boldsymbol{\epsilon}_{t:\tau-1}; \boldsymbol{a}_{t:\tau-1}, f) \\ &\doteq f(\boldsymbol{\epsilon}_{\tau-1}; f(\dots; (\boldsymbol{\epsilon}_{t+1}; f(\boldsymbol{\epsilon}_t; \boldsymbol{x}_t, \boldsymbol{a}_t), \boldsymbol{a}_{t+1}), \dots), \boldsymbol{a}_{\tau-1}). \end{aligned} \tag{13.19}$$

Observe that the epistemic and aleatoric uncertainty are treated differently. Within a particular MDP f, we ensure that randomness (i.e., aleatoric uncertainty) is simulated consistently using our previous framework from our discussion of planning.[7] The Monte Carlo samples of f take into account the epistemic uncertainty about the transition model. In our previous discussion of planning, we assumed the Markov decision process f to be fixed. Essentially, in eq. (13.17) we are now using

[7] see eq. (13.10)

Monte Carlo trajectory sampling as a subroutine and average over an "ensemble" of Markov decision processes.

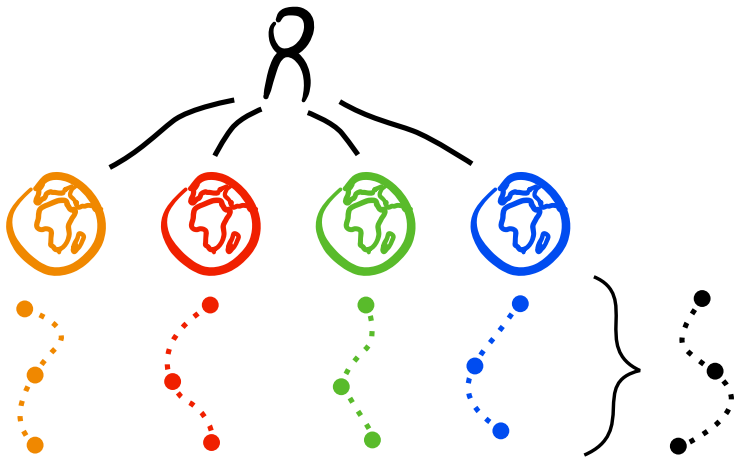


Figure 13.4: Illustration of planning with epistemic uncertainty and Monte Carlo sampling. The agent considers m alternative "worlds". Within each world, it plans a sequence of actions over a finite time horizon. Then, the agent averages all optimal initial actions from all worlds. Crucially, each world by itself is *consistent*. That is, its transition model (i.e., the aleatoric uncertainty of the model) is constant.

The same approach that we have seen in section 13.1.3 can be used to "compile" these plans into a parametric policy that can be trained offline, in which case, we write $J_H(\pi)$ instead of $J_H(\boldsymbol{a}_{t:t+H-1})$.

This leads us to a first greedily exploitative algorithm for model-based reinforcement learning, which is shown in alg. 13.10. This algorithm is purely exploitative as it greedily maximizes the expected reward with respect to the transition model, taking into account epistemic uncertainty.

Algorithm 13.10: Greedy exploitation for model-based RL

start with (possibly empty) data $\mathcal{D} = \emptyset$ and a prior $p(f) = p(f \mid \mathcal{D})$
for several episodes **do**
  plan a new policy π to (approximately) maximize,

$$\max_{\pi} \mathbb{E}_{f \sim p(\cdot \mid \mathcal{D})}[J_H(\pi; f)] \tag{13.20}$$

  roll out policy π to collect more data
  update posterior $p(f \mid \mathcal{D})$

In the context of Gaussian process models, this algorithm is called *probabilistic inference for learning control* (PILCO) (Deisenroth and Rasmussen, 2011), which was the first demonstration of how sample efficient model-based reinforcement learning can be. As was mentioned in remark 13.6, the originally proposed PILCO uses moment matching instead of Monte Carlo averaging.

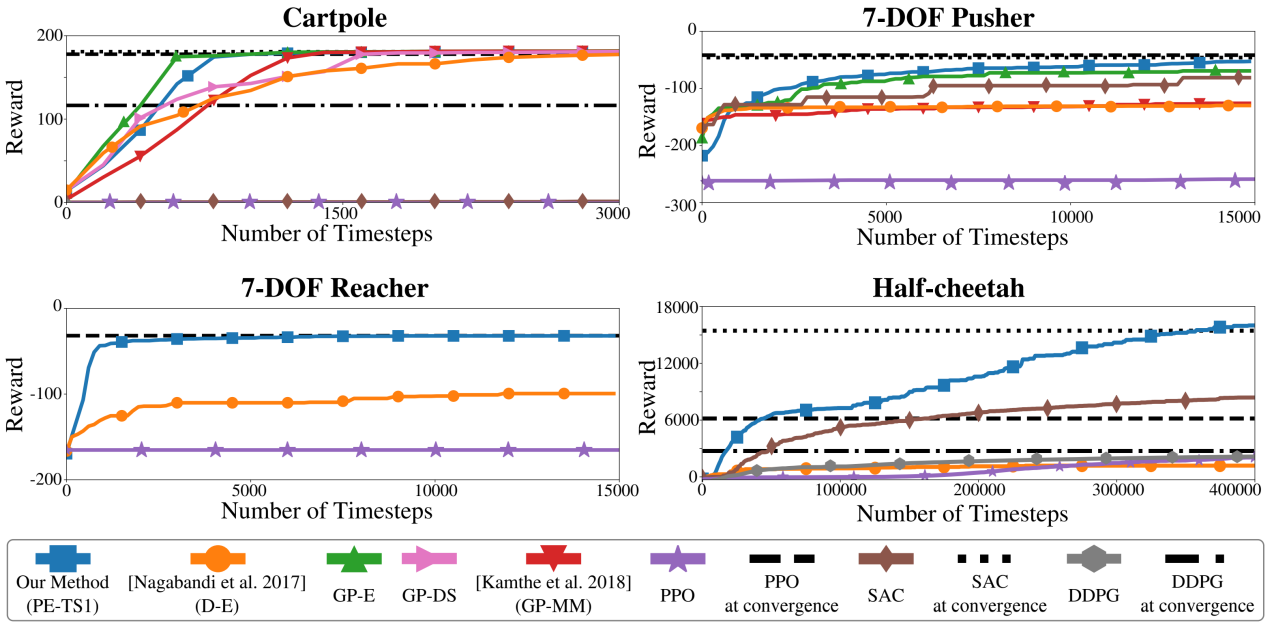


Figure 13.5: Sample efficiency of model-free and model-based reinforcement learning. DDPG and SAC are shown as constant (black) lines, because they take an order of magnitude more time steps before learning a good model. They also compare the PETS algorithm (blue) to deterministic ensembles (orange), where the transition models are assumed to be deterministic (or to have covariance 0). Taken from "Deep reinforcement learning in a handful of trials using probabilistic dynamics models" (Chua et al., 2018).

In the context of neural networks, this algorithm is called *probabilistic ensembles with trajectory sampling* (PETS) (Chua et al., 2018), which is one of the state-of-the-art algorithms. PETS uses an ensemble of conditional Gaussian distributions over weights, trajectory sampling for evaluating the performance and model predictive control for planning.

Notably, PETS does not explicitly explore. Exploration only happens due to the uncertainty in the model, which already drives exploration to some extent. In many settings, however, this incentive is not sufficient for exploration — especially when rewards are sparse.

Optional Readings

- Deisenroth and Rasmussen (2011).
  PILCO: A model-based and data-efficient approach to policy search.
- Chua, Calandra, McAllister, and Levine (2018).
  Deep reinforcement learning in a handful of trials using probabilistic dynamics models.

13.3 *Exploration*

Exploration is critical when interacting with unknown environments, such as in reinforcement learning. We first encountered the exploration-exploitation dilemma in our discussion of Bayesian optimization, where

we aimed at maximizing an unknown function in as little time as possible by making noisy observations.[8] Within the framework of Bayesian optimization, we used so-called acquisition functions for selecting the next point at which to observe the unknown function. Observe that these acquisition functions are analogous to policies in the setting of reinforcement learning. In particular, the policy that uses greedy exploitation like we have seen in the previous section is analogous to simply picking the point that maximizes the mean of the posterior distribution. In the context of Bayesian optimization, we have already seen that this is insufficient for exploration and can easily get stuck in locally optimal solutions. As reinforcement learning is a strict generalization of Bayesian optimization, there is no reason why such a strategy should be sufficient now.

[8] see chapter 9

Recall from our discussion of Bayesian optimization that we "solved" this problem by using the epistemic uncertainty in our model of the unknown function to pick the next point to explore. This is what we will now explore in the context of reinforcement learning.

One simple strategy that we already investigated is the addition of some *exploration noise*. In other words, one follows a greedy exploitative strategy, but every once in a while, one chooses a random action (like in ϵ-greedy); or one adds additional noise to the selected actions (known as Gaussian noise "dithering"). We have already seen that in difficult exploration tasks, these strategies are not systematic enough.

Two other approaches that we have considered before are Thompson sampling (cf. section 9.3.3) and, more generally, the principle of *optimism in the face of uncertainty* (cf. section 9.2.1).

13.3.1 Thompson Sampling

Recall from section 9.3.3 the main idea behind Thompson sampling: namely, that the randomness in the realizations of f from the posterior distribution is already enough to drive exploration. That is, instead of picking the action that performs best on average across all realizations of f, Thompson sampling picks the action that performs best for a *single realization* of f. The epistemic uncertainty in the realizations of f leads to variance in the picked actions and provides an additional incentive for exploration. This yields alg. 13.11 which is an immediate adaptation of greedy exploitation and straightforward to implement.

13.3.2 Optimistic Exploration

We have already seen in the context of Bayesian optimization and tabular reinforcement learning that optimism is a central pillar for explo-

Algorithm 13.11: Thompson sampling

start with (possibly empty) data $\mathcal{D} = \emptyset$ and a prior $p(f) = p(f \mid \mathcal{D})$
for several episodes **do**
  sample a model $f \sim p(\cdot \mid \mathcal{D})$
  plan a new policy π to (approximately) maximize,

$$\max_{\pi} J_H(\pi; f) \tag{13.21}$$

  roll out policy π to collect more data
  update posterior $p(f \mid \mathcal{D})$

ration. But how can we explore optimistically in model-based reinforcement learning?

Let us consider a set $\mathcal{M}(\mathcal{D})$ of *plausible models* given some data $\mathcal{D}$. Optimistic exploration would then optimize for the most advantageous model among all models that are plausible given the seen data.

Example 13.12: Plausible conditional Gaussians

In the context of conditional Gaussians, we can consider the set of all models such that the prediction of a single dimension i lies in some confidence interval,

$$\begin{aligned} f_i(\boldsymbol{x}, \boldsymbol{a}) \in \mathcal{C}_i \doteq [&\mu_i(\boldsymbol{x}, \boldsymbol{a} \mid \mathcal{D}) - \beta_i \sigma_i(\boldsymbol{x}, \boldsymbol{a} \mid \mathcal{D}), \\ &\mu_i(\boldsymbol{x}, \boldsymbol{a} \mid \mathcal{D}) + \beta_i \sigma_i(\boldsymbol{x}, \boldsymbol{a} \mid \mathcal{D})], \end{aligned} \tag{13.22}$$

where β_i tunes the width of the confidence interval, analogously to section 9.3.1. The set of plausible models is then given as

$$\mathcal{M}(\mathcal{D}) \doteq \{f \mid \forall \boldsymbol{x} \in \mathcal{X}, \boldsymbol{a} \in \mathcal{A}, i \in [d] : f_i(\boldsymbol{x}, \boldsymbol{a}) \in \mathcal{C}_i\}. \tag{13.23}$$

When compared to greedy exploitation, instead of taking the optimal step on average with respect to all realizations of the transition model f, optimistic exploration as shown in alg. 13.13 takes the optimal step with respect to the most optimistic model among all transition models that are consistent with the data.

In general, the joint maximization over π and f is very difficult to solve computationally. Yet, remarkably, it turns out that this complex optimization problem can be reduced to standard model-based reinforcement learning with a fixed model. The key idea is to consider an agent that can control its "luck". In other words, we assume that the agent believes it can control the outcome of its actions — or rather

Algorithm 13.13: Optimistic exploration

start with (possibly empty) data $\mathcal{D} = \emptyset$ and a prior $p(f) = p(f \mid \mathcal{D})$
for several episodes **do**
  plan a new policy π to (approximately) maximize,

$$\max_{\pi} \max_{f \in \mathcal{M}(\mathcal{D})} J_H(\pi; f) \tag{13.24}$$

  roll out policy π to collect more data
  update posterior $p(f \mid \mathcal{D})$

choose which of the plausible dynamics it follows. The "luck" of the agent can be considered as additional decision variables. Consider the optimization problem,

$$\pi_{t+1} \doteq \arg\max_{\pi} \max_{\boldsymbol{\eta}(\cdot) \in [-1,1]^d} J_H(\pi; \widehat{f}_t) \tag{13.25}$$

with "optimistic" dynamics

$$\widehat{f}_{t,i}(\boldsymbol{x}, \boldsymbol{a}) \doteq \mu_{t,i}(\boldsymbol{x}, \boldsymbol{a}) + \beta_{t,i} \eta_i(\boldsymbol{x}, \boldsymbol{a}) \sigma_{t,i}(\boldsymbol{x}, \boldsymbol{a}). \tag{13.26}$$

Here the decision variables η_i control the variance of an action. That is, within the confidence bounds of the transition model, the agent can freely choose the state that is reached by playing an action $\boldsymbol{a}$ from state $\boldsymbol{x}$. Essentially, this corresponds to maximizing expected reward in an augmented (optimistic) MDP with *known* dynamics $\widehat{f}$ and with a larger action space that also includes the decision variables $\boldsymbol{\eta}$. This is a known MDP for which we can use our toolbox for planning which we developed in section 13.1.

The algorithm that maximizes expected reward in this augmented MDP is called *hallucinated upper confidence reinforcement learning* (H-UCRL) (Curi et al., 2020; Treven et al., 2023). H-UCRL can be seen as the natural extension of the UCB acquisition function from Bayesian optimization to reinforcement learning. An illustration of the algorithm is given in fig. 13.6.

Intuitively, the agent has the tendency to believe that it can achieve much more than it actually can. As more data is collected, the confidence bounds shrink and the optimistic policy rewards converge to the actual rewards. Yet, crucially, we only collect data in regions of the state-action space that are more promising than the regions that we have already explored. That is, we only collect data in the most promising regions.

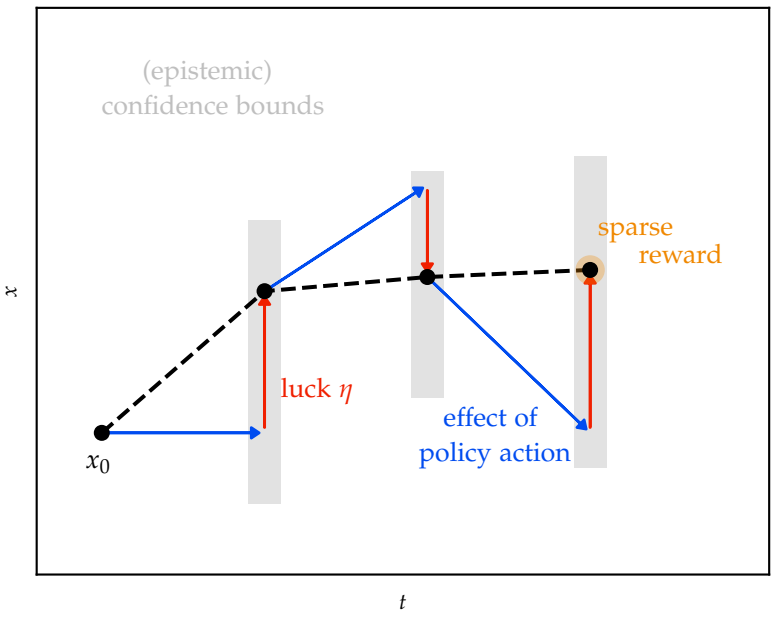


Figure 13.6: Illustration of H-UCRL in a one-dimensional state space. The agent "hallucinates" that it takes the black trajectory when, in reality, the outcomes of its actions are as shown in blue. The agent can hallucinate to land anywhere within the gray confidence regions (i.e., the epistemic uncertainty in the model) using the luck decision variables η. This allows agents to discover long sequences of actions leading to sparse rewards.

Exercise 13.14: Bounding the regret of H-UCRL

Analogously to our analysis of the UCB acquisition function for Bayesian optimization, we can use optimism to bound the regret of H-UCRL. We assume for simplicity that $d = 1$ and drop the index i in the following. Let us denote by $x_{t,k}$ the k-th state visited during episode t when following policy π_t. We denote by $\widehat{x}_{t,k}$ the corresponding state but under the optimistic dynamics $\widehat{f}$ rather than the true dynamics f. The instantaneous regret during episode t is given by

$$r_t = J_H(\pi^\star; f) - J_H(\pi_t; f) \tag{13.27}$$

where we take the objective to be $J_H(\pi; f) = \sum_{k=0}^{H-1} r(x_{t,k}, \pi(x_{t,k}))$.

You may assume that $f(x, \pi(x))$, $r(x, \pi(x))$, and $\sigma(x, \pi(x))$ are Lipschitz continuous in x.

1. Show by induction that for any $k \geq 0$, with high probability,

$$\left\| \widehat{x}_{t,k} - x_{t,k} \right\| \leq 2\beta_t \sum_{l=0}^{k-1} \alpha_t^{k-1-l} \sigma_{t-1}(x_{t,l}, \pi_t(x_{t,l})) \tag{13.28}$$

   where α_t depends on the Lipschitz constants and β_t.
2. Assuming w.l.o.g. that $\alpha_t \geq 1$, show that with high probability,

$$r_t \leq \mathcal{O}\left(\beta_t H \alpha_t^{H-1} \sum_{k=0}^{H-1} \sigma_{t-1}(x_{t,k}, \pi_t(x_{t,k})) \right) \tag{13.29}$$

3. Let $\Gamma_T \doteq \max_{\pi_1,\dots\pi_T} \sum_{t=1}^{T} \sum_{k=0}^{H-1} \sigma_{t-1}^2(x_{t,k}, \pi_t(x_{t,k}))$. Analogously to exercise 9.7, it can be derived that $\Gamma_T \leq \mathcal{O}(H\gamma_T)$ if the dynamics are modeled by a Gaussian process.[9] Bound the cumulative regret $R_T = \sum_{t=1}^{T} r_t \leq \mathcal{O}\left(\beta_T H^{\frac{3}{2}} \alpha_T^{H-1} \sqrt{T\Gamma_T}\right)$.

Thus, if the dynamics are modeled by a Gaussian process with kernel such that γ_T is sublinear, the regret of H-UCRL is sublinear.

▷ *Solution*

[9] For details, see appendix A of Treven et al. (2023).

Optimistic exploration yields the strongest effects for hard exploration tasks, for example, settings with large penalties associated with performing certain actions and settings with sparse rewards.[10] In those settings, most other strategies (i.e., those that are not sufficiently explorative), learn not to act at all. However, even in settings of "ordinary rewards", optimistic exploration often learns good policies faster.

[10] Action penalties are often used to discourage the agent from exhibiting unwanted behavior. However, increasing the action penalty increases the difficulty of exploration. Therefore, optimistic exploration is especially useful in settings where we want to practically disallow many actions by attributing large penalties to them.

13.3.3 *Constrained Exploration*

Besides making exploration more efficient, another use of uncertainty is to make exploration more safe. Today, reinforcement learning is still far away from being deployed directly to the real world. In practice, reinforcement learning is almost always used together with a simulator, in which the agent can "safely" train and explore. Yet, in many domains, it is not possible to simulate the training process, either because we are lacking a perfect model of the environment, or because simulating such a model is too computationally inefficient. This is where we can make use of uncertainty estimates of our model to avoid unsafe states.

Figure 13.7: Illustration of planning with confidence bounds. The unsafe set of states is shown as the red region. The agent starts at the position denoted by the black dot and plans a sequence of actions. The confidence bounds on the subsets of states that are reached by this action sequence are shown in gray.

Let us denote by $\mathcal{X}_{\text{unsafe}}$ the subset of unsafe states of our state space $\mathcal{X}$. A natural idea is to perform planning using confidence bounds of our epistemic uncertainty. This allows us to pessimistically forecast the plausible consequences of our actions, given what we have already learned about the transition model. As we collect more data, the confidence bounds will shrink, requiring us to be less conservative over time. This idea is also at the heart of fields like *robust control*.

A general formalism for planning under constraints is the notion of *constrained Markov decision processes* (Altman, 1999). Given dynamics f, planning in constrained MDPs amounts to the following optimization problem:

$$\max_{\pi} \quad J_\mu(\pi; f) \doteq \mathbb{E}_{\boldsymbol{x}_0 \sim \mu, \boldsymbol{x}_{1:\infty} | \pi, f} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \tag{13.30a}$$

$$\text{subject to} \quad J_\mu^c(\pi; f) \leq \delta \tag{13.30b}$$

where μ is a distribution over the initial state and

$$J_\mu^c(\pi; f) \doteq \mathbb{E}_{\boldsymbol{x}_0 \sim \mu, \boldsymbol{x}_{1:\infty} | \pi, f} \left[\sum_{t=0}^{\infty} \gamma^t c(\boldsymbol{x}_t) \right] \tag{13.31}$$

are expected discounted costs with respect to a cost function $c : \mathcal{X} \to \mathbb{R}_{\geq 0}$.[11] Observe that for the cost function $c(\boldsymbol{x}) \doteq \mathbb{1}\{\boldsymbol{x} \in \mathcal{X}_{\text{unsafe}}\}$, the value $J_\mu^c(\pi; f)$ can be interpreted as an upper bound on the (discounted) probability of visiting unsafe states under dynamics f,[12] and hence, the constraint $J_\mu^c(\pi; f) \leq \delta$ bounds the probability of visiting an unsafe state when following policy π.

[11] It is straightforward to extend this framework to allow for multiple constraints.

[12] This follows from a simple union bound (1.1).

The *augmented Lagrangian method* can be used to solve the optimization problem of eq. (13.30).[13] Thereby, one solves

[13] For an introduction, read chapter 17 of "Numerical optimization" (Wright et al., 1999).

$$\max_\pi \min_{\lambda \geq 0} \quad J_\mu(\pi; f) - \lambda (J_\mu^c(\pi; f) - \delta) \tag{13.32}$$

$$= \max_\pi \begin{cases} J_\mu(\pi; f) & \text{if } \pi \text{ is feasible} \\ -\infty & \text{otherwise.} \end{cases} \tag{13.33}$$

Observe that if π is feasible, then $J_\mu^c(\pi; f) \leq \delta$ and so the minimum over λ is satisfied if $\lambda = 0$. Conversely, if π is infeasible, λ can be made arbitrarily large to solve the optimization problem. In practice, an additional penalty term is added to smooth the objective when transitioning between feasible and infeasible policies.

Note that solving constrained optimization problems such as eq. (13.30) yields an optimal safe policy. However, it is not ensured that constraints are not violated during the search for the optimal policy. Generally, exterior penalty methods such as the augmented Lagrangian method allow for generating infeasible points during the search, and are therefore not suitable when constraints have to be strictly enforced at all times. Thus, this method is more applicable in the episodic setting (e.g. when an agent is first "trained" in a simulated environment and then "deployed" to the actual task) rather than in the continuous setting where the agent has to operate in the "real world" from the beginning and cannot easily be reset.

Remark 13.15: Barrier functions

The augmented Lagrangian method is merely one example of optimizing a joint objective of maximizing rewards and minimizing costs to find a policy with bounded costs. Another example of this approach are so-called *Barrier functions* which augment the reward objective with a smoothed approximation of the boundary

of a set $\mathcal{X}_{\text{unsafe}}$. That is, one solves

$$\max_{\pi} J_\mu(\pi; f) - \lambda \cdot B^c_\mu(\pi; f) \tag{13.34}$$

for some $\lambda > 0$ where the barrier function $B^c_\mu(\pi; f)$ goes to infinity as a state on the boundary of $\mathcal{X}_{\text{unsafe}}$ is approached. Examples for barrier functions are $-\log c(\boldsymbol{x})$ and $\frac{1}{c(\boldsymbol{x})}$.

Barrier functions are an interior penalty method which ensure that points are feasible during the search for the optimal solution.

So far, we have assumed knowledge of the true dynamics to solve the optimization problem of eq. (13.30). If we do not know the true dynamics but instead have access to a set of plausible models $\mathcal{M}(\mathcal{D})$ given data $\mathcal{D}$ (cf. section 13.3.2) which encompasses the true dynamics, then a natural strategy is to be optimistic with respect to future rewards and pessimistic with respect to future constraint violations. More specifically, we solve the optimization problem,

$$\max_{\pi} \max_{f \in \mathcal{M}(\mathcal{D})} \quad J_\mu(\pi; f) \tag{13.35a}$$

optimistic

$$\text{subject to} \quad \max_{f \in \mathcal{M}(\mathcal{D})} J^c_\mu(\pi; f) \leq \delta. \tag{13.35b}$$

pessimistic

Intuitively, jointly maximizing $J_\mu(\pi; f)$ with respect to π and (plausible) f can lead the agent to try behaviors with potentially high reward due to *optimism* (i.e., the agent "dreams" about potential outcomes). On the other hand, being *pessimistic* with respect to constraint violations enforces the safety constraints (i.e., the agent has "nightmares" about potential outcomes).

If the policy values $J_\mu(\pi; f)$ and $J^c_\mu(\pi; f)$ are modeled as a distribution (e.g., using Bayesian neural networks), then the inner maximizations over plausible dynamics can be approximated using samples from the posterior distributions. Thus, the augmented Lagrangian method can also be used to solve the general optimization problem of eq. (13.35). The resulting algorithm is known as *Lagrangian model-based agent* (LAMBDA) (As et al., 2022).

13.3.4 *Safe Exploration*

As noted in the previous section, in many settings we do not only want to eventually find a safe policy, but we also want to ensure that we act safely while searching for an optimal policy. To this end, recall the general approach outlined in the beginning of the previous section wherein we plan using (pessimistic) confidence bounds of the plausible consequences of our actions.

One key challenge of this approach is that we need to forecast plausible trajectories. The confidence bounds of such trajectories cannot be nicely represented anymore. One approach is to over-approximate the confidence bounds over reachable states from trajectories.

Theorem 13.16 (Koller et al. (2018)). *For conditional Gaussian dynamics, the reachable states of trajectories can be over-approximated with high probability.*

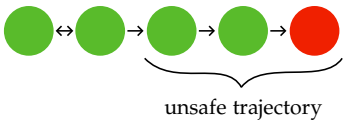


Figure 13.8: Illustration of long-term consequence when planning a finite number of steps. Green dots are to denote safe states and the red dot is to denote an unsafe state. After performing the first action, the agent is still able to return to the previous state. Yet, after reaching the third state, the agent is already guaranteed to end in an unsafe state. When using only a finite horizon of $H = 2$ for planning, the agent might make this transition regardless.

Another key challenge is that actions might have consequences that exceed the time horizon used for planning. In other words, by performing an action now, our agent might put itself into a state that is not yet unsafe, but out of which it cannot escape and which will eventually lead to an unsafe state. You may think of a car driving towards a wall. When a crash with the wall is designated as an unsafe state, there are quite a few states in advance at which it is already impossible to avoid the crash. Thus, looking ahead a finite number of steps is not sufficient to prevent entering unsafe states.

It turns out that it is still possible to use the epistemic uncertainty about the model and short-term plausible behaviors to make guarantees about certain long-term consequences. One idea is to also consider a set of safe states $\mathcal{X}_{\text{safe}}$ alongside the set of unsafe states $\mathcal{X}_{\text{unsafe}}$, for which our agent knows how to stay inside (i.e., remain safe). In other words, for states $\boldsymbol{x} \in \mathcal{X}_{\text{safe}}$, we know that our agent can always behave in such a way that it will not reach an unsafe state.[14] An illustration of this approach is shown in fig. 13.9.

[14] In the example of driving a car, the set of safe states corresponds to those states where we know that we can still safely break before hitting the wall.

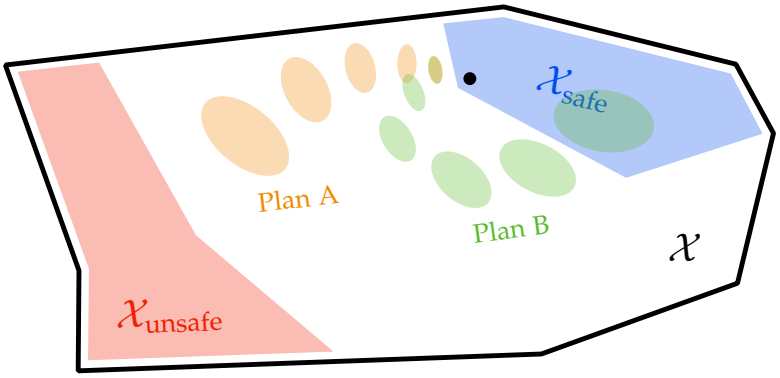


Figure 13.9: Illustration of long-term consequence when planning a finite number of steps. The unsafe set of states is shown in red and the safe set of states is shown in blue. The confidence intervals corresponding to actions of the performance plan and safety plan are shown in orange and green, respectively.

The problem is that this set of safe states might be very conservative. That is to say, it is likely that rewards are mostly attained *outside* of the set of safe states. The key idea is to plan two sequences of actions, instead of only one. "Plan A" (the *performance plan*) is planned with the objective to solve the task, that is, attain maximum reward. "Plan

B" (the *safety plan*) is planned with the objective to return to the set of safe states $\mathcal{X}_{\text{safe}}$. In addition, we enforce that both plans must agree on the first action to be played.

Under the assumption that the confidence bounds are conservative estimates, this guarantees that after playing this action, our agent will still be in a state of which it can return to the set of safe states. In this way, we can gradually increase the set of states that are safe to explore. It can be shown that under suitable conditions, returning to the safe set can be guaranteed (Koller et al., 2018).

13.3.5 Safety Filters

An alternative approach is to (slightly) adjust a potentially unsafe policy π to obtain a policy $\hat{\pi}$ which avoids entering unsafe states with high probability.

Following our interpretation of constrained policy optimization in terms of optimism and pessimism from section 13.3.3, to pessimistically evaluate the safety of a policy with respect to a cost function c given a set of plausible models $\mathcal{M}(\mathcal{D})$, we can use

$$C^\pi(\boldsymbol{x}) \doteq \max_{f \in \mathcal{M}(\mathcal{D})} J^c_{\delta_{\boldsymbol{x}}}(\pi; f) \tag{13.36}$$

$\delta_{\boldsymbol{x}}$ is the point density at $\boldsymbol{x}$

where the initial-state distribution $\delta_{\boldsymbol{x}}$ is to denote that the initial state is $\boldsymbol{x}$. Observe that eq. (13.36) permits a reparameterization in terms of additional decision variables $\boldsymbol{\eta}$ which is analogous to our discussion in section 13.3.2. Concretely, we have

$$C^\pi(\boldsymbol{x}) = \max_{\boldsymbol{\eta}} J^c_{\delta_{\boldsymbol{x}}}(\pi; \widehat{f}) \tag{13.37}$$

where $\widehat{f}$ are the adjusted dynamics (13.26) which are based on the "luck" variables $\boldsymbol{\eta}$. In the context of estimating costs which we aim to minimize (as opposed to rewards which we aim to maximize), $\boldsymbol{\eta}$ can be interpreted as the "bad luck" of the agent.

When our only objective is to act safely, that is, we only aim to minimize cost and are indifferent to rewards, then this reparameterization allows us to find a "maximally safe" policy,

$$\pi_{\text{safe}} \doteq \arg\min_\pi \mathbb{E}_{\boldsymbol{x}\sim\mu}[C^\pi(\boldsymbol{x})] = \arg\min_\pi \max_{\boldsymbol{\eta}} J^c_\mu(\pi; \widehat{f}). \tag{13.38}$$

Under some conditions π_{safe} can be shown to satisfy $\pi_{\text{safe}}(\boldsymbol{x}) = \pi_{\boldsymbol{x}}(\boldsymbol{x})$ for any $\boldsymbol{x}$ where $\pi_{\boldsymbol{x}} \doteq \arg\min_\pi C^\pi(\boldsymbol{x})$ (Curi et al., 2022, proposition 4.2). On its own, the policy π_{safe} is rather useless for exploring the state space. In particular, when in a state that we already deem safe, following policy π_{safe}, the agent aims simply to "stay" within the set of

safe states which means that it has no incentive to explore / maximize reward.

Instead, one can interpret π_{safe} as a "backup" policy in case we realize upon exploring that a certain trajectory is too dangerous, akin to our notion of the "safety plan" in section 13.3.4. That is, given any (potentially explorative and dangerous) policy π, we can find the adjusted policy,

$$\hat{\pi}(\boldsymbol{x}) \doteq \arg\min_{\boldsymbol{a} \in \mathcal{A}} \quad d(\pi(\boldsymbol{x}), \boldsymbol{a}) \tag{13.39a}$$

$$\text{subject to} \quad \max_{\boldsymbol{\eta}} C^{\pi_{\text{safe}}}(\widehat{f}(\boldsymbol{x}, \boldsymbol{a})) \leq \delta \tag{13.39b}$$

for some metric $d(\cdot, \cdot)$ on $\mathcal{A}$. The constraint ensures that the pessimistic next state $\widehat{f}(\boldsymbol{x}, \boldsymbol{a})$ is recoverable by following policy π_{safe}. In this way,

$$\tilde{\pi}(\boldsymbol{x}) \doteq \begin{cases} \hat{\pi}(\boldsymbol{x}) & \text{if } C^{\pi_{\text{safe}}}(\boldsymbol{x}) \leq \delta \\ \pi_{\text{safe}}(\boldsymbol{x}) & \text{if } C^{\pi_{\text{safe}}}(\boldsymbol{x}) > \delta \end{cases} \tag{13.40}$$

is the policy "closest" to π which is δ-safe with respect to the pessimistic cost estimates $C^{\pi_{\text{safe}}}$ (Curi et al., 2022).[15] This is also called a *safety filter*. Similar approaches using backup policies also allow safe exploration in the model-free setting (Sukhija et al., 2023; Widmer et al., 2023).

[15] The policy $\tilde{\pi}$ is required as, a priori, it is not guaranteed that the state $\boldsymbol{x}_t$ will satisfy $C^{\pi_{\text{safe}}}(\boldsymbol{x}_t) \leq \delta$ for all t unless $\hat{\pi}$ is replanned after every step.

Optional Readings

- Curi, Berkenkamp, and Krause (2020).
  Efficient model-based reinforcement learning through optimistic policy search and planning.
- Berkenkamp, Turchetta, Schoellig, and Krause (2017).
  Safe model-based reinforcement learning with stability guarantees.
- Koller, Berkenkamp, Turchetta, and Krause (2018).
  Learning-based model predictive control for safe exploration.
- As, Usmanova, Curi, and Krause (2022).
  Constrained Policy Optimization via Bayesian World Models.
- Curi, Lederer, Hirche, and Krause (2022).
  Safe Reinforcement Learning via Confidence-Based Filters.
- Turchetta, Berkenkamp, and Krause (2019).
  Safe exploration for interactive machine learning.

A

Mathematical Background

A.1 Useful Matrix Identities and Inequalities

- *Woodbury's matrix identity* states that for any matrices $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{C} \in \mathbb{R}^{m\times m}$, and $\boldsymbol{U}, \boldsymbol{V}^\top \in \mathbb{R}^{n\times m}$ where $\boldsymbol{A}$ and $\boldsymbol{C}$ are invertible,

$$(\boldsymbol{A} + \boldsymbol{UCV})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{C}^{-1} + \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}\boldsymbol{A}^{-1}. \tag{A.1}$$

  The following identity is a direct consequence:

$$(\boldsymbol{A} + \boldsymbol{x}\boldsymbol{x}^\top)^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{x}(1 + \boldsymbol{x}^\top\boldsymbol{A}^{-1}\boldsymbol{x})^{-1}\boldsymbol{x}^\top\boldsymbol{A}^{-1} \tag{A.2a}$$

$$= \boldsymbol{A}^{-1} - \frac{(\boldsymbol{A}^{-1}\boldsymbol{x})(\boldsymbol{A}^{-1}\boldsymbol{x})^\top}{1 + \boldsymbol{x}^\top\boldsymbol{A}^{-1}\boldsymbol{x}} \tag{A.2b}$$

  for any symmetric and invertible matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{x} \in \mathbb{R}^n$.
- The *matrix inversion lemma* states that for matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$,

$$(\boldsymbol{A} + \boldsymbol{B})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}(\boldsymbol{A}^{-1} + \boldsymbol{B}^{-1})^{-1}\boldsymbol{A}^{-1}. \tag{A.3}$$

- *Hadamard's inequality* states that

$$\det(\boldsymbol{M}) \leq \prod_{i=0}^{d} \boldsymbol{M}(i,i) \tag{A.4}$$

  for any positive definite matrix $\boldsymbol{M} \in \mathbb{R}^{d\times d}$.
- The *Weinstein-Aronszajn identity* for positive definite matrices $\boldsymbol{A} \in \mathbb{R}^{d\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times d}$ states that

$$\det(\boldsymbol{I}_{d\times d} + \boldsymbol{AB}) = \det(\boldsymbol{I}_{n\times n} + \boldsymbol{BA}). \tag{A.5}$$

B

Solutions

Fundamentals

Solution to exercise 1.5 (Properties of probability).

1. By the third axiom and $B = A \cup (B \setminus A)$,

$$\mathbb{P}(B) = \mathbb{P}(A) + \mathbb{P}(B \setminus A).$$

   Noting from the first axiom that $\mathbb{P}(B \setminus A) \geq 0$ completes the proof.
2. By the second axiom,

$$\mathbb{P}\big(A \cup \overline{A}\big) = \mathbb{P}(\Omega) = 1$$

   and by the third axiom,

$$\mathbb{P}\big(A \cup \overline{A}\big) = \mathbb{P}(A) + \mathbb{P}\big(\overline{A}\big).$$

   Reorganizing the equations completes the proof.
3. Define the countable sequence of events

$$A_i' \doteq A_i \setminus \left(\bigcup_{j=1}^{i-1} A_j \right).$$

   Note that the sequence of events $\{A_i'\}_i$ is disjoint. Thus, we have by the third axiom and then using (1) that

$$\mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i \right) = \mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i' \right) = \sum_{i=1}^{\infty} \mathbb{P}\big(A_i'\big) \leq \sum_{i=1}^{\infty} \mathbb{P}(A_i).$$

Solution to exercise 1.19 (Random walks on graphs). We show that any vertex v is visited eventually with probability 1.

We denote by $w \to v$ the event that the random walk starting at vertex w visits the vertex v eventually, we denote by $\Gamma(w)$ the neighborhood

of w, and we write $\deg(w) \doteq |\Gamma(w)|$. We have,

$$\begin{aligned}\mathbb{P}(w \to v) &= \sum_{w' \in \Gamma(w)} \mathbb{P}\big(\text{the random walk first visits } w'\big) \cdot \mathbb{P}\big(w' \to v\big) && \text{using the law of total probability (1.15)} \\ &= \frac{1}{\deg(w)} \sum_{w' \in \Gamma(w)} \mathbb{P}\big(w' \to v\big). && \text{using that the random walk moves to a neighbor uniformly at random}\end{aligned}$$

Take u to be the vertex such that $\mathbb{P}(u \to v)$ is minimized. Then,

$$\mathbb{P}(u \to v) = \frac{1}{\deg(u)} \sum_{u' \in \Gamma(u)} \underbrace{\mathbb{P}\big(u' \to v\big)}_{\geq \mathbb{P}(u \to v)} \geq \mathbb{P}(u \to v).$$

That is, for all neighbors u' of u, $\mathbb{P}(u \to v) = \mathbb{P}(u' \to v)$. Using that the graph is connected and finite, we conclude $\mathbb{P}(u \to v) = \mathbb{P}(w \to v)$ for any vertex w. Finally, note that $\mathbb{P}(v \to v) = 1$, and hence, the random walk starting at any vertex u visits the vertex v eventually with probability 1.

Solution to exercise 1.26 (Law of total expectation). Let $\mathbb{1}\{A_i\}$ be the indicator random variable for the event A_i. Then,

$$\begin{aligned}\mathbb{E}[\mathbf{X} \cdot \mathbb{1}\{A_i\}] &= \mathbb{E}[\mathbb{E}[\mathbf{X} \cdot \mathbb{1}\{A_i\} \mid \mathbb{1}\{A_i\}]] && \text{by the tower rule (1.33)} \\ &= \mathbb{E}[\mathbf{X} \cdot \mathbb{1}\{A_i\} \mid \mathbb{1}\{A_i\} = 1] \cdot \mathbb{P}(\mathbb{1}\{A_i\} = 1) && \text{expanding the outer expectation} \\ &\qquad + 0 \cdot \mathbb{P}(\mathbb{1}\{A_i\} = 0) \\ &= \mathbb{E}[\mathbf{X} \mid A_i] \cdot \mathbb{P}(A_i). && \text{the event } \mathbb{1}\{A_i\} = 1 \text{ is equivalent to } A_i\end{aligned}$$

Summing up for all i, the left-hand side becomes

$$\mathbb{E}[\mathbf{X}] = \sum_{i=1}^{k} \mathbb{E}[\mathbf{X} \cdot \mathbb{1}\{A_i\}].$$

Solution to exercise 1.28 (Covariance matrices are positive semi-definite). Let $\boldsymbol{\Sigma} \doteq \mathrm{Var}[\mathbf{X}]$ be a covariance matrix of the random vector $\mathbf{X}$ and fix any $\boldsymbol{z} \in \mathbb{R}^n$. Then,

$$\begin{aligned}\boldsymbol{z}^\top \boldsymbol{\Sigma} \boldsymbol{z} &= \boldsymbol{z}^\top \mathbb{E}\Big[(\mathbf{X} - \mathbb{E}[\mathbf{X}])(\mathbf{X} - \mathbb{E}[\mathbf{X}])^\top\Big] \boldsymbol{z} && \text{using the definitiion of variance (1.44)} \\ &= \mathbb{E}\Big[\boldsymbol{z}^\top (\mathbf{X} - \mathbb{E}[\mathbf{X}])(\mathbf{X} - \mathbb{E}[\mathbf{X}])^\top \boldsymbol{z}\Big]. && \text{using linearity of expectation (1.24)}\end{aligned}$$

Define the random variable $Z \doteq \boldsymbol{z}^\top (\mathbf{X} - \mathbb{E}[\mathbf{X}])$. Then,

$$= \mathbb{E}\Big[Z^2\Big] \geq 0.$$

Solution to exercise 1.33 (Bayes' rule). Let us start by defining some events that we will reason about later. For ease of writing, let us call the person in question X.

$$D = \text{X has the disease},$$

$P =$ The test shows a positive response.

Now we can translate the information in the question to formal statements in terms of D and P,

$$\mathbb{P}(D) = 10^{-4} \qquad \text{the disease is rare}$$
$$\mathbb{P}(P \mid D) = \mathbb{P}\big(\overline{P} \mid \overline{D}\big) = 0.99. \qquad \text{the test is accurate}$$

We want to determine $\mathbb{P}(D \mid P)$. One can find this probability by using Bayes' rule (1.59),

$$\mathbb{P}(D \mid P) = \frac{\mathbb{P}(P \mid D) \cdot \mathbb{P}(D)}{\mathbb{P}(P)}.$$

From the quantities above, we have everything except for $\mathbb{P}(P)$. This, however, we can compute using the law of total probability,

$$\begin{aligned}\mathbb{P}(P) &= \mathbb{P}(P \mid D) \cdot \mathbb{P}(D) + \mathbb{P}\big(P \mid \overline{D}\big) \cdot \mathbb{P}\big(\overline{D}\big) \\ &= 0.99 \cdot 10^{-4} + 0.01 \cdot (1 - 10^{-4}) \\ &= 0.010098.\end{aligned}$$

Hence, $\mathbb{P}(D \mid P) = 0.99 \cdot 10^{-4} / 0.010098 \approx 0.0098 = 0.98\%$.

Solution to exercise 1.38 (Sample variance). Using that X is zero-mean, we have that

$$\mathbb{E}\Big[\overline{X}_n^2\Big] = \mathrm{Var}\big[\overline{X}_n\big] = \frac{1}{n}\mathrm{Var}[X] \qquad \text{and}$$
$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} {X^{(i)}}^2\right] = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\Big[{X^{(i)}}^2\Big] = \mathrm{Var}[X].$$

Thus,

$$\mathbb{E}\Big[S_n^2\Big] = \frac{n}{n-1}\left(\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} {X^{(i)}}^2\right] - \mathbb{E}\Big[\overline{X}_n^2\Big]\right) = \mathrm{Var}[X].$$

Solution to exercise 1.43 (Simple concentration inequalities).

1. W.l.o.g. we assume that X is continuous. We have

$$\begin{aligned}\mathbb{E}X &= \int_0^\infty x \cdot p(x)\,dx && \text{using the definition of expectation (1.23b)} \\ &= \underbrace{\int_0^\epsilon x \cdot p(x)\,dx}_{\geq 0} + \int_\epsilon^\infty x \cdot p(x)\,dx \geq \epsilon \underbrace{\int_\epsilon^\infty \cdot p(x)\,dx}_{\mathbb{P}(X \geq \epsilon)}.\end{aligned}$$

2. Consider the non-negative random variable $Y \doteq (X - \mathbb{E}X)^2$. We have

$$\begin{aligned}\mathbb{P}(|X - \mathbb{E}X| \geq \epsilon) &= \mathbb{P}\Big((X - \mathbb{E}X)^2 \geq \epsilon^2\Big) \\ &\leq \frac{\mathbb{E}\big[(X - \mathbb{E}X)^2\big]}{\epsilon^2} && \text{using Markov's inequality (1.82)} \\ &= \frac{\mathrm{Var}X}{\epsilon^2}. && \text{using the definition of variance (1.44)}\end{aligned}$$

Solution to exercise 1.45 (Weak law of large numbers). Fix any $\epsilon > 0$. Applying Chebyshev's inequality and noting that $\mathbb{E}\overline{X}_n = \mathbb{E}X$, we obtain

$$\mathbb{P}\big(\big|\overline{X}_n - \mathbb{E}X\big| \geq \epsilon\big) \leq \frac{\mathrm{Var}\overline{X}_n}{\epsilon^2}.$$

We further have for the variance of the sample mean that

$$\mathrm{Var}\overline{X}_n = \mathrm{Var}\left[\frac{1}{n}\sum_{i=1}^{n} X_i\right] = \frac{1}{n^2}\sum_{i=1}^{n}\mathrm{Var}[X_i] = \frac{\mathrm{Var}X}{n}.$$

Thus,

$$\lim_{n\to\infty}\mathbb{P}\big(\big|\overline{X}_n - \mathbb{E}X\big| \geq \epsilon\big) \leq \lim_{n\to\infty}\frac{\mathrm{Var}X}{\epsilon^2 n} = 0$$

which is precisely the definition of $\overline{X}_n \xrightarrow{\mathbb{P}} \mathbb{E}X$.

Solution to exercise 1.55 (Directional derivatives). Fix a $\lambda > 0$. Using a first-order expansion, we have

$$f(\boldsymbol{x} + \lambda\boldsymbol{d}) = f(\boldsymbol{x}) + \lambda\boldsymbol{\nabla}f(\boldsymbol{x})^\top\boldsymbol{d} + o(\lambda\left\|\boldsymbol{d}\right\|_2).$$

Dividing by λ yields,

$$\frac{f(\boldsymbol{x} + \lambda\boldsymbol{d}) - f(\boldsymbol{x})}{\lambda} = \boldsymbol{\nabla}f(\boldsymbol{x})^\top\boldsymbol{d} + \underbrace{\frac{o(\lambda\left\|\boldsymbol{d}\right\|_2)}{\lambda}}_{\to 0}.$$

Taking the limit $\lambda \to 0$ gives the desired result.

Solution to exercise 1.65 (Product of Gaussian PDFs). We need to find $\boldsymbol{\mu} \in \mathbb{R}^n$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$ such that

$$(\boldsymbol{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \propto (\boldsymbol{x}-\boldsymbol{\mu}_1)^\top\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) + (\boldsymbol{x}-\boldsymbol{\mu}_2)^\top\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2). \tag{B.1}$$

for any $\boldsymbol{x} \in \mathbb{R}^n$.

The left-hand side of eq. (B.1) is equal to

$$\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x} - 2\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \boldsymbol{\mu}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}.$$

The right-hand side of eq. (B.1) is equal to

$$\begin{aligned}
&\Big(\boldsymbol{x}^\top\boldsymbol{\Sigma}_1^{-1}\boldsymbol{x} + \boldsymbol{x}^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{x}\Big) - 2\Big(\boldsymbol{x}^\top\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{x}^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\Big) \\
&\qquad\qquad + \Big(\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\Big) \\
&= \boldsymbol{x}^\top\Big(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}\Big)\boldsymbol{x} - 2\boldsymbol{x}^\top\Big(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\Big) \\
&\qquad\qquad + \Big(\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\Big).
\end{aligned}$$

We observe that both sides are equal if

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \quad \text{and} \quad \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} = \boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2.$$

Solution to exercise 1.67 (Marginal / conditional distribution of a Gaussian). Let $\boldsymbol{x} \sim \mathbf{X}$. The joint distribution can be expressed as

$$\begin{aligned} p(\boldsymbol{x}) &= p(\boldsymbol{x}_A, \boldsymbol{x}_B) \\ &= \frac{1}{Z} \exp\left(-\frac{1}{2} \begin{bmatrix} \boldsymbol{x}_A - \boldsymbol{\mu}_A \\ \boldsymbol{x}_B - \boldsymbol{\mu}_B \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{\Lambda}_{AA} & \boldsymbol{\Lambda}_{AB} \\ \boldsymbol{\Lambda}_{BA} & \boldsymbol{\Lambda}_{BB} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_A - \boldsymbol{\mu}_A \\ \boldsymbol{x}_B - \boldsymbol{\mu}_B \end{bmatrix} \right) \end{aligned}$$

where Z denotes the normalizing constant. To simplify the notation, we write

$$\begin{bmatrix} \boldsymbol{\Delta}_A \\ \boldsymbol{\Delta}_B \end{bmatrix} \doteq \begin{bmatrix} \boldsymbol{x}_A - \boldsymbol{\mu}_A \\ \boldsymbol{x}_B - \boldsymbol{\mu}_B \end{bmatrix}.$$

1. We obtain

$$\begin{aligned} & p(\boldsymbol{x}_A) \\ &= \frac{1}{Z} \int \exp\left(-\frac{1}{2} \begin{bmatrix} \boldsymbol{\Delta}_A \\ \boldsymbol{\Delta}_B \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{\Lambda}_{AA} & \boldsymbol{\Lambda}_{AB} \\ \boldsymbol{\Lambda}_{BA} & \boldsymbol{\Lambda}_{BB} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta}_A \\ \boldsymbol{\Delta}_B \end{bmatrix} \right) d\boldsymbol{x}_B && \text{using the sum rule (1.10)} \\ &= \frac{1}{Z} \exp\left(-\frac{1}{2} \left[\boldsymbol{\Delta}_A^\top (\boldsymbol{\Lambda}_{AA} - \boldsymbol{\Lambda}_{AB} \boldsymbol{\Lambda}_{BB}^{-1} \boldsymbol{\Lambda}_{BA}) \boldsymbol{\Delta}_A \right] \right) && \text{using the first hint} \\ &\quad \cdot \int \exp\left(-\frac{1}{2} \left[(\boldsymbol{\Delta}_B + \boldsymbol{\Lambda}_{BB}^{-1} \boldsymbol{\Lambda}_{BA} \boldsymbol{\Delta}_A)^\top \boldsymbol{\Lambda}_{BB} (\boldsymbol{\Delta}_B + \boldsymbol{\Lambda}_{BB}^{-1} \boldsymbol{\Lambda}_{BA} \boldsymbol{\Delta}_A) \right] \right) d\boldsymbol{x}_B. \end{aligned}$$

   Observe that the integrand is an unnormalized Gaussian PDF, and hence, the integral evaluates to $\sqrt{\det\left(2\pi \boldsymbol{\Lambda}_{BB}^{-1}\right)}$ (so is constant with respect to $\boldsymbol{x}_A$). We can therefore simplify to

$$\begin{aligned} &= \frac{1}{Z'} \exp\left(-\frac{1}{2} \left[\boldsymbol{\Delta}_A^\top (\boldsymbol{\Lambda}_{AA} - \boldsymbol{\Lambda}_{AB} \boldsymbol{\Lambda}_{BB}^{-1} \boldsymbol{\Lambda}_{BA}) \boldsymbol{\Delta}_A \right] \right) \\ &= \frac{1}{Z'} \exp\left(-\frac{1}{2} \left[\boldsymbol{\Delta}_A^\top \boldsymbol{\Sigma}_{AA}^{-1} \boldsymbol{\Delta}_A \right] \right). && \text{using the second hint} \end{aligned}$$

2. We obtain

$$\begin{aligned} & p(\boldsymbol{x}_B \mid \boldsymbol{x}_A) \\ &= \frac{p(\boldsymbol{x}_A, \boldsymbol{x}_B)}{p(\boldsymbol{x}_A)} && \text{using the definition of conditional distributions (1.13)} \\ &= \frac{1}{Z'} \exp\left(-\frac{1}{2} \begin{bmatrix} \boldsymbol{\Delta}_A \\ \boldsymbol{\Delta}_B \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{\Lambda}_{AA} & \boldsymbol{\Lambda}_{AB} \\ \boldsymbol{\Lambda}_{BA} & \boldsymbol{\Lambda}_{BB} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta}_A \\ \boldsymbol{\Delta}_B \end{bmatrix} \right) && \text{noting that } p(\boldsymbol{x}_A) \text{ is constant with respect to } \boldsymbol{x}_B \\ &= \frac{1}{Z'} \exp\left(-\frac{1}{2} \left[\boldsymbol{\Delta}_A^\top (\boldsymbol{\Lambda}_{AA} - \boldsymbol{\Lambda}_{AB} \boldsymbol{\Lambda}_{BB}^{-1} \boldsymbol{\Lambda}_{BA}) \boldsymbol{\Delta}_A \right] \right) && \text{using the first hint} \\ &\quad \cdot \exp\left(-\frac{1}{2} \left[(\boldsymbol{\Delta}_B + \boldsymbol{\Lambda}_{BB}^{-1} \boldsymbol{\Lambda}_{BA} \boldsymbol{\Delta}_A)^\top \boldsymbol{\Lambda}_{BB} (\boldsymbol{\Delta}_B + \boldsymbol{\Lambda}_{BB}^{-1} \boldsymbol{\Lambda}_{BA} \boldsymbol{\Delta}_A) \right] \right) \\ &= \frac{1}{Z''} \exp\left(-\frac{1}{2} \left[(\boldsymbol{\Delta}_B + \boldsymbol{\Lambda}_{BB}^{-1} \boldsymbol{\Lambda}_{BA} \boldsymbol{\Delta}_A)^\top \boldsymbol{\Lambda}_{BB} (\boldsymbol{\Delta}_B + \boldsymbol{\Lambda}_{BB}^{-1} \boldsymbol{\Lambda}_{BA} \boldsymbol{\Delta}_A) \right] \right). && \text{observing that the first exponential is constant with respect to } \boldsymbol{x}_B \end{aligned}$$

Finally, observe that

$$\boldsymbol{\Delta}_B + \boldsymbol{\Lambda}_{BB}^{-1}\boldsymbol{\Lambda}_{BA}\boldsymbol{\Delta}_A = \boldsymbol{\mu}_B - \boldsymbol{\Lambda}_{BB}^{-1}\boldsymbol{\Lambda}_{BA}(\boldsymbol{x}_A - \boldsymbol{\mu}_A) = \boldsymbol{\mu}_{B|A} \quad \text{and} \qquad \text{using the third hint}$$

$$\boldsymbol{\Lambda}_{BB}^{-1} = \boldsymbol{\Sigma}_{BB} - \boldsymbol{\Sigma}_{BA}\boldsymbol{\Sigma}_{AA}^{-1}\boldsymbol{\Sigma}_{AB} = \boldsymbol{\Sigma}_{B|A}. \qquad \text{using the second hint}$$

Thus, $\boldsymbol{x}_B \mid \boldsymbol{x}_A \sim \mathcal{N}(\boldsymbol{\mu}_{B|A}, \boldsymbol{\Sigma}_{B|A})$.

Solution to exercise 1.69 (Closedness properties of Gaussians).

1. Let $\mathbf{Y} \doteq \boldsymbol{A}\mathbf{X} + \boldsymbol{b}$ and define $\boldsymbol{s} \doteq \boldsymbol{A}^\top \boldsymbol{t}$. We have for the MGF of $\mathbf{Y}$ that

$$\begin{aligned}
\varphi_{\mathbf{Y}}(\boldsymbol{t}) &= \mathbb{E}\exp\left(\boldsymbol{t}^\top \mathbf{Y}\right) \\
&= \mathbb{E}\exp\left(\boldsymbol{t}^\top \boldsymbol{A}\mathbf{X}\right) \cdot \exp\left(\boldsymbol{t}^\top \boldsymbol{b}\right) \\
&= \mathbb{E}\exp\left(\boldsymbol{s}^\top \mathbf{X}\right) \cdot \exp\left(\boldsymbol{t}^\top \boldsymbol{b}\right) \\
&= \varphi_{\mathbf{X}}(\boldsymbol{s}) \cdot \exp\left(\boldsymbol{t}^\top \boldsymbol{b}\right) \\
&= \exp\left(\boldsymbol{s}^\top \boldsymbol{\mu} + \frac{1}{2}\boldsymbol{s}^\top \boldsymbol{\Sigma}\boldsymbol{s}\right) \cdot \exp\left(\boldsymbol{t}^\top \boldsymbol{b}\right) \\
&= \exp\left(\boldsymbol{t}^\top (\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b}) + \frac{1}{2}\boldsymbol{t}^\top \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top \boldsymbol{t}\right),
\end{aligned}$$

   which implies that $\mathbf{Y} \sim \mathcal{N}(\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b}, \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top)$.
2. We have for the MGF of $\mathbf{Y} \doteq \mathbf{X} + \mathbf{X}'$ that

$$\begin{aligned}
\varphi_{\mathbf{Y}}(\boldsymbol{t}) &= \mathbb{E}\exp\left(\boldsymbol{t}^\top \mathbf{Y}\right) \\
&= \mathbb{E}\exp\left(\boldsymbol{t}^\top \mathbf{X}\right) \cdot \mathbb{E}\exp\left(\boldsymbol{t}^\top \mathbf{X}'\right) && \text{using the independence of } \mathbf{X} \text{ and } \mathbf{X}' \\
&= \varphi_{\mathbf{X}}(\boldsymbol{t}) \cdot \varphi_{\mathbf{X}'}(\boldsymbol{t}) \\
&= \exp\left(\boldsymbol{t}^\top \boldsymbol{\mu} + \frac{1}{2}\boldsymbol{t}^\top \boldsymbol{\Sigma}\boldsymbol{t}\right) \cdot \exp\left(\boldsymbol{t}^\top \boldsymbol{\mu}' + \frac{1}{2}\boldsymbol{t}^\top \boldsymbol{\Sigma}'\boldsymbol{t}\right) \\
&= \exp\left(\boldsymbol{t}^\top (\boldsymbol{\mu} + \boldsymbol{\mu}') + \frac{1}{2}\boldsymbol{t}^\top (\boldsymbol{\Sigma} + \boldsymbol{\Sigma}')\boldsymbol{t}\right),
\end{aligned}$$

   which implies that $\mathbf{Y} \sim \mathcal{N}(\boldsymbol{\mu} + \boldsymbol{\mu}', \boldsymbol{\Sigma} + \boldsymbol{\Sigma}')$.

Solution to exercise 1.72 (Expectation and Variance of Gaussians). Note that if $Y \sim \mathcal{N}(0,1)$ is a (univariate) standard normal random variable, then

$$\begin{aligned}
\mathbb{E}[Y] &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} y \cdot \exp\left(-\frac{1}{2}y^2\right) dy && \text{using the PDF of the standard normal distribution (1.6)} \\
&= \frac{1}{\sqrt{2\pi}} \left(\int_{-\infty}^{0} y \cdot \exp\left(-\frac{1}{2}y^2\right) dy + \int_{0}^{\infty} y \cdot \exp\left(-\frac{1}{2}y^2\right) dy \right) \\
&= \frac{1}{\sqrt{2\pi}} \left(\left[-\exp\left(-\frac{1}{2}y^2\right) \right]_{-\infty}^{0} + \left[-\exp\left(-\frac{1}{2}y^2\right) \right]_{0}^{\infty} \right)
\end{aligned}$$

$$= \frac{1}{\sqrt{2\pi}}([-1+0]+[0+1]) = 0,$$

$$\begin{aligned}
\operatorname{Var}[Y] &= \mathbb{E}\left[Y^2\right] - \underbrace{\mathbb{E}[Y]^2}_{0} && \text{using the definition of variance (1.45)}\\
&= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} y^2 \cdot \exp\left(-\frac{1}{2}y^2\right) dy && \text{using the PDF of the standard normal distribution (1.6)}\\
&= \frac{1}{\sqrt{2\pi}} \left(\int_{-\infty}^{0} y^2 \cdot \exp\left(-\frac{1}{2}y^2\right) dy + \int_{0}^{\infty} y^2 \cdot \exp\left(-\frac{1}{2}y^2\right) dy \right)\\
&= \frac{1}{\sqrt{2\pi}} \left(\left[-y \cdot \exp\left(-\frac{1}{2}y^2\right) \right]_{-\infty}^{0} + \int_{-\infty}^{0} \exp\left(-\frac{1}{2}y^2\right) dy \right. && \text{integrating by parts}\\
&\qquad \left. + \left[-y \cdot \exp\left(-\frac{1}{2}y^2\right) \right]_{0}^{\infty} + \int_{0}^{\infty} \exp\left(-\frac{1}{2}y^2\right) dy \right)\\
&= \frac{1}{\sqrt{2\pi}} \left(\int_{-\infty}^{0} \exp\left(-\frac{1}{2}y^2\right) dy + \int_{0}^{\infty} \exp\left(-\frac{1}{2}y^2\right) dy \right)\\
&= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp\left(-\frac{1}{2}y^2\right) dy = 1. && \text{a PDF integrates to 1}
\end{aligned}$$

Recall from eq. (1.125) that we can express $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ as

$$\mathbf{X} = \boldsymbol{\Sigma}^{1/2}\mathbf{Y} + \boldsymbol{\mu}$$

where $\mathbf{Y} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Using that $\mathbf{Y}$ is a vector of independent univariate standard normal random variables, we conclude that $\mathbb{E}[\mathbf{Y}] = \mathbf{0}$ and $\operatorname{Var}[\mathbf{Y}] = \boldsymbol{I}$. We obtain

$$\mathbb{E}[\mathbf{X}] = \mathbb{E}\left[\boldsymbol{\Sigma}^{1/2}\mathbf{Y} + \boldsymbol{\mu}\right] = \boldsymbol{\Sigma}^{1/2}\underbrace{\mathbb{E}[\mathbf{Y}]}_{\mathbf{0}} + \boldsymbol{\mu} = \boldsymbol{\mu}, \quad \text{and} \qquad \text{using linearity of expectation (1.24)}$$

$$\operatorname{Var}[\mathbf{X}] = \operatorname{Var}\left[\boldsymbol{\Sigma}^{1/2}\mathbf{Y} + \boldsymbol{\mu}\right] = \boldsymbol{\Sigma}^{1/2}\underbrace{\operatorname{Var}[\mathbf{Y}]}_{\boldsymbol{I}}{\boldsymbol{\Sigma}^{1/2}}^{\top} = \boldsymbol{\Sigma}. \qquad \text{using eq. (1.47)}$$

Bayesian Linear Regression

Solution to exercise 2.2 (Closed-form linear regression).

1. We begin by deriving the gradient and Hessian of the least squares and ridge losses. For least squares,

$$\begin{aligned}
\boldsymbol{\nabla}_{\boldsymbol{w}} \, \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 &= \boldsymbol{\nabla}_{\boldsymbol{w}} \left(\boldsymbol{w}^\top \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{w}^\top \boldsymbol{X}^\top \boldsymbol{y} + \|\boldsymbol{y}\|_2^2\right)\\
&= 2(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^\top \boldsymbol{y}),\\
\boldsymbol{H}_{\boldsymbol{w}} \, \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 &= 2\boldsymbol{X}^\top \boldsymbol{X}.
\end{aligned}$$

   Similarly, for ridge regression,

$$\begin{aligned}
\boldsymbol{\nabla}_{\boldsymbol{w}} \, \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda \, \|\boldsymbol{w}\|_2^2 &= 2(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^\top \boldsymbol{y}) + \lambda \boldsymbol{w},\\
\boldsymbol{H}_{\boldsymbol{w}} \, \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda \, \|\boldsymbol{w}\|_2^2 &= 2\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I}.
\end{aligned}$$

From the assumption that the Hessians are positive definite, we know that any minimizer is a unique globally optimal solution (due to strict convexity), and that $\boldsymbol{X}^\top \boldsymbol{X}$ and $\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I}$ are invertible.
Using the first-order optimality condition for convex functions, we attain the solutions to least squares and ridge regression by setting the respective gradient to $\mathbf{0}$.

2. We choose $\hat{\boldsymbol{w}}_{\text{ls}}$ such that $\boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ls}} = \boldsymbol{\Pi}_{\boldsymbol{X}}\boldsymbol{y}$. This implies that $\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ls}} \perp \boldsymbol{X}\boldsymbol{w}$ for all $\boldsymbol{w} \in \mathbb{R}^d$. In other words, $(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ls}})^\top \boldsymbol{X}\boldsymbol{w} = 0$ for all $\boldsymbol{w}$, which implies that $(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ls}})^\top \boldsymbol{X} = \mathbf{0}$. By simple algebraic manipulation it can be seen that this condition is equivalent to the gradient condition of the previous exercise.

Solution to exercise 2.4 (Variance of least squares around training data). In the two-dimensional setting, i.e., data is of the form $\boldsymbol{x}_i = [1\ x_i]^\top$ $(x_i \in \mathbb{R})$, we have

$$\boldsymbol{X}^\top \boldsymbol{X} = \begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix}.$$

Thus,

$$\operatorname{Var}[\hat{\boldsymbol{w}}_{\text{ls}}] = \sigma_n^2 (\boldsymbol{X}^\top \boldsymbol{X})^{-1} = \frac{\sigma_n^2}{Z} \begin{bmatrix} \sum x_i^2 & -\sum x_i \\ -\sum x_i & n \end{bmatrix} \qquad \text{using eq. (2.9)}$$

where $Z \doteq n(\sum x_i^2) - (\sum x_i)^2$.

Therefore, the predictive variance at a point $[1\ x^\star]^\top$ is

$$\begin{bmatrix} 1 & x^\star \end{bmatrix} \operatorname{Var}[\hat{\boldsymbol{w}}_{\text{ls}}] \begin{bmatrix} 1 \\ x^\star \end{bmatrix} = \frac{\sigma_n^2}{Z} \sum x_i^2 - 2x_i x^\star + (x^\star)^2 = \frac{\sigma_n^2}{Z} \sum (x_i - x^\star)^2.$$

Thus, the predictive variance is minimized for $x^\star = \frac{1}{n} \sum x_i$.

Solution to exercise 2.8 (Aleatoric and epistemic uncertainty of BLR). The law of total variance (2.22) yields the following decomposition of the predictive variance,

$$\begin{aligned} \operatorname{Var}_{y^\star}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, y_{1:n}] = \mathbb{E}_{\boldsymbol{w}}\left[\operatorname{Var}_{y^\star}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{w}] \mid \boldsymbol{x}_{1:n}, y_{1:n}\right] \\ + \operatorname{Var}_{\boldsymbol{w}}\left[\mathbb{E}_{y^\star}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{w}] \mid \boldsymbol{x}_{1:n}, y_{1:n}\right] \end{aligned}$$

wherein the first term corresponds to the aleatoric uncertainty and the second term corresponds to the epistemic uncertainty.

The aleatoric uncertainty is given by

$$\mathbb{E}_{\boldsymbol{w}}\left[\operatorname{Var}_{y^\star}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{w}] \mid \boldsymbol{x}_{1:n}, y_{1:n}\right] = \mathbb{E}_{\boldsymbol{w}}\left[\sigma_n^2 \,\middle|\, \boldsymbol{x}_{1:n}, y_{1:n}\right] = \sigma_n^2 \qquad \text{using the definition of } \sigma_n^2 \text{. (2.7)}$$

For the epistemic uncertainty,

$$\begin{aligned}
\mathrm{Var}_{\boldsymbol{w}}\left[\mathbb{E}_{y^\star}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{w}] \mid \boldsymbol{x}_{1:n}, y_{1:n}\right] &= \mathrm{Var}_{\boldsymbol{w}}\left[\boldsymbol{w}^\top \boldsymbol{x}^\star \;\middle|\; \boldsymbol{x}_{1:n}, y_{1:n}\right] && \text{using that } y^\star = \boldsymbol{w}^\top \boldsymbol{x}^\star + \epsilon \text{ where } \epsilon \text{ is zero-mean noise} \\
&= \boldsymbol{x}^{\star\top} \mathrm{Var}_{\boldsymbol{w}}[\boldsymbol{w} \mid \boldsymbol{x}_{1:n}, y_{1:n}]\, \boldsymbol{x}^\star && \text{using eq. (1.47)} \\
&= \boldsymbol{x}^{\star\top} \boldsymbol{\Sigma} \boldsymbol{x}^\star && \text{using eq. (2.13)}
\end{aligned}$$

where $\boldsymbol{\Sigma}$ is the posterior covariance matrix.

Solution to exercise 2.7 (Bayesian linear regression).

1. Recall from section 2.1.1 that the MLE and least squares estimate coincide if the noise is zero-mean Gaussian. We therefore have,

$$\hat{\boldsymbol{w}}_{\mathrm{MLE}} = \hat{\boldsymbol{w}}_{\mathrm{ls}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y} = \begin{bmatrix} 0.63 \\ 1.83 \end{bmatrix}.$$

2. Recall from eq. (2.13) that $\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with

$$\begin{aligned}
\boldsymbol{\Sigma} &= \left(\sigma_n^{-2} \boldsymbol{X}^\top \boldsymbol{X} + \sigma_p^{-2} \boldsymbol{I}\right)^{-1} \quad \text{and} \\
\boldsymbol{\mu} &= \sigma_n^{-2} \boldsymbol{\Sigma} \boldsymbol{X}^\top \boldsymbol{y}.
\end{aligned}$$

   Inserting $\sigma_n^2 = 0.1$ and $\sigma_p^2 = 0.05$ yields,

$$\boldsymbol{\Sigma} = \begin{bmatrix} 0.019 & -0.014 \\ -0.014 & 0.019 \end{bmatrix}.$$

   Then,

$$\hat{\boldsymbol{w}}_{\mathrm{MAP}} = \boldsymbol{\mu} = \begin{bmatrix} 0.91 \\ 1.31 \end{bmatrix}.$$

3. Recall from eq. (2.21) that $y^\star \mid \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}^\star \sim \mathcal{N}(\mu_y^\star, \sigma_{y^\star}^2)$ with

$$\mu_y^\star = \boldsymbol{\mu}^\top \boldsymbol{x}^\star \quad \text{and} \quad \sigma_{y^\star}^2 = \boldsymbol{x}^{\star\top} \boldsymbol{\Sigma} \boldsymbol{x}^\star + \sigma_n^2.$$

   Inserting $\boldsymbol{x}^\star = [3\ 3]^\top$, and $\sigma_n^2, \boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ from above yields,

$$\mu_y^\star = 6.66 \quad \text{and} \quad \sigma_{y^\star}^2 = 0.186.$$

4. One would have to let $\sigma_p^2 \to \infty$.

Solution to exercise 2.9 (Online Bayesian linear regression). We denote by $\boldsymbol{X}_t$ the design matrix and by $\boldsymbol{y}_t$ the vector of observations including the first t data points.

1. Note that

$$\boldsymbol{X}_t^\top \boldsymbol{X}_t = \sum_{i=1}^{t} \boldsymbol{x}_i \boldsymbol{x}_i^\top \quad \text{and} \quad \boldsymbol{X}_t^\top \boldsymbol{y}_t = \sum_{i=1}^{t} y_i \boldsymbol{x}_i.$$

This means that after observing the $(t+1)$-st data point, we have that

$$\begin{aligned} \boldsymbol{X}_{t+1}^\top \boldsymbol{X}_{t+1} &= \boldsymbol{X}_t^\top \boldsymbol{X}_t + \boldsymbol{x}_{t+1}\boldsymbol{x}_{t+1}^\top \quad \text{and} \\ \boldsymbol{X}_{t+1}^\top \boldsymbol{y}_{t+1} &= \boldsymbol{X}_t^\top \boldsymbol{y}_t + y_{t+1}\boldsymbol{x}_{t+1}. \end{aligned}$$

Hence, by just keeping $\boldsymbol{X}_t^\top \boldsymbol{X}_t$ (which is a $d \times d$ matrix) and $\boldsymbol{X}_t^\top \boldsymbol{y}_t$ (which is a vector in $\mathbb{R}^d$) in memory, and updating them as above, we do not need to keep the whole data in memory.

2. One has to compute $(\sigma_n^{-2}\boldsymbol{X}_t^\top \boldsymbol{X}_t + \sigma_p^{-2}\boldsymbol{I})^{-1}$ for finding $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ in every round. We can write

$$\begin{aligned} (\sigma_n^{-2}\boldsymbol{X}_{t+1}^\top \boldsymbol{X}_{t+1} + \sigma_p^{-2}\boldsymbol{I})^{-1} &= \sigma_n^2(\boldsymbol{X}_{t+1}^\top \boldsymbol{X}_{t+1} + \sigma_n^2\sigma_p^{-2}\boldsymbol{I})^{-1} \\ &= \sigma_n^2(\underbrace{\boldsymbol{X}_t^\top \boldsymbol{X}_t + \sigma_n^2\sigma_p^{-2}\boldsymbol{I}}_{\boldsymbol{A}_t} + \boldsymbol{x}_{t+1}\boldsymbol{x}_{t+1}^\top)^{-1} \end{aligned}$$

where $\boldsymbol{A}_t \in \mathbb{R}^{d\times d}$. Using the Woodbury matrix identity (A.2) and that we know the inverse of $\boldsymbol{A}_t$ (from the previous iteration), the computation of the inverse of $(\boldsymbol{A}_t + \boldsymbol{x}_{t+1}\boldsymbol{x}_{t+1}^\top)$ is of $\mathcal{O}(d^2)$, which is much better than computing the inverse of $(\boldsymbol{A}_t + \boldsymbol{x}_{t+1}\boldsymbol{x}_{t+1}^\top)$ from scratch.

Kalman Filters

Solution to exercise 3.6 (Kalman update). Recall from eq. (3.10) that

$$p(x_{t+1} \mid y_{1:t+1}) = \frac{1}{Z} p(x_{t+1} \mid y_{1:t}) p(y_{t+1} \mid x_{t+1}). \tag{B.2}$$

Using the sensor model (3.14b),

$$p(y_{t+1} \mid x_{t+1}) = \frac{1}{Z'} \exp\left(-\frac{1}{2}\frac{(y_{t+1} - x_{t+1})^2}{\sigma_y^2}\right).$$

It remains to compute the predictive distribution,

$$\begin{aligned} p(x_{t+1} \mid y_{1:t}) &= \int p(x_{t+1} \mid x_t) p(x_t \mid y_{1:t})\, dx_t && \text{using eq. (3.11)} \\ &= \frac{1}{Z''} \int \exp\left(-\frac{1}{2}\left[\frac{(x_{t+1} - x_t)^2}{\sigma_x^2} + \frac{(x_t - \mu_t)^2}{\sigma_t^2}\right]\right) dx_t && \text{using the motion model (3.14a) and previous update} \\ &= \frac{1}{Z''} \int \exp\left(-\frac{1}{2}\left[\frac{\sigma_t^2(x_{t+1} - x_t)^2 + \sigma_x^2(x_t - \mu_t)^2}{\sigma_t^2\sigma_x^2}\right]\right) dx_t. \end{aligned}$$

The exponent is the sum of two expressions that are quadratic in x_t. *Completing the square* allows rewriting any quadratic $ax^2 + bx + c$ as the sum of a squared term $a(x + \frac{b}{2a})^2$ and a residual term $c - \frac{b^2}{4a}$ that is independent of x. In this case, we have $a = (\sigma_t^2 + \sigma_x^2)/(\sigma_t^2\sigma_x^2)$, $b = -2(\sigma_t^2 x_{t+1} + \sigma_x^2\mu_t)/(\sigma_t^2\sigma_x^2)$, and $c = (\sigma_t^2 x_{t+1}^2 + \sigma_x^2\mu_t^2)/(\sigma_t^2\sigma_x^2)$. The residual term can be taken outside the integral, giving

$$= \frac{1}{Z''} \exp\left(-\frac{1}{2}\left[c - \frac{b^2}{4a}\right]\right) \int \exp\left(-\frac{a}{2}\left[x_t + \frac{b}{2a}\right]^2\right) dx_t.$$

The integral is simply the integral of a Gaussian over its entire support, and thus evaluates to 1. We are therefore left with only the residual term from the quadratic. Plugging back in the expressions for a, b, and c and simplifying, we obtain

$$= \frac{1}{Z''} \exp\left(-\frac{1}{2}\frac{(x_{t+1} - \mu_t)^2}{\sigma_t^2 + \sigma_x^2}\right).$$

That is, $X_{t+1} \mid y_{1:t} \sim \mathcal{N}(\mu_t, \sigma_t^2 + \sigma_x^2)$.

Plugging our results back into eq. (B.2), we obtain

$$p(x_{t+1} \mid y_{1:t+1}) = \frac{1}{Z'''} \exp\left(-\frac{1}{2}\left[\frac{(x_{t+1} - \mu_t)^2}{\sigma_t^2 + \sigma_x^2} + \frac{(y_{t+1} - x_{t+1})^2}{\sigma_y^2}\right]\right).$$

Completing the square analogously to our derivation of the predictive distribution yields,

$$= \frac{1}{Z'''} \exp\left(-\frac{1}{2}\frac{\left(x_{t+1} - \frac{(\sigma_t^2+\sigma_x^2)y_{t+1} + \sigma_y^2\mu_t}{\sigma_t^2+\sigma_x^2+\sigma_y^2}\right)^2}{\frac{(\sigma_t^2+\sigma_x^2)\sigma_y^2}{\sigma_t^2+\sigma_x^2+\sigma_y^2}}\right).$$

Hence, $X_{t+1} \mid y_{t+1} \sim \mathcal{N}(\mu_{t+1}, \sigma_{t+1}^2)$ as defined in eq. (3.15).

Solution to exercise 3.7 (Parameter estimation using Kalman filters).

1. The given parameter estimation problem can be formulated as a Kalman filter in the following way:

$$\begin{aligned} x_t &= \pi && \forall t, \\ y_t &= x_t + \eta_t && \eta_t \sim \mathcal{N}(0, \sigma_y^2). \end{aligned}$$

   Thus, in terms of Kalman filters, this yields $f = h = 1, \epsilon_t = \sigma_x^2 = 0$. Using eq. (3.16), the *Kalman gain* is given by

$$k_{t+1} = \frac{\sigma_t^2}{\sigma_t^2 + \sigma_y^2},$$

whereas the *variance of the estimation error* σ_t^2 satisfies

$$\sigma_{t+1}^2 = \sigma_y^2 k_{t+1} = \frac{\sigma_t^2 \sigma_y^2}{\sigma_t^2 + \sigma_y^2}. \qquad \text{using eq. (3.19)}$$

To get the closed form, observe that

$$\frac{1}{\sigma_{t+1}^2} = \frac{1}{\sigma_t^2} + \frac{1}{\sigma_y^2} = \frac{1}{\sigma_{t-1}^2} + \frac{2}{\sigma_y^2} = \cdots = \frac{1}{\sigma_0^2} + \frac{t+1}{\sigma_y^2},$$

yielding,

$$\sigma_{t+1}^2 = \frac{\sigma_0^2 \sigma_y^2}{(t+1)\sigma_0^2 + \sigma_y^2} \quad \text{and} \quad k_{t+1} = \frac{\sigma_0^2}{(t+1)\sigma_0^2 + \sigma_y^2}.$$

2. When $t \to \infty$, we get $k_{t+1} \to 0$ and $\sigma_{t+1}^2 \to 0$, giving

$$\mu_{t+1} = \mu_t + k_{t+1}(y_{t+1} - \mu_t) = \mu_t, \qquad \text{using eq. (3.18)}$$

thus resulting in a stationary sequence.
3. Observe that $\sigma_0^2 \to \infty$ implies $k_{t+1} = \frac{1}{t+1}$. Therefore,

$$\begin{aligned}
\mu_{t+1} &= \mu_t + \frac{1}{t+1}(y_{t+1} - \mu_t) && \text{using eq. (3.18)} \\
&= \frac{t}{t+1}\mu_t + \frac{y_{t+1}}{t+1} \\
&= \frac{t}{t+1}\left(\frac{t-1}{t}\mu_{t-1} + \frac{y_t}{t}\right) + \frac{y_{t+1}}{t+1} && \text{using eq. (3.18)} \\
&= \frac{t-1}{t+1}\mu_{t-1} + \frac{y_t + y_{t+1}}{t+1} \\
&\vdots \\
&= \frac{y_1 + \cdots + y_{t+1}}{t+1},
\end{aligned}$$

which is simply the *sample mean*.

Gaussian Processes

Solution to exercise 4.4 (Feature space of Gaussian kernel).

1. We have for any $j \in \mathbb{N}_0$ and $x, y \in \mathbb{R}$,

$$\frac{1}{\sqrt{j!}} e^{-\frac{x^2}{2}} x^j \cdot \frac{1}{\sqrt{j!}} e^{-\frac{y^2}{2}} y^j = e^{-\frac{x^2+y^2}{2}} \frac{(xy)^j}{j!}.$$

Summing over all j, we obtain

$$\boldsymbol{\phi}(x)^\top \boldsymbol{\phi}(y) = \sum_{j=0}^{\infty} \phi_j(x)\phi_j(y)$$

$$
\begin{aligned}
&= e^{-\frac{x^2+y^2}{2}} \sum_{j=0}^{\infty} \frac{(xy)^j}{j!} \\
&= e^{-\frac{x^2+y^2}{2}} e^{xy} && \text{using the Taylor series expansion for the exponential function} \\
&= e^{-\frac{(x-y)^2}{2}} \\
&= k(x, y).
\end{aligned}
$$

2. As we have seen in section 2.6, the effective dimension is n. The crucial difference of kernelized regression (e.g., Gaussian processes) to linear regression is that the effective dimension *grows* with the sample size, whereas it is fixed for linear regression. Models where the effective dimension may depend on the sample size are called *non-parametric* models, and models where the effective dimension is fixed are called *parametric* models.

Solution to exercise 4.5 (A Kalman filter as a Gaussian process). First we look at the mean,

$$\mu(t) = \mathbb{E}[X_t] = \mathbb{E}[X_{t-1} + \epsilon_{t-1}] = \mathbb{E}[X_{t-1}] = \mu(t-1).$$

Knowing that $\mu(0) = 0$, we can derive that $\mu(t) = 0$ $(\forall t)$.

Now we look at the variance of X_t,

$$\mathrm{Var}[X_t] = \mathrm{Var}\left[X_0 + \sum_{\tau=0}^{t-1} \epsilon_\tau\right] = \sigma_0^2 + t\sigma_x^2. \qquad \text{using that the noise is independent}$$

Finally, we look at the distribution of $[f(t)\ f(t')]^\top$ for arbitrary $t \leq t'$.

$$\begin{bmatrix} X_t \\ X_{t'} \end{bmatrix} = \begin{bmatrix} X_t \\ X_t \end{bmatrix} + \sum_{\tau=t}^{t'-1} \begin{bmatrix} 0 \\ \epsilon_\tau \end{bmatrix}.$$

Therefore, we get that

$$
\begin{aligned}
\begin{bmatrix} X_t \\ X_{t'} \end{bmatrix} &\sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mathrm{Var}[X_t] \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + (t'-t) \begin{bmatrix} 0 & 0 \\ 0 & \sigma_x^2 \end{bmatrix}\right) \\
&= \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathrm{Var}[X_t] & \mathrm{Var}[X_t] \\ \mathrm{Var}[X_t] & \mathrm{Var}[X_t] + (t'-t)\sigma_x^2 \end{bmatrix}\right).
\end{aligned}
$$

We take the kernel $k_{\mathrm{KF}}(t, t')$ to be the covariance between $f(t)$ and $f(t')$, which is $\mathrm{Var}[X_t] = \sigma_0^2 + \sigma_x^2 t$. Notice, however, that we assumed $t \leq t'$. Thus, overall, the kernel is described by

$$k_{\mathrm{KF}}(t, t') = \sigma_0^2 + \sigma_x^2 \min\{t, t'\}.$$

Solution to exercise 4.9 (Reproducing property and RKHS norm).

1. As $f \in \mathcal{H}_k(\mathcal{X})$ we can express $f(\boldsymbol{x})$ for some $\beta_i \in \mathbb{R}$ and $\boldsymbol{x}_i \in \mathcal{X}$ as

$$\begin{aligned} f(\boldsymbol{x}) &= \sum_{i=1}^{n} \beta_i k(\boldsymbol{x}, \boldsymbol{x}_i) \\ &= \sum_{i=1}^{n} \beta_i \left\langle k(\boldsymbol{x}_i, \cdot), k(\boldsymbol{x}, \cdot) \right\rangle_k \\ &= \left\langle \sum_{i=1}^{n} \beta_i k(\boldsymbol{x}_i, \cdot), k(\boldsymbol{x}, \cdot) \right\rangle_k \\ &= \left\langle f(\cdot), k(\boldsymbol{x}, \cdot) \right\rangle_k . \end{aligned}$$

2. By applying Cauchy-Schwarz,

$$\begin{aligned} |f(\boldsymbol{x}) - f(\boldsymbol{y})| &= |\left\langle f, k(\boldsymbol{x}, \cdot) - k(\boldsymbol{y}, \cdot) \right\rangle_k| \\ &\leq \|f\|_k \, \|k(\boldsymbol{x}, \cdot) - k(\boldsymbol{y}, \cdot)\|_k \end{aligned}$$

Solution to exercise 4.11 (MAP estimate of Gaussian processes).

1. By the representer theorem, $\hat{f}(\boldsymbol{x}) = \hat{\boldsymbol{\alpha}}^\top \boldsymbol{k}_{\boldsymbol{x},A}$. In particular, we have $\boldsymbol{f} = \boldsymbol{K}\hat{\boldsymbol{\alpha}}$ and therefore

$$\begin{aligned} -\log p(y_{1:n} \mid \boldsymbol{x}_{1:n}, f) &= \frac{1}{2\sigma_n^2} \|\boldsymbol{y} - \boldsymbol{f}\|_2^2 + \text{const} \\ &= \frac{1}{2\sigma_n^2} \|\boldsymbol{y} - \boldsymbol{K}\hat{\boldsymbol{\alpha}}\|_2^2 + \text{const}. \end{aligned}$$

   The regularization term simplifies to

$$\|f\|_k^2 = \langle f, f \rangle_k = \hat{\boldsymbol{\alpha}}^\top \boldsymbol{K} \hat{\boldsymbol{\alpha}} = \|\hat{\boldsymbol{\alpha}}\|_{\boldsymbol{K}}^2 .$$

   Combining, we have that

$$\hat{\boldsymbol{\alpha}} = \operatorname*{arg\,min}_{\boldsymbol{\alpha} \in \mathbb{R}^n} \frac{1}{2\sigma_n^2} \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha}\|_2^2 + \frac{1}{2} \|\boldsymbol{\alpha}\|_{\boldsymbol{K}}^2$$

   as desired. It follows by multiplying through with $2\sigma_n^2$ that $\lambda = \sigma_n^2$.
2. Expanding the objective determined in (1), we are looking for the minimizer of

$$\boldsymbol{\alpha}^\top (\sigma_n^2 \boldsymbol{K} + \boldsymbol{K}^2)\boldsymbol{\alpha} - 2\boldsymbol{y}^\top \boldsymbol{K}\boldsymbol{\alpha} + \boldsymbol{y}^\top \boldsymbol{y}.$$

   Differentiating with respect to the coefficients $\boldsymbol{\alpha}$, we obtain the minimizer $\hat{\boldsymbol{\alpha}} = (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1}\boldsymbol{y}$. Thus, the prediction at a point $\boldsymbol{x}^\star$ is $\boldsymbol{k}_{\boldsymbol{x}^\star,A}^\top (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1}\boldsymbol{y}$ which coincides with the MAP estimate.

Solution to exercise 4.13 (Gradient of the marginal likelihood).

1. Applying the two hints eqs. (4.39) and (4.40) to eq. (4.37) yields,

$$\frac{\partial}{\partial \theta_j} \log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) = \frac{1}{2} \boldsymbol{y}^\top \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}^{-1} \frac{\partial \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}}{\partial \theta_j} \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}^{-1} \boldsymbol{y} - \frac{1}{2} \operatorname{tr}\left(\boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}^{-1} \frac{\partial \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}}{\partial \theta_j} \right).$$

We can simplify to

$$= \frac{1}{2}\mathrm{tr}\left(\boldsymbol{y}^\top \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}^{-1}\frac{\partial \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}}{\partial \theta_j}\boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}^{-1}\boldsymbol{y}\right) - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}^{-1}\frac{\partial \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}}{\partial \theta_j}\right) \quad \text{using that } \boldsymbol{y}^\top \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}^{-1}\tfrac{\partial \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}}{\partial \theta_j}\boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}^{-1}\boldsymbol{y} \text{ is a scalar}$$

$$= \frac{1}{2}\mathrm{tr}\left(\boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}^{-1}\boldsymbol{y}\boldsymbol{y}^\top \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}^{-1}\frac{\partial \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}}{\partial \theta_j} - \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}^{-1}\frac{\partial \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}}{\partial \theta_j}\right) \quad \text{using the cyclic property and linearity of the trace}$$

$$= \frac{1}{2}\mathrm{tr}\left(\boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}^{-1}\boldsymbol{y}(\boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}^{-1}\boldsymbol{y})^\top \frac{\partial \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}}{\partial \theta_j} - \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}^{-1}\frac{\partial \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}}{\partial \theta_j}\right) \quad \text{using that } \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}^{-1} \text{ is symmetric}$$

$$= \frac{1}{2}\mathrm{tr}\left((\alpha\alpha^\top - \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}^{-1})\frac{\partial \boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}}}{\partial \theta_j}\right).$$

2. We denote by $\tilde{\boldsymbol{K}}$ the covariance matrix of $\boldsymbol{y}$ for the covariance function $\tilde{k}$, so $\boldsymbol{K}_{\boldsymbol{y},\boldsymbol{\theta}} = \theta_0\tilde{\boldsymbol{K}}$. Then,

$$\frac{\partial}{\partial \theta_0}\log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) = \frac{1}{2}\mathrm{tr}\Big((\theta_0^{-2}\tilde{\boldsymbol{K}}^{-1}\boldsymbol{y}(\tilde{\boldsymbol{K}}^{-1}\boldsymbol{y})^\top - \theta_0^{-1}\tilde{\boldsymbol{K}}^{-1})\tilde{\boldsymbol{K}}\Big). \quad \text{using eq. (4.38)}$$

Simplifying the terms and using linearity of the trace, we obtain that

$$\frac{\partial}{\partial \theta_0}\log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) = 0 \quad \iff \quad \theta_0 = \frac{1}{n}\mathrm{tr}\Big(\boldsymbol{y}\boldsymbol{y}^\top \tilde{\boldsymbol{K}}^{-1}\Big).$$

If we define $\tilde{\boldsymbol{\Lambda}} \doteq \tilde{\boldsymbol{K}}^{-1}$ as the precision matrix associated to $\boldsymbol{y}$ for the covariance function $\tilde{k}$, we can express $\theta_0^\star$ in closed form as

$$\theta_0^\star = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\tilde{\boldsymbol{\Lambda}}(i,j)y_i y_j. \tag{B.3}$$

3. We immediately see from eq. (B.3) that $\theta_0^\star$ scales by s^2 if $\boldsymbol{y}$ is scaled by s.

Solution to exercise 4.19 (Uniform convergence of Fourier features).

1. We have that $s(\cdot) \in [-\sqrt{2}, \sqrt{2}]$, and hence, $s(\boldsymbol{\Delta}_i)$ is $\sqrt{2}$-sub-Gaussian.[1] It then follows from Hoeffding's inequality (1.88) that [1] see example 1.47

$$\mathbb{P}(|f(\boldsymbol{\Delta}_i)| \geq \epsilon) \leq 2\exp\left(-\frac{m\epsilon^2}{4}\right).$$

2. We can apply Markov's inequality (1.82) to obtain

$$\mathbb{P}\Big(\|\boldsymbol{\nabla} f(\boldsymbol{\Delta}^\star)\|_2 \geq \frac{\epsilon}{2r}\Big) = \mathbb{P}\left(\|\boldsymbol{\nabla} f(\boldsymbol{\Delta}^\star)\|_2^2 \geq \left(\frac{\epsilon}{2r}\right)^2\right) \leq \left(\frac{2r\mathbb{E}\,\|\boldsymbol{\nabla} f(\boldsymbol{\Delta}^\star)\|_2}{\epsilon}\right)^2.$$

It remains to bound the expectation. We have

$$\mathbb{E}\,\|\boldsymbol{\nabla} f(\boldsymbol{\Delta}^\star)\|_2^2 = \mathbb{E}\,\|\boldsymbol{\nabla} s(\boldsymbol{\Delta}^\star) - \boldsymbol{\nabla} k(\boldsymbol{\Delta}^\star)\|_2^2$$

$$= \mathbb{E}\,\|\nabla s(\Delta^\star)\|_2^2 - 2\nabla k(\Delta^\star)^\top \mathbb{E}\nabla s(\Delta^\star) + \mathbb{E}\,\|\nabla k(\Delta^\star)\|_2^2. \quad \text{using linearity of expectation (1.24)}$$

Note that $\mathbb{E}\nabla s(\Delta) = \nabla k(\Delta)$ using eq. (1.28) and using that $s(\Delta)$ is an unbiased estimator of $k(\Delta)$. Therefore,

$$\begin{aligned}
&= \mathbb{E}\,\|\nabla s(\Delta^\star)\|_2^2 - \mathbb{E}\,\|\nabla k(\Delta^\star)\|_2^2 \\
&\leq \mathbb{E}\,\|\nabla s(\Delta^\star)\|_2^2 \\
&\leq \mathbb{E}_{\omega\sim p}\,\|\omega\|_2^2 = \sigma_p^2. && \text{using that } s \text{ is the cos of a linear function in } \omega
\end{aligned}$$

3. Using a union bound (1.1) and then the result of (1),

$$\mathbb{P}\left(\bigcup_{i=1}^{T} |f(\Delta_i)| \geq \frac{\epsilon}{2}\right) \leq \sum_{i=1}^{T} \mathbb{P}\left(|f(\Delta_i)| \geq \frac{\epsilon}{2}\right) \leq 2T \exp\left(-\frac{m\epsilon^2}{16}\right).$$

4. First, note that by contraposition,

$$\sup_{\Delta\in\mathcal{M}_\Delta} |f(\Delta)| \geq \epsilon \implies \exists i.\ |f(\Delta_i)| \geq \frac{\epsilon}{2} \text{ or } \|\nabla f(\Delta^\star)\|_2 \geq \frac{\epsilon}{2r},$$

and therefore,

$$\begin{aligned}
\mathbb{P}\left(\sup_{\Delta\in\mathcal{M}_\Delta} |f(\Delta)| \geq \epsilon\right) &\leq \mathbb{P}\left(\bigcup_{i=1}^{T} |f(\Delta_i)| \geq \frac{\epsilon}{2} \cup \|\nabla f(\Delta^\star)\|_2 \geq \frac{\epsilon}{2r}\right) \\
&\leq \mathbb{P}\left(\bigcup_{i=1}^{T} |f(\Delta_i)| \geq \frac{\epsilon}{2}\right) + \mathbb{P}\left(\|\nabla f(\Delta^\star)\|_2 \geq \frac{\epsilon}{2r}\right) && \text{using a union bound (1.1)} \\
&\leq 2T \exp\left(-\frac{m\epsilon^2}{2^4}\right) + \left(\frac{2r\sigma_p}{\epsilon}\right)^2 && \text{using the results from (2) and (3)} \\
&\leq \alpha r^{-d} + \beta r^2 && \text{using } T \leq (4\,\mathrm{diam}(\mathcal{M})/r)^d
\end{aligned}$$

with $\alpha = 2(4\,\mathrm{diam}(\mathcal{M}))^d \exp\left(-\frac{m\epsilon^2}{2^4}\right)$ and $\beta = \left(\frac{2\sigma_p}{\epsilon}\right)^2$. Using the hint, we obtain

$$\begin{aligned}
&= 2\beta^{\frac{d}{d+2}} \alpha^{\frac{2}{d+2}} \\
&= 2\left(2\left(\frac{2^3\sigma_p \mathrm{diag}(\mathcal{M})}{\epsilon}\right)^{2d}\right)^{\frac{1}{d+2}} \exp\left(-\frac{m\epsilon^2}{2^3(d+2)}\right) \\
&\leq 2^8\left(\frac{\sigma_p \mathrm{diag}(\mathcal{M})}{\epsilon}\right)^2 \exp\left(-\frac{m\epsilon^2}{2^3(d+2)}\right) && \text{using } \tfrac{\sigma_p \mathrm{diam}(\mathcal{M})}{\epsilon} \geq 1
\end{aligned}$$

5. We have

$$\begin{aligned}
\sigma_p^2 &= \mathbb{E}_{\omega\sim p}\left[\omega^\top \omega\right] \\
&= \int \omega^\top \omega \cdot p(\omega)\, d\omega \\
&= \int \omega^\top \omega \cdot e^{i\omega^\top \mathbf{0}} p(\omega)\, d\omega.
\end{aligned}$$

Now observe that $\frac{\partial^2}{\partial \Delta_j^2} \int e^{i\boldsymbol{\omega}^\top \boldsymbol{\Delta}} p(\boldsymbol{\omega})\, d\boldsymbol{\omega} = -\int \omega_j^2 e^{i\boldsymbol{\omega}^\top \boldsymbol{\Delta}} p(\boldsymbol{\omega})\, d\boldsymbol{\omega}$. Thus,

$$
\begin{aligned}
&= -\mathrm{tr}\left(\boldsymbol{H}_{\boldsymbol{\Delta}} \int p(\boldsymbol{\omega}) e^{i\boldsymbol{\omega}^\top \boldsymbol{\Delta}}\, d\boldsymbol{\omega} \Big|_{\boldsymbol{\Delta}=\mathbf{0}} \right) \\
&= -\mathrm{tr}(\boldsymbol{H}_{\boldsymbol{\Delta}} k(\mathbf{0})). && \text{using that } p \text{ is the Fourier transform of } k \text{ (4.48)}
\end{aligned}
$$

Finally, we have for the Gaussian kernel that

$$
\frac{\partial^2}{\partial \Delta_j^2} \exp\left(-\frac{\boldsymbol{\Delta}^\top \boldsymbol{\Delta}}{2\ell^2} \right) \Big|_{\Delta_j = 0} = -\frac{1}{\ell^2}.
$$

Solution to exercise 4.22 (Subset of regressors).

1. We write $\tilde{\boldsymbol{f}} \doteq [\boldsymbol{f}\ f^\star]^\top$. From the definition of SoR (4.59), we gather

$$
q_{\mathrm{SoR}}(\tilde{\boldsymbol{f}} \mid \boldsymbol{u}) = \mathcal{N}\left(\tilde{\boldsymbol{f}};\ \begin{bmatrix} \boldsymbol{K}_{AU} \boldsymbol{K}_{UU}^{-1} \\ \boldsymbol{K}_{\star U} \boldsymbol{K}_{UU}^{-1} \end{bmatrix} \boldsymbol{u}, \mathbf{0} \right). \tag{B.4}
$$

   We know that $\tilde{\boldsymbol{f}}$ and $\boldsymbol{u}$ are jointly Gaussian, and hence, the marginal distribution of $\tilde{\boldsymbol{f}}$ is also Gaussian. We have for the mean and variance that

$$
\begin{aligned}
\mathbb{E}[\tilde{\boldsymbol{f}}] &= \mathbb{E}[\mathbb{E}[\tilde{\boldsymbol{f}} \mid \boldsymbol{u}]] && \text{using the tower rule (1.33)} \\
&= \mathbb{E}_{\boldsymbol{u}}\left[\begin{bmatrix} \boldsymbol{K}_{AU} \boldsymbol{K}_{UU}^{-1} \\ \boldsymbol{K}_{\star U} \boldsymbol{K}_{UU}^{-1} \end{bmatrix} \boldsymbol{u} \right] && \text{using eq. (B.4)} \\
&= \begin{bmatrix} \boldsymbol{K}_{AU} \boldsymbol{K}_{UU}^{-1} \\ \boldsymbol{K}_{\star U} \boldsymbol{K}_{UU}^{-1} \end{bmatrix} \mathbb{E}[\boldsymbol{u}] = \mathbf{0} && \text{using linearity of expectation (1.24)} \\
\mathrm{Var}[\tilde{\boldsymbol{f}}] &= \mathbb{E}\underbrace{\mathrm{Var}[\tilde{\boldsymbol{f}} \mid \boldsymbol{u}]}_{\mathbf{0}} + \mathrm{Var}\,\mathbb{E}[\tilde{\boldsymbol{f}} \mid \boldsymbol{u}] && \text{using the law of total variance (1.50)} \\
&= \mathrm{Var}_{\boldsymbol{u}}\left[\begin{bmatrix} \boldsymbol{K}_{AU} \boldsymbol{K}_{UU}^{-1} \\ \boldsymbol{K}_{\star U} \boldsymbol{K}_{UU}^{-1} \end{bmatrix} \boldsymbol{u} \right] && \text{using eq. (B.4)} \\
&= \begin{bmatrix} \boldsymbol{K}_{AU} \boldsymbol{K}_{UU}^{-1} \\ \boldsymbol{K}_{\star U} \boldsymbol{K}_{UU}^{-1} \end{bmatrix} \underbrace{\mathrm{Var}[\boldsymbol{u}]}_{\boldsymbol{K}_{UU}} \begin{bmatrix} \boldsymbol{K}_{AU} \boldsymbol{K}_{UU}^{-1} \\ \boldsymbol{K}_{\star U} \boldsymbol{K}_{UU}^{-1} \end{bmatrix}^\top && \text{using eq. (1.47)} \\
&= \begin{bmatrix} \boldsymbol{Q}_{AA} & \boldsymbol{Q}_{A\star} \\ \boldsymbol{Q}_{\star A} & \boldsymbol{Q}_{\star\star} \end{bmatrix}.
\end{aligned}
$$

   Having determined $q_{\mathrm{SoR}}(\boldsymbol{f}, f^\star)$, $q_{\mathrm{SoR}}(f^\star \mid \boldsymbol{y})$ follows directly using the formulas for finding the Gaussian process predictive posterior (4.6).
2. The given covariance function follows directly from inspecting the derived covariance matrix, $\mathrm{Var}[\boldsymbol{f}] = \boldsymbol{Q}_{AA}$.

Variational Inference

Solution to exercise 5.3 (Logistic loss).

1. We have

$$\begin{aligned}
\nabla_w \ell_{\log}(w^\top x; y) &= \nabla_w \log(1 + \exp(-y w^\top x)) && \text{using the definition of the logistic loss (5.14)} \\
&= \nabla_w \log \frac{1 + \exp(y w^\top x)}{\exp(y w^\top x)} \\
&= \nabla_w \log(1 + \exp(y w^\top x)) - y w^\top x \\
&= \frac{1}{1 + \exp(-y w^\top x)} \cdot \exp(-y w^\top x) \cdot y x - y x && \text{using the chain rule} \\
&= -y x \cdot \left(1 - \frac{\exp(y w^\top x)}{1 + \exp(y w^\top x)}\right) \\
&= -y x \cdot \sigma(-y w^\top x). && \text{using the definition of the logistic function (5.10)}
\end{aligned}$$

2. As suggested in the hint, we compute the first derivative of σ,

$$\begin{aligned}
\sigma'(z) \doteq \frac{d}{dz}\sigma(z) &= \frac{d}{dz}\frac{1}{1 + \exp(-z)} && \text{using the definition of the logistic function (5.10)} \\
&= \frac{\exp(-z)}{(1 + \exp(-z))^2} && \text{using the quotient rule} \\
&= \sigma(z) \cdot (1 - \sigma(z)). && \text{using the definition of the logistic function (5.10)}
\end{aligned}$$

We get for the Hessian of $\ell_{\log}$,

$$\begin{aligned}
H_w \ell_{\log}(w^\top x; y) &= D_w \nabla_w \ell_{\log}(w^\top x; y) && \text{using the definition of a Hessian (1.106) and symmetry} \\
&= -y x \, D_w \sigma(-y w^\top x). && \text{using the gradient of the logistic loss from (1)}
\end{aligned}$$

We have $D_w - y w^\top x = -y x^\top$ and $D_z \sigma(z) = \sigma'(z)$, and therefore, using the chain rule of multivariate calculus (5.20),

$$\begin{aligned}
&= -y x \cdot \sigma'(-y w^\top x) \cdot (-y x^\top) \\
&= x x^\top \cdot \sigma'(-y w^\top x). && \text{using } y^2 = 1
\end{aligned}$$

Finally, recall that

$$\begin{aligned}
\sigma(w^\top x) &= \mathbb{P}(y = +1 \mid x, w) \quad \text{and} \\
\sigma(-w^\top x) &= \mathbb{P}(y = -1 \mid x, w).
\end{aligned}$$

Thus,

$$\begin{aligned}
\sigma'(-y w^\top x) &= \mathbb{P}(Y \neq y \mid x, w) \cdot (1 - \mathbb{P}(Y \neq y \mid x, w)) \\
&= \mathbb{P}(Y \neq y \mid x, w) \cdot \mathbb{P}(Y = y \mid x, w) \\
&= (1 - \mathbb{P}(Y = y \mid x, w)) \cdot \mathbb{P}(Y = y \mid x, w) \\
&= \sigma'(w^\top x).
\end{aligned}$$

3. By the second-order characterization of convexity (cf. remark 1.57), a twice differentiable function f is convex if and only if its Hessian is positive semi-definite. Writing $c \doteq \sigma'(\boldsymbol{w}^\top \boldsymbol{x})$, we have for any $\boldsymbol{\delta} \in \mathbb{R}^n$ that

$$\boldsymbol{\delta}^\top \boldsymbol{H}_{\boldsymbol{w}} \ell_{\log}(\boldsymbol{w}^\top \boldsymbol{x}; y)\boldsymbol{\delta} = \boldsymbol{\delta}^\top \left(c \cdot \boldsymbol{x}\boldsymbol{x}^\top\right)\boldsymbol{\delta} = c(\boldsymbol{\delta}^\top \boldsymbol{x})^2 \geq 0, \qquad \text{using } c \geq 0 \text{ and } (\cdot)^2 \geq 0$$

and hence, the logistic loss is convex in $\boldsymbol{w}$.

Solution to exercise 5.5 (Gaussian process classification).

1. Using the law of total probability (1.15),

$$\begin{aligned} &p(y^\star = +1 \mid \boldsymbol{x}_{1:n}, y_{1:n}, \boldsymbol{x}^\star) \\ &= \int p(y^\star = +1 \mid f^\star) p(f^\star \mid \boldsymbol{x}_{1:n}, y_{1:n}, \boldsymbol{x}^\star)\, df^\star \\ &= \int \sigma(f^\star) p(f^\star \mid \boldsymbol{x}_{1:n}, y_{1:n}, \boldsymbol{x}^\star)\, df^\star. && \text{using } y^\star \sim \text{Bern}(\sigma(f^\star)) \end{aligned}$$

   Due to the non-Gaussian likelihood, the integral is analytically intractable. However, as the integral is one-dimensional, numerical approximations such as the Gauss-Legendre quadrature can be used.

2. (a) According to Bayes' rule (1.59), we know that

$$\begin{aligned} \psi(\boldsymbol{f}) &= \log p(\boldsymbol{f} \mid \boldsymbol{x}_{1:n}, y_{1:n}) && \text{using eq. (5.2)} \\ &= \log p(y_{1:n} \mid \boldsymbol{f}) + \log p(\boldsymbol{f} \mid \boldsymbol{x}_{1:n}) - \log p(y_{1:n} \mid \boldsymbol{x}_{1:n}) && \text{using Bayes' rule (1.59)} \\ &= \log p(y_{1:n} \mid \boldsymbol{f}) + \log p(\boldsymbol{f} \mid \boldsymbol{x}_{1:n}) + \text{const.} \end{aligned}$$

   Note that $p(\boldsymbol{f} \mid \boldsymbol{x}_{1:n}) = \mathcal{N}(\boldsymbol{f}; \boldsymbol{0}, \boldsymbol{K}_{AA})$. Plugging in this closed-form Gaussian distribution of the GP prior gives

$$= \log p(y_{1:n} \mid \boldsymbol{f}) - \frac{1}{2}\boldsymbol{f}^\top \boldsymbol{K}_{AA}^{-1}\boldsymbol{f} + \text{const}$$

   Differentiating with respect to $\boldsymbol{f}$ yields

$$\begin{aligned} \boldsymbol{\nabla}\psi(\boldsymbol{f}) &= \boldsymbol{\nabla} \log p(y_{1:n} \mid \boldsymbol{f}) - \boldsymbol{K}_{AA}^{-1}\boldsymbol{f} \\ \boldsymbol{H}\psi(\boldsymbol{f}) &= \boldsymbol{H}_{\boldsymbol{f}} \log p(y_{1:n} \mid \boldsymbol{f}) - \boldsymbol{K}_{AA}^{-1}. \end{aligned}$$

   Hence, $\boldsymbol{\Lambda} = \boldsymbol{K}_{AA}^{-1} + \boldsymbol{W}$ where $\boldsymbol{W} \doteq -\boldsymbol{H}_{\boldsymbol{f}} \log p(y_{1:n} \mid \boldsymbol{f})\big|_{\boldsymbol{f}=\hat{\boldsymbol{f}}}$.
   It remains to derive $\boldsymbol{W}$ for the probit likelihood. Using independence of the training examples,

$$\log p(y_{1:n} \mid \boldsymbol{f}) = \sum_{i=1}^{n} \log p(y_i \mid f_i),$$

   and hence, the Hessian of this expression is diagonal. Using the symmetry of $\Phi(z; 0, \sigma_n^2)$ around zero, we can write

$$\log p(y_i \mid f_i) = \log \Phi(y_i f_i; 0, \sigma_n^2).$$

In the following, we write $\mathcal{N}(z) \doteq \mathcal{N}(z; 0, \sigma_n^2)$ and $\Phi(z) \doteq \Phi(z; 0, \sigma_n^2)$ to simplify the notation. Differentiating with respect to f_i, we obtain

$$\frac{\partial}{\partial f_i} \log \Phi(y_i f_i) = \frac{y_i \mathcal{N}(f_i)}{\Phi(y_i f_i)}$$
using $\mathcal{N}(y_i f_i) = \mathcal{N}(f_i)$ since $\mathcal{N}$ is an even function and $y_i \in \{\pm 1\}$

$$\frac{\partial^2}{\partial f_i^2} \log \Phi(y_i f_i) = -\frac{\mathcal{N}(f_i)^2}{\Phi(y_i f_i)^2} - \frac{y_i f_i \mathcal{N}(f_i)}{\sigma_n^2 \Phi(y_i f_i)},$$

and $\boldsymbol{W} = -\operatorname{diag}\{\frac{\partial^2}{\partial f_i^2} \log \Phi(y_i f_i)\}_{i=1}^n \Big|_{f=\hat{f}}$.

(b) Note that $\boldsymbol{\Lambda}'$ is a precision matrix over weights $\boldsymbol{w}$ and $\boldsymbol{f} = \boldsymbol{X}\boldsymbol{w}$, so by eq. (1.47) the corresponding variance over latent values $\boldsymbol{f}$ is $\boldsymbol{X}\boldsymbol{\Lambda}'^{-1}\boldsymbol{X}^\top$. The two precision matrices are therefore equivalent if $\boldsymbol{\Lambda}^{-1} = \boldsymbol{X}\boldsymbol{\Lambda}'^{-1}\boldsymbol{X}^\top$.
Analogously to example 5.2 and exercise 5.3, we have that

$$\begin{aligned}\boldsymbol{W} &= -\boldsymbol{H}_{\boldsymbol{f}} \log p(\boldsymbol{y}_{1:n} \mid \boldsymbol{f})\big|_{\boldsymbol{f}=\hat{\boldsymbol{f}}} \\ &= \operatorname{diag}_{i\in[n]}\{\sigma(\hat{f}_i)(1-\sigma(\hat{f}_i))\} \\ &= \operatorname{diag}_{i\in[n]}\{\pi_i(1-\pi_i)\}.\end{aligned}$$

By Woodbury's matrix identity (A.1),

$$\begin{aligned}\boldsymbol{\Lambda}'^{-1} &= (\boldsymbol{I} + \boldsymbol{X}^\top \boldsymbol{W} \boldsymbol{X})^{-1} \\ &= \boldsymbol{I} - \boldsymbol{X}^\top (\boldsymbol{W}^{-1} + \boldsymbol{X}\boldsymbol{X}^\top)^{-1} \boldsymbol{X}.\end{aligned}$$

Thus,

$$\begin{aligned}\boldsymbol{X}\boldsymbol{\Lambda}'^{-1}\boldsymbol{X}^\top &= \boldsymbol{X}\boldsymbol{X}^\top - \boldsymbol{X}\boldsymbol{X}^\top (\boldsymbol{W}^{-1} + \boldsymbol{X}\boldsymbol{X}^\top)^{-1} \boldsymbol{X}\boldsymbol{X}^\top \\ &= \boldsymbol{K}_{AA} - \boldsymbol{K}_{AA}(\boldsymbol{W}^{-1} + \boldsymbol{K}_{AA})^{-1}\boldsymbol{K}_{AA} && \text{using that } \boldsymbol{K}_{AA} = \boldsymbol{X}\boldsymbol{X}^\top \\ &= (\boldsymbol{K}_{AA}^{-1} + \boldsymbol{W})^{-1} && \text{by the matrix inversion lemma (A.3)} \\ &= \boldsymbol{\Lambda}^{-1}.\end{aligned}$$

(c) Using the formulas for the conditional distribution of a Gaussian (1.118), evaluating a conditional GP at a test point $\boldsymbol{x}^\star$ yields,

$$f^\star \mid \boldsymbol{x}^\star, \boldsymbol{f} \sim \mathcal{N}(\mu^\star, k^\star), \quad \text{where} \tag{B.5a}$$
$$\mu^\star \doteq \boldsymbol{k}_{\boldsymbol{x}^\star, A}^\top \boldsymbol{K}_{AA}^{-1} \boldsymbol{f}, \tag{B.5b}$$
$$k^\star \doteq k(\boldsymbol{x}^\star, \boldsymbol{x}^\star) - \boldsymbol{k}_{\boldsymbol{x}^\star, A}^\top \boldsymbol{K}_{AA}^{-1} \boldsymbol{k}_{\boldsymbol{x}^\star, A}. \tag{B.5c}$$

Then, using the tower rule (1.33),

$$\begin{aligned}\mathbb{E}_q[f^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n}] &= \mathbb{E}_{f\sim q}\Big[\mathbb{E}_{f^\star \sim p}[f^\star \mid \boldsymbol{x}^\star, \boldsymbol{f}] \;\Big|\; \boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n}\Big] \\ &= \boldsymbol{k}_{\boldsymbol{x}^\star, A}^\top \boldsymbol{K}_{AA}^{-1} \mathbb{E}_q[\boldsymbol{f} \mid \boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n}] && \text{using eq. (B.5b) and linearity of expectation (1.24)}\end{aligned}$$

$$= \boldsymbol{k}_{\boldsymbol{x}^\star,A}^\top \boldsymbol{K}_{AA}^{-1} \hat{\boldsymbol{f}}. \tag{B.6}$$

Observe that any maximum $\hat{\boldsymbol{f}}$ of $\psi(\boldsymbol{f})$ needs to satisfy $\boldsymbol{\nabla}\psi(\boldsymbol{f}) = \boldsymbol{0}$. Hence, $\hat{\boldsymbol{f}} = \boldsymbol{K}_{AA}(\boldsymbol{\nabla}_{\boldsymbol{f}} \log p(y_{1:n} \mid \boldsymbol{f}))$, and the expectation simplifies to

$$= \boldsymbol{k}_{\boldsymbol{x}^\star,A}^\top (\boldsymbol{\nabla}_{\boldsymbol{f}} \log p(y_{1:n} \mid \boldsymbol{f})).$$

Using the law of total variance (1.50),

$$\begin{aligned}
&\operatorname{Var}_q[f^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, y_{1:n}] \\
&= \mathbb{E}_{f\sim q}\left[\operatorname{Var}_{f^\star\sim p}[f^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, \boldsymbol{f}] \;\middle|\; \boldsymbol{x}_{1:n}, y_{1:n}\right] \\
&\quad + \operatorname{Var}_{f\sim q}\left[\mathbb{E}_{f^\star\sim p}[f^\star \mid \boldsymbol{x}^\star, \boldsymbol{x}_{1:n}, \boldsymbol{f}] \;\middle|\; \boldsymbol{x}_{1:n}, y_{1:n}\right] \\
&= \mathbb{E}_q[k^\star \mid \boldsymbol{x}_{1:n}, y_{1:n}] + \operatorname{Var}_q\left[\boldsymbol{k}_{\boldsymbol{x}^\star,A}^\top \boldsymbol{K}_{AA}^{-1} \boldsymbol{f} \;\middle|\; \boldsymbol{x}_{1:n}, y_{1:n}\right] && \text{using eqs. (B.5b) and (B.5c)} \\
&= k^\star + \boldsymbol{k}_{\boldsymbol{x}^\star,A}^\top \boldsymbol{K}_{AA}^{-1} \operatorname{Var}_q[\boldsymbol{f} \mid \boldsymbol{x}_{1:n}, y_{1:n}] \, \boldsymbol{K}_{AA}^{-1} \boldsymbol{k}_{\boldsymbol{x}^\star,A}. && \text{using that } k^\star \text{ is independent of } \boldsymbol{f}, \text{ eq. (1.47), and symmetry of } \boldsymbol{K}_{AA}^{-1}
\end{aligned}$$

Recall from (a) that $\operatorname{Var}_q[\boldsymbol{f} \mid \boldsymbol{x}_{1:n}, y_{1:n}] = (\boldsymbol{K}_{AA} + \boldsymbol{W}^{-1})^{-1}$, so

$$\begin{aligned}
&= k(\boldsymbol{x}^\star, \boldsymbol{x}^\star) - \boldsymbol{k}_{\boldsymbol{x}^\star,A}^\top \boldsymbol{K}_{AA}^{-1} \boldsymbol{k}_{\boldsymbol{x}^\star,A} && \text{plugging in the expression for } k^\star \text{ (B.5c)} \\
&\quad + \boldsymbol{k}_{\boldsymbol{x}^\star,A}^\top \boldsymbol{K}_{AA}^{-1} (\boldsymbol{K}_{AA} + \boldsymbol{W}^{-1})^{-1} \boldsymbol{K}_{AA}^{-1} \boldsymbol{k}_{\boldsymbol{x}^\star,A} \\
&= k(\boldsymbol{x}^\star, \boldsymbol{x}^\star) - \boldsymbol{k}_{\boldsymbol{x}^\star,A}^\top (\boldsymbol{K}_{AA} + \boldsymbol{W}^{-1})^{-1} \boldsymbol{k}_{\boldsymbol{x}^\star,A}. && \text{using the matrix inversion lemma (A.3)}
\end{aligned} \tag{B.7}$$

Notice the similarity of eqs. (B.6) and (B.7) to eqs. (B.5b) and (B.5c). The latter is the conditional mean and variance if $\boldsymbol{f}$ is known whereas the former is the conditional mean and variance given the noisy observation $y_{1:n}$ of $\boldsymbol{f}$. The matrix $\boldsymbol{W}$ quantifies the noise in the observations.

(d) Recall that

$$\begin{aligned}
p(y^\star = +1 \mid \boldsymbol{x}_{1:n}, y_{1:n}, \boldsymbol{x}^\star) &\approx \int \sigma(f^\star) q(f^\star \mid \boldsymbol{x}_{1:n}, y_{1:n}, \boldsymbol{x}^\star) \, df^\star && \text{using the Laplace-approximated latent predictive posterior} \\
&= \mathbb{E}_q[\sigma(f^\star)]. && \text{using LOTUS (1.30)}
\end{aligned}$$

This quantity can be interpreted as the *averaged prediction* over all latent predictions $f^\star$. In contrast, $\sigma(\mathbb{E}_q[f^\star])$ can be understood as the *"MAP" prediction*, which is obtained using the MAP estimate of $f^\star$.[2] As σ is non-linear, the two quantities are not identical, and generally the averaged prediction is preferred.

[2] As q is a Gaussian, its mode (i.e., the MAP estimate) and its mean coincide.

3. We have

$$\begin{aligned}
\mathbb{E}_\epsilon \mathbb{1}\{f(\boldsymbol{x}) + \epsilon \geq 0\} &= \mathbb{P}_\epsilon(f(\boldsymbol{x}) + \epsilon \geq 0) && \text{using } \mathbb{E}[X] = p \text{ if } X \sim \operatorname{Bern}(p) \\
&= \mathbb{P}_\epsilon(-\epsilon \leq f(\boldsymbol{x})) \\
&= \mathbb{P}_\epsilon(\epsilon \leq f(\boldsymbol{x})) && \text{using that the distribution of } \epsilon \text{ is symmetric around } 0 \\
&= \Phi(f(\boldsymbol{x}); 0, \sigma_n^2).
\end{aligned}$$

Solution to exercise 5.17 (Binary cross-entropy loss). If $y = 1$ then

$$\ell_{\text{bce}}(\hat{y}; y) = -\log \hat{y} = \log(1 + e^{-\hat{f}}) = \ell_{\log}(\hat{F}; y).$$

If $y = -1$ then

$$\ell_{\text{bce}}(\hat{y}; y) = -\log(1 - \hat{y}) = \log(1 + e^{\hat{f}}) = \ell_{\log}(\hat{f}; y).$$

Here the second equality follows from the simple algebraic fact

$$1 - \frac{1}{1 + e^{-z}} = 1 - \frac{e^z}{e^z + 1} = \frac{1}{1 + e^z}. \qquad \text{multiplying by } \tfrac{e^z}{e^z}$$

Solution to exercise 5.13 (Jensen's inequality).

1. Recall that as f is convex,

   $$\forall \boldsymbol{x}_1, \boldsymbol{x}_2, \forall \lambda \in [0,1]: \quad f(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2) \leq \lambda f(\boldsymbol{x}_1) + (1-\lambda) f(\boldsymbol{x}_2).$$

   We prove the statement by induction on k. The base case, $k = 2$, follows trivially from the convexity of f. For the induction step, suppose that the statement holds for some fixed $k \geq 2$ and assume w.l.o.g. that $\theta_{k+1} \in (0,1)$. We then have,

   $$\begin{aligned}
   \sum_{i=1}^{k+1} \theta_i f(\boldsymbol{x}_i) &= (1 - \theta_{k+1}) \left(\sum_{i=1}^{k} \frac{\theta_i}{1 - \theta_{k+1}} f(\boldsymbol{x}_i) \right) + \theta_{k+1} f(\boldsymbol{x}_{k+1}) \\
   &\geq (1 - \theta_{k+1}) \cdot f\left(\sum_{i=1}^{k} \frac{\theta_i}{1 - \theta_{k+1}} \boldsymbol{x}_i \right) + \theta_{k+1} f(\boldsymbol{x}_{k+1}) && \text{using the induction hypothesis} \\
   &\geq f\left(\sum_{i=1}^{k+1} \theta_i \boldsymbol{x}_i \right). && \text{using the convexity of } f
   \end{aligned}$$

2. Noting that $\log_2$ is concave, we have by Jensen's inequality,

   $$\begin{aligned}
   \mathrm{H}[p] &= \mathbb{E}_{x \sim p}\left[\log_2 \left(\frac{1}{p(x)} \right) \right] && \text{by definition of entropy (5.33a)} \\
   &\leq \log_2 \mathbb{E}_{x \sim p}\left[\frac{1}{p(x)} \right] && \text{by Jensen's inequality (5.38)} \\
   &= \log_2 n.
   \end{aligned}$$

Solution to exercise 5.19 (Gibbs' inequality).

1. Let p and q be two (continuous) distributions. The KL-divergence between p and q is

   $$\begin{aligned}
   \mathrm{KL}(p \| q) &= \mathbb{E}_{\boldsymbol{x} \sim p}\left[\log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \right] && \text{using the definition of KL-divergence (5.44)} \\
   &= \mathbb{E}_{\boldsymbol{x} \sim p}\left[\mathrm{S}\left[\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})} \right] \right].
   \end{aligned}$$

Note that the surprise $\mathrm{S}[u] = -\log u$ is a convex function in u, and hence,

$$\begin{aligned}
&\geq \mathrm{S}\left[\mathbb{E}_{\boldsymbol{x}\sim p}\left[\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right]\right] && \text{using Jensen's inequality (5.37)}\\
&= \mathrm{S}\left[\int q(\boldsymbol{x})\, d\boldsymbol{x}\right]\\
&= \mathrm{S}[1] = 0. && \text{a probability density integrates to 1}
\end{aligned}$$

2. We observe from the derivation of (1) that $\mathrm{KL}(p\|q) = 0$ iff equality holds for Jensen's inequality. Now, if p and q are discrete with final and identical support, we can follow from the hint that Jensen's inequality degenerates to an equality iff p and q are point wise identical.

Solution to exercise 5.20 (Maximum entropy principle).

1. We have

$$\begin{aligned}
\mathrm{H}[f\|g] &= -\int_{\mathbb{R}} f(x)\log g(x)\, dx && \text{using the definition of cross-entropy (5.40)}\\
&= -\int_{\mathbb{R}} f(x)\cdot\left(\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{(x-\mu)^2}{2\sigma^2}\right) dx && \text{using that } g(x) = \mathcal{N}(x;\mu,\sigma^2)\\
&= -\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)\underbrace{\int_{\mathbb{R}} f(x)\, dx}_{1} + \frac{1}{2\sigma^2}\int_{\mathbb{R}} f(x)(x-\mu)^2\, dx\\
&= \log(\sigma\sqrt{2\pi}) + \frac{1}{2\sigma^2}\underbrace{\mathbb{E}_{x\sim f}\left[(x-\mu)^2\right]}_{\sigma^2}\\
&= \log(\sigma\sqrt{2\pi}) + \frac{1}{2}\\
&= \mathrm{H}[g]. && \text{using the entropy of Gaussians (5.35)}
\end{aligned}$$

2. We have shown that

$$\mathrm{H}[g] - \mathrm{H}[f] = \mathrm{KL}(f\|g) \geq 0,$$

and hence, $\mathrm{H}[g] \geq \mathrm{H}[f]$. That is, for a fixed mean μ and variance σ^2, the distribution that has maximum entropy among all distributions on $\mathbb{R}$ is the normal distribution.

Solution to exercise 5.22 (KL-divergence of Gaussians). We can rewrite the KL-divergence as

$$\begin{aligned}
\mathrm{KL}(p\|q) &= \mathbb{E}_{\boldsymbol{x}\sim p}[\log p(\boldsymbol{x}) - \log q(\boldsymbol{x})] && \text{using the definition of KL-divergence (5.44)}\\
&= \mathbb{E}_{\boldsymbol{x}\sim p}\left[\frac{1}{2}\log\frac{\det(\boldsymbol{\Sigma}_q)}{\det(\boldsymbol{\Sigma}_p)} - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_p)^\top\boldsymbol{\Sigma}_p^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_p)\right. && \text{using the Gaussian PDF (1.115)}\\
&\qquad\left. + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_q)^\top\boldsymbol{\Sigma}_q^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_q)\right]
\end{aligned}$$

$$= \frac{1}{2}\log\frac{\det(\boldsymbol{\Sigma}_q)}{\det(\boldsymbol{\Sigma}_p)} - \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p}\Big[(\boldsymbol{x}-\boldsymbol{\mu}_p)^\top \boldsymbol{\Sigma}_p^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_p)\Big] \qquad \text{using linearity of expectation (1.24)}$$
$$+ \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p}\Big[(\boldsymbol{x}-\boldsymbol{\mu}_q)^\top \boldsymbol{\Sigma}_q^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_q)\Big]$$

As $(\boldsymbol{x}-\boldsymbol{\mu}_p)^\top \boldsymbol{\Sigma}_p^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_p) \in \mathbb{R}$, we can rewrite the second term as

$$\begin{aligned}
&\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p}\Big[\mathrm{tr}\Big((\boldsymbol{x}-\boldsymbol{\mu}_p)^\top \boldsymbol{\Sigma}_p^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_p)\Big)\Big] \\
&= \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p}\Big[\mathrm{tr}\Big((\boldsymbol{x}-\boldsymbol{\mu}_p)(\boldsymbol{x}-\boldsymbol{\mu}_p)^\top \boldsymbol{\Sigma}_p^{-1}\Big)\Big] && \text{using the cyclic property of the trace} \\
&= \frac{1}{2}\mathrm{tr}\Big(\mathbb{E}_{\boldsymbol{x}\sim p}\Big[(\boldsymbol{x}-\boldsymbol{\mu}_p)(\boldsymbol{x}-\boldsymbol{\mu}_p)^\top\Big]\boldsymbol{\Sigma}_p^{-1}\Big) && \text{using linearity of the trace and linearity of expectation (1.24)} \\
&= \frac{1}{2}\mathrm{tr}\Big(\boldsymbol{\Sigma}_p\boldsymbol{\Sigma}_p^{-1}\Big) && \text{using the definition of the covariance matrix (1.44)} \\
&= \frac{1}{2}\mathrm{tr}(\boldsymbol{I}) = \frac{d}{2}.
\end{aligned}$$

For the third term, we use the hint (5.51) to obtain

$$\begin{aligned}
&\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p}\Big[(\boldsymbol{x}-\boldsymbol{\mu}_q)^\top \boldsymbol{\Sigma}_q^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_q)\Big] \\
&= \frac{1}{2}\Big((\boldsymbol{\mu}_p-\boldsymbol{\mu}_q)^\top \boldsymbol{\Sigma}_q^{-1}(\boldsymbol{\mu}_p-\boldsymbol{\mu}_q) + \mathrm{tr}\Big(\boldsymbol{\Sigma}_q^{-1}\boldsymbol{\Sigma}_p\Big)\Big).
\end{aligned}$$

Putting all terms together we get

$$\mathrm{KL}(p\|q) = \frac{1}{2}\left(\log\frac{\det(\boldsymbol{\Sigma}_q)}{\det(\boldsymbol{\Sigma}_p)} - d + (\boldsymbol{\mu}_p-\boldsymbol{\mu}_q)^\top \boldsymbol{\Sigma}_q^{-1}(\boldsymbol{\mu}_p-\boldsymbol{\mu}_q) + \mathrm{tr}\Big(\boldsymbol{\Sigma}_q^{-1}\boldsymbol{\Sigma}_p\Big)\right).$$

Solution to exercise 5.25 (Forward vs reverse KL).

1. Let p and q be discrete distributions. The derivation is analogous if p and q are taken to be continuous. First, we write the KL-divergence between p and q as

$$\begin{aligned}
&\mathrm{KL}(p\|q) \\
&= \sum_x\sum_y p(x,y)\log_2\frac{p(x,y)}{q(x)q(y)} && \text{using the definition of KL-divergence (5.44)} \\
&= \sum_x\sum_y p(x,y)\log_2 p(x,y) - \sum_x\sum_y p(x,y)\log_2 q(x) \\
&\qquad - \sum_x\sum_y p(x,y)\log_2 q(y) \\
&= \sum_x\sum_y p(x,y)\log_2 p(x,y) - \sum_x p(x)\log_2 q(x) - \sum_y p(y)\log_2 q(y) && \text{using the sum rule (1.10)} \\
&= -\mathrm{H}[p(x,y)] + \mathrm{H}[p(x)\|q(x)] + \mathrm{H}[p(y)\|q(y)] && \text{using the definitions of entropy (5.32) and cross-entropy (5.40)} \\
&= -\mathrm{H}[p(x,y)] + \mathrm{H}[p(x)] + \mathrm{H}[p(y)] + \mathrm{KL}(p(x)\|q(x)) && \text{using eq. (5.41)} \\
&\qquad + \mathrm{KL}(p(y)\|q(y))
\end{aligned}$$

$= \mathrm{KL}(p(x)\|q(x)) + \mathrm{KL}(p(y)\|q(y)) + \text{const.}$

Hence, to minimize $\mathrm{KL}(p\|q)$ with respect to the variational distributions $q(x)$ and $q(y)$ we should set $\mathrm{KL}(p(x)\|q(x)) = 0$ and $\mathrm{KL}(p(y)\|q(y)) = 0$, respectively. This is obtained when

$$q(x) = p(x) \qquad \text{and} \qquad q(y) = p(y).$$

2. The reverse KL-divergence $\mathrm{KL}(q\|p)$ on the finite domain $x, y \in \{1, 2, 3, 4\}$ is defined as

$$\mathrm{KL}(q\|p) = \sum_x \sum_y q(x)q(y) \log_2 \frac{q(x)q(y)}{p(x, y)}.$$

We can easily observe from the above formula that the support of q must be a subset of the support of p. In other words, if $q(x, y)$ is positive outside the support of p (i.e., when $p(x, y) = 0$) then $\mathrm{KL}(q\|p) = \infty$. Hence, the reverse KL-divergence has an infinite value except when the support of q is either $\{1, 2\} \times \{1, 2\}$ or $\{(3, 3)\}$ or $\{(4, 4)\}$. Thus, it has three local minima.
For the first case, the minimum is achieved when $q(x) = q(y) = (\frac{1}{2}, \frac{1}{2}, 0, 0)$. The corresponding KL-divergence is $\mathrm{KL}(q\|p) = \log_2 2 = 1$. For the second case and the third case, $q(x) = q(y) = (0, 0, 1, 0)$ and $q(x) = q(y) = (0, 0, 0, 1)$, respectively. The KL-divergence in both cases is $\mathrm{KL}(q\|p) = \log_2 4 = 2$.

3. Let us compute $p(x = 4)$ and $p(y = 1)$:

$$p(x = 4) = \sum_y p(x = 4, y) = \frac{1}{4},$$
$$p(y = 1) = \sum_x p(x, y = 1) = \frac{1}{4}.$$

Hence, $q(x = 4, y = 1) = p(x = 4)p(y = 1) = \frac{1}{16}$, however, $p(x = 4, y = 1) = 0$. We therefore have for the reverse KL-divergence that $\mathrm{KL}(q\|p) = \infty$.

Solution to exercise 5.29 (Gaussian VI vs Laplace approximation).

1. Recall from section 5.4.6 that $q^\star$ matches the first and second *moments* of p. In contrast, the Laplace approximation matches the *mode* of p and the second derivative of $-\log p$. In general, the mean is different from the mode, and so is the second moment from the second derivative.
2. We have

$$\begin{aligned} &\arg\min_{q \in \mathcal{Q}} \mathrm{KL}(q\|p(\cdot \mid \mathcal{D})) \\ &= \arg\max_{q \in \mathcal{Q}} L(q, p; \mathcal{D}) && \text{using eq. (5.62)} \end{aligned}$$

$$= \arg\max_{q \in \mathcal{Q}} \mathbb{E}_{\boldsymbol{\theta} \sim q}[\log p(y_{1:n}, \boldsymbol{\theta} \mid \boldsymbol{x}_{1:n})] + \mathrm{H}[q] \qquad \text{using eq. (5.63a)}$$

$$= \arg\max_{q \in \mathcal{Q}} \mathbb{E}_{\boldsymbol{\theta} \sim q}[\log p(y_{1:n}, \boldsymbol{\theta} \mid \boldsymbol{x}_{1:n})] + \frac{n}{2}\log(2\pi e) + \frac{1}{2}\log\det(\boldsymbol{\Sigma}). \qquad \text{using eq. (5.36)}$$

Differentiating with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we have that $\tilde{q}$ must satisfy

$$\begin{aligned} \mathbf{0} &= \boldsymbol{\nabla}_{\boldsymbol{\mu}} \mathbb{E}_{\boldsymbol{\theta} \sim \tilde{q}}[\log p(y_{1:n}, \boldsymbol{\theta} \mid \boldsymbol{x}_{1:n})] \\ \boldsymbol{\Sigma}^{-1} &= -2\boldsymbol{\nabla}_{\boldsymbol{\Sigma}} \mathbb{E}_{\boldsymbol{\theta} \sim \tilde{q}}[\log p(y_{1:n}, \boldsymbol{\theta} \mid \boldsymbol{x}_{1:n})]. \qquad \text{using the first hint} \end{aligned}$$

The result follows by eq. (5.72).

Solution to exercise 5.30 (Gradient of reverse-KL). To simplify the notation, we write $\boldsymbol{\Sigma} \doteq \mathrm{diag}\{\sigma_1^2, \dots, \sigma_d^2\}$. The reverse KL-divergence can be expressed as

$$\begin{aligned} \mathrm{KL}(q_\lambda \| p(\cdot)) &= \frac{1}{2}\left(\mathrm{tr}\left(\sigma_p^{-2}\boldsymbol{\Sigma}\right) + \sigma_p^{-2}\boldsymbol{\mu}^\top\boldsymbol{\mu} - d + \log\frac{(\sigma_p^2)^d}{\det(\boldsymbol{\Sigma})}\right) \qquad \text{using the expression for the KL-divergence of Gaussians (5.50)} \\ &= \frac{1}{2}\left(\sigma_p^{-2}\sum_{i=1}^d \sigma_i^2 + \sigma_p^{-2}\boldsymbol{\mu}^\top\boldsymbol{\mu} - d + d\log\sigma_p^2 - \sum_{i=1}^d \log\sigma_i^2\right). \end{aligned}$$

It follows immediately that $\boldsymbol{\nabla}_{\boldsymbol{\mu}} \mathrm{KL}(q_\lambda \| p(\cdot)) = \sigma_p^{-2}\boldsymbol{\mu}$. Moreover,

$$\frac{\partial}{\partial\sigma_i}\mathrm{KL}(q_\lambda \| p(\cdot)) = \frac{1}{2}\left(\sigma_p^{-2}\underbrace{\frac{\partial}{\partial\sigma_i}\sigma_i^2}_{2\sigma_i} - \underbrace{\frac{\partial}{\partial\sigma_i}\log\sigma_i^2}_{2/\sigma_i}\right) = \frac{\sigma_i}{\sigma_p^2} - \frac{1}{\sigma_i}.$$

Solution to exercise 5.33 (Reparameterizable distributions).

1. Let $Y \sim \mathrm{Unif}([0,1])$. Then, using the two hints,
$$(b-a)Y + a \sim \mathrm{Unif}([a,b]).$$
2. Let $Z_1 \sim \mathcal{N}(\mu, \sigma^2)$ and $Z_2 \sim \mathcal{N}(0,1)$. We have, $X = e^{Z_1}$. Recall from eq. (1.125) that Z_1 can equivalently be expressed in terms of Z_2 as $Z_1 = \sigma Z_2 + \mu$. This yields,
$$X = e^{Z_1} = e^{\sigma Z_2 + \mu}.$$
3. Denote by F the CDF of $\mathrm{Cauchy}(0,1)$. Observe that F is invertible with inverse $F^{-1}(y) = \tan(\pi(y - \frac{1}{2}))$. Let $Y \sim \mathrm{Unif}([0,1])$ and write $X = F^{-1}(Y)$. Then,
$$\begin{aligned} P_X(x) &= \mathbb{P}(X \leq x) \\ &= \mathbb{P}\left(F^{-1}(Y) \leq x\right) \\ &= \mathbb{P}(Y \leq F(x)) \\ &= F(x). \end{aligned}$$

This reparameterization works for any distribution with invertible CDF (not just Cauchy) and is known as the *universality of the uniform* (cf. section 1.1.5). The universality of the uniform is commonly used in pseudo-random number generators as it allows "lifting" samples from a uniform distribution to countless other well-known distributions.

4. The derivative of $\mathrm{ReLU}(z)$ is $\mathbb{1}\{z > 0\}$. Applying the reparameterization trick gives

$$\begin{aligned}
\frac{d}{d\mu}\mathbb{E}_{x\sim\mathcal{N}(\mu,1)}\mathrm{ReLU}(wx) &= \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)}\frac{d}{d\mu}\mathrm{ReLU}(w(\mu+\epsilon)) \\
&= w\mathbb{E}_\epsilon \mathbb{1}\{w(\mu+\epsilon) > 0\} && \text{using the chain rule} \\
&= w\mathbb{E}_\epsilon \mathbb{1}\{\mu+\epsilon > 0\} \\
&= w\mathbb{P}_\epsilon(\mu+\epsilon > 0) \\
&= w\mathbb{P}_\epsilon(\epsilon < \mu) \\
&= w\mathbb{P}_\epsilon(\epsilon \leq \mu) \\
&= w\Phi(\mu).
\end{aligned}$$

Markov Chain Monte Carlo Methods

Solution to exercise 6.3 (Markov chain update). We have

$$q_{t+1}(x') = \mathbb{P}\big(X_{t+1} = x'\big) = \sum_x \underbrace{\mathbb{P}(X_t = x)}_{q_t(x)} \underbrace{\mathbb{P}\big(X_{t+1} = x' \mid X_t = x\big)}_{p(x'|x)}. \qquad \text{using the sum rule (1.10)}$$

Noting that $p(x' \mid x) = \boldsymbol{P}(x, x')$, we conclude $\boldsymbol{q}_{t+1} = \boldsymbol{q}_t\boldsymbol{P}$.

Solution to exercise 6.4 (k-step transitions). It follows directly from the definition of matrix multiplication that

$$\begin{aligned}
\boldsymbol{P}^k(x, x') &= \sum_{x_1,\ldots,x_{k-1}} \boldsymbol{P}(x, x_1) \cdot \boldsymbol{P}(x_1, x_2) \cdots \boldsymbol{P}(x_{k-1}, x') \\
&= \sum_{x_1,\ldots,x_{k-1}} \mathbb{P}(X_1 = x_1 \mid X_0 = x) \cdots \mathbb{P}\big(X_k = x' \mid X_{k-1} = x_{k-1}\big) && \text{using the definition of the transition matrix (6.8)} \\
&= \sum_{x_1,\ldots,x_{k-1}} \mathbb{P}\big(X_1 = x_1, \cdots, X_{k-1} = x_{k-1}, X_k = x' \mid X_0 = x\big) && \text{using the product rule (1.14)} \\
&= \mathbb{P}\big(X_k = x' \mid X_0 = x\big). && \text{using the sum rule (1.10)}
\end{aligned}$$

Solution to exercise 6.14 (Finding stationary distributions). We consider the transition matrix

$$\boldsymbol{P} = \begin{bmatrix} 0.60 & 0.30 & 0.10 \\ 0.50 & 0.25 & 0.25 \\ 0.20 & 0.40 & 0.40 \end{bmatrix}.$$

We note that the entries of $\boldsymbol{P}$ are all different from 0, thus the Markov chain corresponding to this transition matrix is ergodic.[3] Thus, there exists a unique stationary distribution π to which the Markov chain converges irrespectively of the distribution over initial states q_0.

[3] All elements of the transition matrix being strictly greater than 0 is a sufficient, but not necessary, condition for ergodicity.

We know that $\boldsymbol{P}^\top \boldsymbol{\pi} = \boldsymbol{\pi}$ (where we write $\boldsymbol{\pi}$ as a column vector), therefore, to find the stationary distribution π, we need to find the normalized eigenvector associated with eigenvalue 1 of the matrix $\boldsymbol{P}^\top$. That is, we want to solve $(\boldsymbol{P}^\top - \boldsymbol{I})\boldsymbol{\pi} = \boldsymbol{0}$ for $\boldsymbol{\pi}$. We obtain the linear system of equations,

$$\begin{aligned} -0.40\pi_1 + 0.50\pi_2 + 0.20\pi_3 &= 0 \\ 0.30\pi_1 - 0.75\pi_2 + 0.40\pi_3 &= 0 \\ 0.10\pi_1 + 0.25\pi_2 - 0.60\pi_3 &= 0. \end{aligned}$$

Note that the left hand side of equation i corresponds to the probability of entering state i at stationarity minus π_i. Quite intuitively, this difference should be 0, that is, after one iteration the random walk is at state i with the same probability as before the iteration.

Solving this system of equations, for example, using the Gaussian elimination algorithm, we obtain the normalized eigenvector

$$\boldsymbol{\pi} = \frac{1}{72}\begin{bmatrix} 35 \\ 22 \\ 15 \end{bmatrix}.$$

Thus, we conclude that in the long run, the percentage of news days that will be classified as "good" is $35/72$.

Solution to exercise 6.23 (Practical examples of Gibbs sampling).

1. We have to compute the conditional distributions. Notice that for $x \in \{0, \dots, n\}$ and $y \in [0, 1]$,

   $$p(x, y) = \binom{n}{x} y^x (1-y)^{n-x} \cdot y^{\alpha-1}(1-y)^{\beta-1} = \mathrm{Bin}(x; n, y) \cdot C_y,$$

   where $\mathrm{Bin}(n, y)$ is the PMF of the binomial distribution (1.66) with n trials and success probability y, and C_y is a constant depending on y. It is clear that

   $$\begin{aligned} p(x \mid y) &= \frac{p(x, y)}{p(y)} && \text{using the definition of conditional probability (1.11)} \\ &= \frac{\mathrm{Bin}(x; n, y) \cdot C_y}{p(y)} \\ &= \mathrm{Bin}(x; n, y). && \text{using that } p(x \mid y) \text{ is a probability distribution over } x \text{ and } \mathrm{Bin}(x; n, y) \text{ already sums to 1, so } C_y = p(y) \end{aligned}$$

So in short, sampling from $p(x \mid y)$ is equivalent to sampling from a binomial distribution, which amounts to n times throwing a coin with bias y, and outputting the number of heads.
For the other conditional distribution, recall the PMF of the beta distribution with parameters α, β,

$$\text{Beta}(y; \alpha, \beta) = C \cdot y^{\alpha-1}(1-y)^{\beta-1}$$

where C is some constant depending on α and β only. We then have

$$p(x, y) = \text{Beta}(y; x+\alpha, n-x+\beta) \cdot C_x,$$

where C_x is some constant depending on x, α, β. This shows (analogously to above), that

$$p(y \mid x) = \text{Beta}(y; x+\alpha, n-x+\beta).$$

So for sampling y given x, one can sample from the beta distribution. There are several methods for sampling from a beta distribution, and we refer the reader to the corresponding Wikipedia page.

2. We first derive the posterior distribution of μ. We have

$$\begin{aligned}
\log p(\mu \mid \lambda, x_{1:n}) &= \log p(\mu) + \log p(x_{1:n} \mid \mu, \lambda) + \text{const} \\
&= -\frac{\lambda_0}{2}(\mu - \mu_0)^2 - \frac{\lambda}{2}\sum_{i=1}^{n}(x_i - \mu)^2 + \text{const} \\
&= -\frac{1}{2}(\underbrace{\lambda_0 + n\lambda}_{l_\lambda})\mu^2 + \Big(\underbrace{\lambda_0\mu_0 + \lambda\sum_{i=1}^{n} x_i}_{m_\lambda l_\lambda}\Big)\mu + \text{const.} && \text{by expanding the squares}
\end{aligned}$$

It follows that $l_\lambda = \lambda_0 + n\lambda, m_\lambda = (\lambda_0\mu_0 + \lambda\sum_{i=1}^{n} x_i)/l_\lambda$, and $\mu \mid \lambda, x_{1:n} \sim \mathcal{N}(\mu_\lambda, l_\lambda^{-1})$. That is, the posterior precision is the sum of prior precision and the precisions of each observation, and the posterior mean is a weighted average of prior mean and observations (where the weights are the precisions).
For the posterior distribution of λ, we have

$$\begin{aligned}
p(\lambda \mid \mu, x_{1:n}) &\propto p(\lambda) \cdot p(x_{1:n} \mid \mu, \lambda) \\
&= \lambda^{\alpha-1}e^{-\beta\lambda} \cdot \lambda^{\frac{n}{2}}e^{-\frac{\lambda}{2}\sum_{i=1}^{n}(x_i-\mu)^2} \\
&= \lambda^{\alpha+\frac{n}{2}-1}e^{-\lambda\left(\beta+\frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^2\right)} \\
&= \text{Gamma}(\lambda; a_\mu, b_\mu)
\end{aligned}$$

where $a_\mu = \alpha + \frac{n}{2}$ and $b_\mu = \beta + \frac{1}{2}\sum_{i=1}^{n}(x_i - \mu)^2$.

3. We have

$$p(\alpha, c \mid x_{1:n}) \propto p(\alpha, c) \cdot p(x_{1:n} \mid \alpha, c)$$

$$\propto \mathbb{1}\{\alpha, c > 0\} \prod_{i=1}^{n} \frac{\alpha c^\alpha}{x_i^{\alpha+1}} \mathbb{1}\{x_i \geq c\}$$
$$= \frac{\alpha^n c^{n\alpha}}{(\prod_{i=1}^{n} x_i)^{\alpha+1}} \mathbb{1}\{c < x^\star\} \mathbb{1}\{\alpha, c > 0\}$$

where $x^\star \doteq \min\{x_1, \ldots, x_n\}$. It is not clear how one could sample from this distribution directly. Instead, we use Gibbs sampling. For the posterior distribution of α, we have

$$\begin{aligned} p(\alpha \mid c, x_{1:n}) &\propto p(\alpha, c \mid x_{1:n}) \\ &\propto \frac{\alpha^n c^{n\alpha}}{(\prod_{i=1}^{n} x_i)^{\alpha+1}} \mathbb{1}\{\alpha > 0\} \\ &= \alpha^n \exp\left(-\alpha \left(\sum_{i=1}^{n} \log x_i - n \log c\right)\right) \mathbb{1}\{\alpha > 0\} \\ &\propto \text{Gamma}(\alpha; a, b) \end{aligned}$$

where $a \doteq n + 1$ and $b \doteq \sum_{i=1}^{n} \log x_i - n \log c$. On the other hand,

$$p(c \mid \alpha, x_{1:n}) \propto p(\alpha, c \mid x_{1:n}) \propto c^{n\alpha} \mathbb{1}\{0 < c < x^\star\}.$$

It remains to show that it is easy to sample from a random variable X with $p_X(x) \propto x^a \mathbb{1}\{0 < x < b\}$ for $a, b > 0$. We have that the normalizing constant is given by $\int_0^b x^a \, dx = \frac{b^{a+1}}{a+1}$. Therefore, the CDF of X is

$$\begin{aligned} P_X(x) &= \int_0^x p(y) \, dy \\ &= \frac{a+1}{b^{a+1}} \int_0^x y^a \, dy \\ &= \frac{a+1}{b^{a+1}} \frac{x^{a+1}}{a+1} \\ &= \left(\frac{x}{b}\right)^{a+1}. \end{aligned}$$

The inverse CDF is given by $P_X^{-1}(y) = b y^{\frac{1}{a+1}}$. Therefore, we can sample from X using inverse transform sampling (cf. section 1.1.5) by sampling $Y \sim \text{Unif}([0, 1])$ and setting $X = b Y^{\frac{1}{a+1}}$.

Solution to exercise 6.25 (Maximum entropy property of Gibbs distribution).

1. Our goal is to solve the optimization problem

$$\begin{aligned} &\max_{p \in \Delta^{\mathcal{T}}} -\sum_{x \in \mathcal{T}} p(x) \log_2 p(x) \\ &\text{subject to} \sum_{x \in \mathcal{T}} p(x) f(x) = \mu \end{aligned} \tag{B.8}$$

for some $\mu \in \mathbb{R}$. The Lagrangian with dual variables λ_0 and λ_1 is given by

$$\begin{aligned} L(p, \lambda_0, \lambda_1) = &- \sum_{x \in \mathcal{T}} p(x) \log_2 p(x) + \lambda_0 \left(1 - \sum_{x \in \mathcal{T}} p(x) \right) + \\ &\lambda_1 \left(\mu - \sum_{x \in \mathcal{T}} p(x) f(x) \right) \\ = &- \sum_{x \in \mathcal{T}} p(x) (\log_2 p(x) + \lambda_0 + \lambda_1 f(x)) + \text{const.} \end{aligned}$$

Note that $L(p, \lambda_0, \lambda_1) = \mathrm{H}[p]$ if the constraints are satisfied. Thus, we simply need to solve the (dual) optimization problem

$$\max_{\boldsymbol{p} \geq \mathbf{0}} \min_{\lambda_0, \lambda_1 \in \mathbb{R}} L(\boldsymbol{p}, \lambda_0, \lambda_1).$$

We have

$$\frac{\partial}{\partial p(x)} L(p, \lambda_0, \lambda_1) = -\log_2 p(x) - \lambda_0 - \lambda_1 f(x) - 1.$$

Setting the partial derivatives to zero, we obtain

$$\begin{aligned} p(x) &= 2 \exp(-\lambda_0 - \lambda_1 f(x) - 1) && \text{using } \log_2(\cdot) = \tfrac{\log(\cdot)}{\log(2)} \\ &\propto \exp(-\lambda_1 f(x)). \end{aligned}$$

Clearly, p is a valid probability mass function when normalized (i.e., for an appropriate choice of λ_0). We complete the proof by setting $T \doteq \frac{1}{\lambda_1}$.

2. As $T \to \infty$ (and $\lambda_1 \to 0$), the optimization problem reduces to picking the maximum entropy distribution without the first-moment constraint. This distribution is the uniform distribution over $\mathcal{T}$. Conversely, as $T \to 0$ (and $\lambda_1 \to \infty$), the Gibbs distribution reduces to a point density around its mode.

Solution to exercise 6.26 (Energy function of Bayesian logistic regression). First, note that the sum of convex functions is convex, hence, we consider each term individually.

The Hessian of the regularization term is $\lambda \boldsymbol{I}$, and thus, by the second-order characterization of convexity, this term is convex in $\boldsymbol{w}$.

Finally, note that the second term is a sum of logistic losses $\ell_{\log}$ (5.14), and we have seen in exercise 5.3 that $\ell_{\log}$ is convex in $\boldsymbol{w}$.

Solution to exercise 6.28 (Energy reduction of Gibbs sampling). Recall the following two facts:

1. Gibbs sampling is an instance of Metropolis-Hastings with proposal distribution

$$r(\boldsymbol{x}' \mid \boldsymbol{x}) = \begin{cases} p(x_i' \mid \boldsymbol{x}'_{-i}) & \text{if } \boldsymbol{x}' \text{ differs from } \boldsymbol{x} \text{ only in entry } i \\ 0 & \text{otherwise} \end{cases}$$

   and acceptance distribution $\alpha(\boldsymbol{x}' \mid \boldsymbol{x}) = 1$.
2. The acceptance distribution of Metropolis-Hastings where the stationary distribution p is a Gibbs distribution with energy function f is

$$\alpha(\boldsymbol{x}' \mid \boldsymbol{x}) = \min\left\{1, \frac{r(\boldsymbol{x} \mid \boldsymbol{x}')}{r(\boldsymbol{x}' \mid \boldsymbol{x})} \exp(f(\boldsymbol{x}) - f(\boldsymbol{x}'))\right\}.$$

We therefore know that

$$\frac{r(\boldsymbol{x} \mid \boldsymbol{x}')}{r(\boldsymbol{x}' \mid \boldsymbol{x})} \exp(f(\boldsymbol{x}) - f(\boldsymbol{x}')) \geq 1.$$

We remark that this inequality even holds with equality using our derivation of theorem 6.21. Taking the logarithm and reorganizing the terms, we obtain

$$f(\boldsymbol{x}') \leq f(\boldsymbol{x}) + \log r(\boldsymbol{x} \mid \boldsymbol{x}') - \log r(\boldsymbol{x}' \mid \boldsymbol{x}). \tag{B.9}$$

By the definition of the proposal distribution of Gibbs sampling,

$$r(\boldsymbol{x}' \mid \boldsymbol{x}) = p(x_i' \mid \boldsymbol{x}_{-i}) \quad \text{and} \quad r(\boldsymbol{x} \mid \boldsymbol{x}') = p(x_i \mid \boldsymbol{x}_{-i}). \qquad \text{using } \boldsymbol{x}'_{-i} = \boldsymbol{x}_{-i}$$

Taking the expectation of eq. (B.9),

$$\begin{aligned} \mathbb{E}_{x_i' \sim p(\cdot \mid \boldsymbol{x}_{-i})}\left[f(\boldsymbol{x}')\right] &\leq f(\boldsymbol{x}) + \log p(x_i \mid \boldsymbol{x}_{-i}) - \mathbb{E}_{x_i' \sim p(\cdot \mid \boldsymbol{x}_{-i})}\left[\log p(x_i' \mid \boldsymbol{x}_{-i})\right] \\ &= f(\boldsymbol{x}) - \mathrm{S}[p(x_i \mid \boldsymbol{x}_{-i})] + \mathrm{H}[p(\cdot \mid \boldsymbol{x}_{-i})]. \end{aligned}$$

That is, the energy is expected to decrease if the expected surprise of the new sample $x_i' \mid \boldsymbol{x}_{-i}$ is smaller than the surprise of the current sample $x_i \mid \boldsymbol{x}_{-i}$.

Solution to exercise 6.31 (Mixing time of Langevin dynamics).

1. We take the minimum of eq. (6.52) with respect to $\boldsymbol{y}$ on both sides. The minimum of the left-hand side is $f(\mathbf{0}) = 0$. To find the minimum of the right-hand side, we differentiate with respect to $\boldsymbol{y}$:

$$\frac{\partial}{\partial \boldsymbol{y}}\left[f(\boldsymbol{x}) + \boldsymbol{\nabla} f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}) + \frac{\alpha}{2} \|\boldsymbol{y} - \boldsymbol{x}\|_2^2\right] = \boldsymbol{\nabla} f(\boldsymbol{x}) + \alpha(\boldsymbol{y} - \boldsymbol{x}).$$

   Setting the partial derivative to zero, we obtain

$$\boldsymbol{y} = \boldsymbol{x} - \frac{1}{\alpha} \boldsymbol{\nabla} f(\boldsymbol{x}).$$

Plugging this $\boldsymbol{y}$ into eq. (6.52), we have

$$\begin{aligned} 0 &\geq f(\boldsymbol{x}) - \frac{1}{\alpha} \|\nabla f(\boldsymbol{x})\|_2^2 + \frac{1}{2\alpha} \|\nabla f(\boldsymbol{x})\|_2^2 \\ &= f(\boldsymbol{x}) - \frac{1}{2\alpha} \|\nabla f(\boldsymbol{x})\|_2^2 . \end{aligned}$$

2. Using the chain rule,

$$\frac{d}{dt} f(\boldsymbol{x}_t) = \nabla f(\boldsymbol{x}_t)^\top \frac{d}{dt} \boldsymbol{x}_t .$$

Note that $\frac{d}{dt}\boldsymbol{x}_t = -\nabla f(\boldsymbol{x}_t)$ by eq. (6.51), so

$$\begin{aligned} &= -\|\nabla f(\boldsymbol{x}_t)\|_2^2 \\ &\leq -2\alpha f(\boldsymbol{x}_t) \end{aligned}$$

where the last step follows from the PL-inequality (6.53).

3. Follows directly from (2) and Grönwall's inequality (6.54) by letting $g(t) = f(\boldsymbol{x}_t)$ and noting that $-\int_0^t 2\alpha \, ds = -2\alpha t$.
4. It suffices to show $\nabla q_t = q_t \nabla \log q_t$. By the chain rule,

$$\nabla \log q_t = \frac{\nabla q_t}{q_t}.$$

We obtain the desired result by rearranging the terms.

5. We have

$$\frac{d}{dt} \mathrm{KL}(q_t \| p) = \frac{d}{dt} \int q_t \log \frac{q_t}{p} \, d\boldsymbol{\theta}.$$

By the chain rule,

$$= \int \frac{\partial q_t}{\partial t} \log \frac{q_t}{p} \, d\boldsymbol{\theta} + \int q_t \frac{\partial}{\partial t} \log \frac{q_t}{p} \, d\boldsymbol{\theta}.$$

For the second term, we have

$$\int q_t \frac{\partial}{\partial t} \log \frac{q_t}{p} \, d\boldsymbol{\theta} = \int \frac{\partial q_t}{\partial t} \, d\boldsymbol{\theta} = \frac{d}{dt} \underbrace{\int q_t \, d\boldsymbol{\theta}}_{1} = 0.$$

Plugging in the Fokker-Planck equation (6.56) into the first term, we obtain

$$= \int \nabla \cdot \left(q_t \nabla \log \frac{q_t}{p} \right) \log \frac{q_t}{p} \, d\boldsymbol{\theta}.$$

Letting $\varphi \doteq \log \frac{q_t}{p}$ and $\mathbf{F} \doteq \nabla \varphi$, and then applying the hint, we have

$$\begin{aligned} &= \int (\nabla \cdot q_t \mathbf{F}) \varphi \, d\boldsymbol{\theta} \\ &= -\int q_t \|\nabla \varphi\|_2^2 \, d\boldsymbol{\theta} \end{aligned}$$

$$= -\mathbb{E}_{\boldsymbol{\theta}\sim q_t}\left[\left\|\boldsymbol{\nabla}\log\frac{q_t(\boldsymbol{\theta})}{p(\boldsymbol{\theta})}\right\|_2^2\right]$$
$$= -\mathrm{J}(q_t\|p).$$

6. Noting that p satisfies the LSI with constant α and combining with (5), we have that $\frac{d}{dt}\mathrm{KL}(q_t\|p) \leq -2\alpha\mathrm{KL}(q_t\|p)$. Observe that this result is analogous to the result derived in (2) and the LSI can intuitively be seen as the PL-inequality, but in the space of distributions. Analogously to (3), we obtain the desired convergence result by applying Grönwall's inequality (6.54).
7. By Pinsker's inequality (6.21), $\|q_t - p\|_{\mathrm{TV}} \leq e^{-\alpha t}\sqrt{2\mathrm{KL}(q_0\|p)}$. It follows from elementary algebra that $\|q_t - p\|_{\mathrm{TV}} \leq \epsilon$ if $t \geq \frac{1}{\alpha}\log\frac{C}{\epsilon}$ where $C \doteq \sqrt{2\mathrm{KL}(q_0\|p)}$. Thus, $\tau_{\mathrm{TV}}(\epsilon) \in \mathcal{O}(\log(1/\epsilon))$.

Solution to exercise 6.33 (Hamiltonian Monte Carlo).

1. We show that under the dynamics (6.62), the Hamiltonian $H(\boldsymbol{x},\boldsymbol{y})$ is a constant. In particular, $H(\boldsymbol{x}',\boldsymbol{y}') = H(\boldsymbol{x},\boldsymbol{y})$. This directly implies that

$$\alpha((\boldsymbol{x}',\boldsymbol{y}') \mid (\boldsymbol{x},\boldsymbol{y})) = \min\{1, \exp(H(\boldsymbol{x}',\boldsymbol{y}') - H(\boldsymbol{x},\boldsymbol{y}))\} = 1.$$

   To see why $H(\boldsymbol{x},\boldsymbol{y})$ is constant, we compute

$$\begin{aligned}\frac{d}{dt}H(\boldsymbol{x},\boldsymbol{y}) &= \boldsymbol{\nabla}_{\boldsymbol{x}} H \cdot \frac{d\boldsymbol{x}}{dt} + \boldsymbol{\nabla}_{\boldsymbol{y}} H \cdot \frac{d\boldsymbol{y}}{dt} && \text{using the chain rule}\\ &= \boldsymbol{\nabla}_{\boldsymbol{x}} H \cdot \boldsymbol{\nabla}_{\boldsymbol{y}} H - \boldsymbol{\nabla}_{\boldsymbol{x}} H \cdot \boldsymbol{\nabla}_{\boldsymbol{y}} H && \text{using the Hamiltonian dynamics (6.62)}\\ &= 0.\end{aligned}$$

2. By applying one Leapfrog step, we have for $\boldsymbol{x}$,

$$\begin{aligned}\boldsymbol{x}(t+\tau) &= \boldsymbol{x}(t) + \tau\boldsymbol{y}(t+\tau/2) && \text{using eq. (6.63b)}\\ &= \boldsymbol{x}(t) + \tau\Big(\boldsymbol{y}(t) - \frac{\tau}{2}\boldsymbol{\nabla}_{\boldsymbol{x}} f(\boldsymbol{x}(t))\Big) && \text{using eq. (6.63a)}\\ &= \boldsymbol{x}(t) - \frac{\tau^2}{2}\boldsymbol{\nabla}_{\boldsymbol{x}} f(\boldsymbol{x}(t)) + \tau\boldsymbol{y}(t).\end{aligned}$$

   Now observe that $\boldsymbol{y}(t)$ is a Gaussian random variable, independent of $\boldsymbol{x}$ (because we sample $\boldsymbol{y}$ freshly at the beginning of the L Leapfrog steps, and we are doing just one Leapfrog step). By renaming $\tau' \doteq \tau^2/2$ and $\epsilon \doteq \boldsymbol{y}(t)$, we get

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \tau'\boldsymbol{\nabla}_{\boldsymbol{x}} f(\boldsymbol{x}_t) + \sqrt{2\tau'}\epsilon$$

   which coincides with the proposal distribution of Langevin Monte Carlo (6.47).

Bayesian Deep Learning

Solution to exercise 7.2 (Softmax is a generalization of the logistic function). We have

$$\begin{aligned}\sigma_1(\boldsymbol{f}) &= \frac{\exp(f_1)}{\exp(f_0)+\exp(f_1)} && \text{using the definition of the softmax function (7.5)}\\ &= \frac{1}{\exp(f_0-f_1)+1} && \text{multiplying by } \tfrac{\exp(-f_1)}{\exp(-f_1)}\\ &= \sigma(-(f_0-f_1)). && \text{using the definition of the logistic function (5.10)}\end{aligned}$$

$\sigma_0(\boldsymbol{f}) = 1-\sigma(f)$ follows from the fact that $\sigma_0(\boldsymbol{f})+\sigma_1(\boldsymbol{f}) = 1$.

Active Learning

Solution to exercise 8.4 (Mutual information and KL divergence). We have

$$\begin{aligned}\mathrm{I}(\mathbf{X};\mathbf{Y}) &= \mathrm{H}[\mathbf{X}] - \mathrm{H}[\mathbf{X}\mid\mathbf{Y}] && \text{using the definition of mutual information (8.9)}\\ &= \mathbb{E}_{\boldsymbol{x}}[-\log p(\boldsymbol{x})] - \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}[-\log p(\boldsymbol{x}\mid\boldsymbol{y})] && \text{using the definitions of entropy (5.32) and conditional entropy (8.2)}\\ &= \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}[-\log p(\boldsymbol{x})] - \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}[-\log p(\boldsymbol{x}\mid\boldsymbol{y})] && \text{using the law of total expectation (1.15)}\\ &= \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}\left[\log\frac{p(\boldsymbol{x}\mid\boldsymbol{y})}{p(\boldsymbol{x})}\right].\end{aligned}$$

From this we get directly that

$$\begin{aligned}\mathrm{I}(\mathbf{X};\mathbf{Y}) &= \mathbb{E}_{\boldsymbol{y}}\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y}}\left[\log\frac{p(\boldsymbol{x}\mid\boldsymbol{y})}{p(\boldsymbol{x})}\right]\\ &= \mathbb{E}_{\boldsymbol{y}}[\mathrm{KL}(p(\boldsymbol{x}\mid\boldsymbol{y})\|p(\boldsymbol{x}))], && \text{using the definition of KL-divergence (5.44)}\end{aligned}$$

and we also conclude

$$\begin{aligned}\mathrm{I}(\mathbf{X};\mathbf{Y}) &= \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}\left[\log\frac{p(\boldsymbol{x},\boldsymbol{y})}{p(\boldsymbol{x})p(\boldsymbol{y})}\right] && \text{using the definition of conditional probability (1.11)}\\ &= \mathrm{KL}(p(\boldsymbol{x},\boldsymbol{y})\|p(\boldsymbol{x})p(\boldsymbol{y})). && \text{using the definition of KL-divergence (5.44)}\end{aligned}$$

Solution to exercise 8.7 (Non-monotonicity of cond. mutual information). Symmetry of conditional mutual information (8.19) and eq. (8.18) give the following relationship,

$$\mathrm{I}(\mathbf{X};\mathbf{Y},\mathbf{Z}) = \mathrm{I}(\mathbf{X};\mathbf{Y}) + \mathrm{I}(\mathbf{X};\mathbf{Z}\mid\mathbf{Y}) = \mathrm{I}(\mathbf{X};\mathbf{Z}) + \mathrm{I}(\mathbf{X};\mathbf{Y}\mid\mathbf{Z}). \tag{B.10}$$

1. $\mathbf{X}\perp\mathbf{Z}$ implies $\mathrm{I}(\mathbf{X};\mathbf{Z}) = 0$. Thus, eq. (B.10) simplifies to

$$\mathrm{I}(\mathbf{X};\mathbf{Y}) + \mathrm{I}(\mathbf{X};\mathbf{Z}\mid\mathbf{Y}) = \mathrm{I}(\mathbf{X};\mathbf{Y}\mid\mathbf{Z}).$$

Using that $\mathrm{I}(\mathbf{X};\mathbf{Z}\mid\mathbf{Y}) \geq 0$, we conclude $\mathrm{I}(\mathbf{X};\mathbf{Y}\mid\mathbf{Z}) \geq \mathrm{I}(\mathbf{X};\mathbf{Y})$.

2. $\mathbf{X} \perp \mathbf{Z} \mid \mathbf{Y}$ implies $\mathrm{I}(\mathbf{X}; \mathbf{Z} \mid \mathbf{Y}) = 0$. Equation (B.10) simplifies to

$$\mathrm{I}(\mathbf{X}; \mathbf{Y}) = \mathrm{I}(\mathbf{X}; \mathbf{Z}) + \mathrm{I}(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z}).$$

   Using that $\mathrm{I}(\mathbf{X}; \mathbf{Z}) \geq 0$, we conclude $\mathrm{I}(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z}) \leq \mathrm{I}(\mathbf{X}; \mathbf{Y})$.
3. Again, eq. (B.10) simplifies to

$$\mathrm{I}(\mathbf{X}; \mathbf{Y}) = \mathrm{I}(\mathbf{X}; \mathbf{Z}) + \mathrm{I}(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z}).$$

   Using that $\mathrm{I}(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z}) \geq 0$, we conclude $\mathrm{I}(\mathbf{X}; \mathbf{Z}) \leq \mathrm{I}(\mathbf{X}; \mathbf{Y})$.

Solution to exercise 8.8 (Interaction information).

1. Expanding the definition of interaction information, one obtains

$$\begin{aligned}\mathrm{I}(\mathbf{X}; \mathbf{Y}; \mathbf{Z}) = &(\mathrm{H}[\mathbf{X}] + \mathrm{H}[\mathbf{Y}] + \mathrm{H}[\mathbf{Z}]) \\ &- (\mathrm{H}[\mathbf{X}, \mathbf{Y}] + \mathrm{H}[\mathbf{X}, \mathbf{Z}] + \mathrm{H}[\mathbf{Y}, \mathbf{Z}]) \\ &+ \mathrm{H}[\mathbf{X}, \mathbf{Y}, \mathbf{Z}],\end{aligned}$$

   and hence, interaction information is symmetric.
2. Conditional on either one of X_1 or X_2, the distribution of Y remains unchanged, and hence $\mathrm{I}(Y; X_1 \mid X_2) = \mathrm{I}(Y; X_2 \mid X_1) = 0$. Conversely, conditional on both X_1 and X_2, Y is fully determined, and hence $\mathrm{I}(Y; X_1, X_2) = 1$ noting that Y encodes one bit worth of information. Thus, $\mathrm{I}(Y; X_1; X_2) = -1$ meaning that there is synergy between X_1 and X_2 with respect to Y.

Solution to exercise 8.10 (Marginal gain of maximizing mutual information). As suggested, we derive the result in two steps.

1. First, we have

$$\begin{aligned}\Delta_I(\boldsymbol{x} \mid A) &= I(A \cup \{\boldsymbol{x}\}) - I(A) && \text{using the definition of marginal gain (8.24)} \\ &= \mathrm{I}(\boldsymbol{f}_{A\cup\{\boldsymbol{x}\}}; \boldsymbol{y}_A, y_{\boldsymbol{x}}) - \mathrm{I}(\boldsymbol{f}_A; \boldsymbol{y}_A) && \text{using the definition of } I \text{ (8.23)} \\ &= \mathrm{I}(\boldsymbol{f}_{A\cup\{\boldsymbol{x}\}}; \boldsymbol{y}_A, y_{\boldsymbol{x}}) - \mathrm{I}(\boldsymbol{f}_{A\cup\{\boldsymbol{x}\}}; \boldsymbol{y}_A) && \text{using } \boldsymbol{y}_A \perp f_{\boldsymbol{x}} \mid \boldsymbol{f}_A \\ &= \mathrm{H}[\boldsymbol{f}_{A\cup\{\boldsymbol{x}\}} \mid \boldsymbol{y}_A] - \mathrm{H}[\boldsymbol{f}_{A\cup\{\boldsymbol{x}\}} \mid \boldsymbol{y}_A, y_{\boldsymbol{x}}] && \text{using the definition of MI (8.9)} \\ &= \mathrm{I}(\boldsymbol{f}_{A\cup\{\boldsymbol{x}\}}; y_{\boldsymbol{x}} \mid \boldsymbol{y}_A) && \text{using the definition of cond. MI (8.16)} \\ &= \mathrm{I}(\boldsymbol{f}_A; y_{\boldsymbol{x}} \mid f_{\boldsymbol{x}}, \boldsymbol{y}_A) + \mathrm{I}(f_{\boldsymbol{x}}; y_{\boldsymbol{x}} \mid \boldsymbol{y}_A) && \text{using eq. (8.18)} \\ &= \mathrm{I}(f_{\boldsymbol{x}}; y_{\boldsymbol{x}} \mid \boldsymbol{y}_A). && \text{using } \mathrm{I}(\boldsymbol{f}_A; y_{\boldsymbol{x}} \mid f_{\boldsymbol{x}}, \boldsymbol{y}_A) = 0 \text{ as } y_{\boldsymbol{x}} \perp \boldsymbol{f}_A \mid f_{\boldsymbol{x}}\end{aligned}$$

2. For the second part, we get

$$\begin{aligned}\mathrm{I}(f_{\boldsymbol{x}}; y_{\boldsymbol{x}} \mid \boldsymbol{y}_A) &= \mathrm{I}(y_{\boldsymbol{x}}; f_{\boldsymbol{x}} \mid \boldsymbol{y}_A) && \text{using symmetry of conditional MI (8.19)} \\ &= \mathrm{H}[y_{\boldsymbol{x}} \mid \boldsymbol{y}_A] - \mathrm{H}[y_{\boldsymbol{x}} \mid f_{\boldsymbol{x}}, \boldsymbol{y}_A] && \text{using the definition of cond. MI (8.16)} \\ &= \mathrm{H}[y_{\boldsymbol{x}} \mid \boldsymbol{y}_A] - \mathrm{H}[y_{\boldsymbol{x}} \mid f_{\boldsymbol{x}}] && \text{using that } y_{\boldsymbol{x}} \perp \boldsymbol{\epsilon}_A \text{ so } y_{\boldsymbol{x}} \perp \boldsymbol{y}_A \mid f_{\boldsymbol{x}} \\ &= \mathrm{H}[y_{\boldsymbol{x}} \mid \boldsymbol{y}_A] - \mathrm{H}[\epsilon_{\boldsymbol{x}}]. && \text{given } f_{\boldsymbol{x}}\text{, the only randomness in } y_{\boldsymbol{x}} \text{ originates from } \epsilon_{\boldsymbol{x}}\end{aligned}$$

Solution to exercise 8.14 (Submodularity means no synergy). We have

$$\begin{aligned}
\mathrm{I}(f_x; y_x; \boldsymbol{y}_{B\setminus A} \mid \boldsymbol{y}_A) &= \mathrm{I}(f_x; y_x \mid \boldsymbol{y}_A) - \mathrm{I}(f_x; y_x \mid \boldsymbol{y}_B) && \text{using the definitiion of interaction information (8.21)} \\
&= \Delta_I(\boldsymbol{x} \mid A) - \Delta_I(\boldsymbol{x} \mid B) && \text{using eq. (8.25)} \\
&\geq 0
\end{aligned}$$

where the final inequality follows from submodularity of I.

Bayesian Optimization

Solution to exercise 9.2 (Convergence to the static optimum). First, observe that

$$\begin{aligned}
\lim_{T\to\infty} \frac{R_T}{T} &= \max_{\boldsymbol{x}} f^\star(\boldsymbol{x}) - \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} f^\star(\boldsymbol{x}_t) && \text{using the definition of regret (9.4)} \\
&= \max_{\boldsymbol{x}} f^\star(\boldsymbol{x}) - \lim_{t\to\infty} f^\star(\boldsymbol{x}_t). && \text{(B.11)} \quad \text{using the Cesàro mean (9.7)}
\end{aligned}$$

Now, suppose that the algorithm converges to the static optimum,

$$\lim_{t\to\infty} f^\star(\boldsymbol{x}_t) = \max_{\boldsymbol{x}} f^\star(\boldsymbol{x}).$$

Together with eq. (B.11) we conclude that the algorithm achieves sublinear regret.

For the other direction, we prove the contrapositive. That is, we assume that the algorithm does not converge to the static optimum and show that it has (super-)linear regret. We distinguish between two cases.

1. First, suppose that $\lim_{t\to\infty} f^\star(\boldsymbol{x}_t)$ does not exist. Then, a suboptimal $\boldsymbol{x}_t$ is sampled infinitely often, and hence, R_T grows at least linearly with T.
2. On the other hand, if $\lim_{t\to\infty} f^\star(\boldsymbol{x}_t)$ exists, our assumption is formalized by

   $$\lim_{t\to\infty} f^\star(\boldsymbol{x}_t) < \max_{\boldsymbol{x}} f^\star(\boldsymbol{x}).$$

   Together with eq. (B.11) we conclude $\lim_{T\to\infty} R_T/T > 0$.

Solution to exercise 9.5 (Bayesian confidence intervals).

1. As suggested by the hint, we let $Z \sim \mathcal{N}(0,1)$ and $c > 0$, and bound

   $$\begin{aligned}
   \mathbb{P}(Z > c) &= \mathbb{P}(Z - c > 0) \\
   &= \int_0^\infty \frac{1}{\sqrt{2\pi}} e^{-(z+c)^2/2}\, dz. && \text{using the PDF of the univariate normal distribution (1.6)}
   \end{aligned}$$

Note that for $z \geq 0$,

$$\frac{(z+c)^2}{2} = \frac{z^2}{2} + zc + \frac{c^2}{2} \geq \frac{z^2}{2} + \frac{c^2}{2}.$$

Thus,

$$\begin{aligned}
&\leq e^{-c^2/2} \int_0^\infty \frac{1}{\sqrt{2\pi}} e^{-z^2/2}\, dz \\
&= e^{-c^2/2} \mathbb{P}(Z > 0) \\
&= \frac{1}{2} e^{-c^2/2}. && \text{(B.12)}
\end{aligned}$$

using symmetry of the standard normal distribution around 0

Since we made the Bayesian assumption $f^\star(\boldsymbol{x}) \sim \mathcal{N}(\mu_0(\boldsymbol{x}), \sigma_0^2(\boldsymbol{x}))$ and assumed Gaussian observation noise $y_t \sim \mathcal{N}(f^\star(\boldsymbol{x}_t), \sigma_n^2)$, the posterior is also Gaussian:

$$f^\star(\boldsymbol{x}) \mid \boldsymbol{x}_{1:t}, y_{1:t} \sim \mathcal{N}(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x})).$$

Hence, writing $\mathbb{P}_t(\cdot) \doteq \mathbb{P}(\cdot \mid \boldsymbol{x}_{1:t}, y_{1:t})$,

$$\begin{aligned}
\mathbb{P}_{t-1}(f^\star(\boldsymbol{x}) \notin \mathcal{C}_t(\boldsymbol{x})) &= \mathbb{P}_{t-1}(|f^\star(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})| > \beta_t \sigma_{t-1}(\boldsymbol{x})) \\
&= 2\mathbb{P}_{t-1}\left(\frac{f^\star(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})}{\sigma_{t-1}(\boldsymbol{x})} > \beta_t\right) && \text{using symmetry of the Gaussian distribution} \\
&\leq e^{-\beta_t^2/2}. && \text{using eq. (B.12) with } c = \beta_t
\end{aligned}$$

2. We have

$$\begin{aligned}
\mathbb{P}\left(\bigcup_{\boldsymbol{x} \in \mathcal{X}} \bigcup_{t \geq 1} f^\star(\boldsymbol{x}) \notin \mathcal{C}_t(\boldsymbol{x})\right) &\leq \sum_{\boldsymbol{x} \in \mathcal{X}} \sum_{t \geq 1} \mathbb{P}(f^\star(\boldsymbol{x}) \notin \mathcal{C}_t(\boldsymbol{x})) && \text{using a union bound (1.1)} \\
&\leq |\mathcal{X}| \sum_{t \geq 1} e^{-\beta_t^2/2}. && \text{using (1)}
\end{aligned}$$

Letting $\beta_t^2 \doteq 2\log(|\mathcal{X}|\,(\pi t)^2/(6\delta))$, we get

$$\begin{aligned}
&= \frac{6\delta}{\pi^2} \sum_{t \geq 1} \frac{1}{t^2} \\
&= \delta. && \text{using } \textstyle\sum_{t \geq 1} \frac{1}{t^2} = \frac{\pi^2}{6}
\end{aligned}$$

Solution to exercise 9.7 (Regret of GP-UCB).

1. We denote the static optimum by $\boldsymbol{x}^\star$. By the definition of $\boldsymbol{x}_t$,

$$\begin{aligned}
\mu_{t-1}(\boldsymbol{x}_t) + \beta_t \sigma_{t-1}(\boldsymbol{x}_t) &\geq \mu_{t-1}(\boldsymbol{x}^\star) + \beta_t \sigma_{t-1}(\boldsymbol{x}^\star) \\
&\geq f^\star(\boldsymbol{x}^\star). && \text{using eq. (9.9)}
\end{aligned}$$

Thus,

$$\begin{aligned}
r_t &= f^\star(\boldsymbol{x}^\star) - f^\star(\boldsymbol{x}_t) \\
&\leq \beta_t \sigma_{t-1}(\boldsymbol{x}_t) + \mu_{t-1}(\boldsymbol{x}_t) - f^\star(\boldsymbol{x}_t) \\
&\leq 2\beta_t \sigma_{t-1}(\boldsymbol{x}_t). && \text{again using eq. (9.9)}
\end{aligned}$$

2. We have for any fixed T,

$$\begin{aligned}
\mathrm{I}(\boldsymbol{f}_{T+1}; \boldsymbol{y}_{T+1}) &= \mathrm{H}[\boldsymbol{y}_{T+1}] - \mathrm{H}[\boldsymbol{\epsilon}_{T+1}] && \text{analogously to eq. (8.26)} \\
&= \mathrm{H}[\boldsymbol{y}_T] - \mathrm{H}[\boldsymbol{\epsilon}_T] + \mathrm{H}\big[y_{\boldsymbol{x}_{T+1}} \mid \boldsymbol{y}_T\big] - \mathrm{H}\big[\epsilon_{\boldsymbol{x}_{T+1}}\big] && \text{using the chain rule for entropy (8.5) and the mutual independence of } \boldsymbol{\epsilon}_{T+1} \\
&= \mathrm{I}(\boldsymbol{f}_T; \boldsymbol{y}_T) + \mathrm{I}\big(f_{\boldsymbol{x}_{T+1}}; y_{\boldsymbol{x}_{T+1}} \mid \boldsymbol{y}_T\big) && \text{using the definition of MI (8.9)} \\
&= \mathrm{I}(\boldsymbol{f}_T; \boldsymbol{y}_T) + \frac{1}{2} \log\left(1 + \frac{\sigma_T^2(\boldsymbol{x}_{T+1})}{\sigma_n^2}\right). && \text{using eq. (8.15)}
\end{aligned}$$

   Note that $\mathrm{I}(\boldsymbol{f}_0; \boldsymbol{y}_0) = 0$. The result then follows by induction.

3. By Cauchy-Schwarz, $R_T^2 \leq T \sum_{t=1}^T r_t^2$, and hence, it suffices to show $\sum_{t=1}^T r_t^2 \leq \mathcal{O}\big(\beta_T^2 \gamma_T\big)$. We have

$$\begin{aligned}
\sum_{t=1}^T r_t^2 &\leq 4\beta_T^2 \sum_{t=1}^T \sigma_{t-1}^2(\boldsymbol{x}_t) && \text{using part (1)} \\
&= 4\sigma_n^2 \beta_T^2 \sum_{t=1}^T \frac{\sigma_{t-1}^2(\boldsymbol{x}_t)}{\sigma_n^2}.
\end{aligned}$$

   Observe that $\sigma_{t-1}^2(\boldsymbol{x}_t)/\sigma_n^2$ is bounded by $M \doteq \max_{\boldsymbol{x} \in \mathcal{X}} \sigma_0^2(\boldsymbol{x})/\sigma_n^2$ as variance is monotonically decreasing (cf. section 1.6). Applying the hint, we obtain

$$\begin{aligned}
&\leq 4C\sigma_n^2 \beta_T^2 \sum_{t=1}^T \log\left(1 + \frac{\sigma_{t-1}^2(\boldsymbol{x}_t)}{\sigma_n^2}\right) \\
&= 8C\sigma_n^2 \beta_T^2 \mathrm{I}(\boldsymbol{f}_T; \boldsymbol{y}_T) && \text{using part (2)} \\
&\leq 8C\sigma_n^2 \beta_T^2 \gamma_T. && \text{using the definition of } \gamma_T \text{ (9.13)}
\end{aligned}$$

Solution to exercise 9.9 (Sublinear regret of GP-UCB for a linear kernel).

1. Let $S \subseteq \mathcal{X}$ be such that $|S| \leq T$. Recall from eq. (8.15) that $\mathrm{I}(\boldsymbol{f}_S; \boldsymbol{y}_S) = \frac{1}{2} \log\det\big(\boldsymbol{I} + \sigma_n^{-2} \boldsymbol{K}_{SS}\big)$. Using that the kernel is linear we can rewrite $\boldsymbol{K}_{SS} = \boldsymbol{X}_S^\top \boldsymbol{X}_S$. Using Weinstein-Aronszajn's identity (A.5) we have

$$\mathrm{I}(\boldsymbol{f}_S; \boldsymbol{y}_S) = \frac{1}{2} \log\det\Big(\boldsymbol{I} + \sigma_n^{-2} \boldsymbol{X}_S^\top \boldsymbol{X}_S\Big) = \frac{1}{2} \log\det\Big(\boldsymbol{I} + \sigma_n^{-2} \boldsymbol{X}_S \boldsymbol{X}_S^\top\Big).$$

   If we define $\boldsymbol{M} \doteq \boldsymbol{I} + \sigma_n^{-2} \boldsymbol{X}_S \boldsymbol{X}_S^\top$ as a sum of symmetric positive definite matrices, $\boldsymbol{M}$ itself is symmetric positive definite. Thus, we have from Hadamard's inequality (A.4),

$$\begin{aligned}
\det(\boldsymbol{M}) &\leq \det\Big(\operatorname{diag}\{\boldsymbol{I} + \sigma_n^{-2} \boldsymbol{X}_S \boldsymbol{X}_S^\top\}\Big) && \operatorname{diag}\{\boldsymbol{A}\} \text{ refers to the diagonal matrix whose elements are those of } \boldsymbol{A} \\
&= \det\Big(\boldsymbol{I} + \sigma_n^{-2} \operatorname{diag}\{\boldsymbol{X}_S \boldsymbol{X}_S^\top\}\Big).
\end{aligned}$$

   Note that

$$\operatorname{diag}\{\boldsymbol{X}_S \boldsymbol{X}_S^\top\}(i,i) = \sum_{t=1}^{|S|} \boldsymbol{x}_t(i)^2$$

$$\leq \sum_{i=1}^{d} \sum_{t=1}^{|S|} \boldsymbol{x}_t(i)^2 = \sum_{t=1}^{|S|} \underbrace{\|\boldsymbol{x}_t\|_2^2}_{\leq 1} \leq |S| \leq T.$$

If we denote by $\lambda \leq T$ the largest term of $\operatorname{diag}\{\boldsymbol{X}_S \boldsymbol{X}_S^\top\}$ then we have

$$\det(\boldsymbol{M}) \leq (1 + \sigma_n^{-2}\lambda)^d \leq (1 + \sigma_n^{-2}T)^d,$$

yielding,

$$\mathrm{I}(\boldsymbol{f}_S; \boldsymbol{y}_S) \leq \frac{d}{2} \log(1 + \sigma_n^{-2}T)$$

implying that $\gamma_T = \mathcal{O}(d \log T)$.

2. Using the regret bound (cf. theorem 9.6) and the Bayesian confidence intervals (cf. theorem 9.4), and then $\gamma_T = \mathcal{O}(d \log T)$, we have

$$R_T = \mathcal{O}\Big(\beta_T \sqrt{\gamma_T T}\Big) = \tilde{\mathcal{O}}\Big(\sqrt{dT}\Big),$$

and hence, $\lim_{T \to \infty} \frac{R_T}{T} = 0$.

Solution to exercise 9.12 (Closed-form expected improvement).

1. Note that f is a Gaussian process, and hence, our posterior distribution after round t is entirely defined by the mean function μ_t and the covariance function k_t. Reparameterizing the posterior distribution using a standard Gaussian (1.125), we obtain

$$f(\boldsymbol{x}) \mid \boldsymbol{x}_{1:t}, y_{1:t} = \mu_t(\boldsymbol{x}) + \sigma_t(\boldsymbol{x})\epsilon$$

for $\epsilon \sim \mathcal{N}(0, 1)$. We get

$$\begin{aligned}
\mathrm{EI}_t(\boldsymbol{x}) &= \mathbb{E}_{f(\boldsymbol{x}) \sim \mathcal{N}(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x}))}[I_t(\boldsymbol{x})] && \text{using the definition of expected improvement (9.23)} \\
&= \mathbb{E}_{f(\boldsymbol{x}) \sim \mathcal{N}(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x}))}\Big[(f(\boldsymbol{x}) - \hat{f}_t)_+\Big] && \text{using the definition of improvement (9.19)} \\
&= \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)}\Big[(\mu_t(\boldsymbol{x}) + \sigma_t(\boldsymbol{x})\epsilon - \hat{f}_t)_+\Big] && \text{using the reparameterization} \\
&= \int_{-\infty}^{+\infty} (\mu_t(\boldsymbol{x}) + \sigma_t(\boldsymbol{x})\epsilon - \hat{f}_t)_+ \cdot \phi(\epsilon)\, d\epsilon.
\end{aligned}$$

For $\epsilon < \frac{\hat{f}_t - \mu_t(\boldsymbol{x})}{\sigma_t(\boldsymbol{x})} \doteq z_t(\boldsymbol{x})$ we have $(\mu_t(\boldsymbol{x}) + \sigma_t(\boldsymbol{x})\epsilon - \hat{f}_t)_+ = 0$. Thus, we obtain

$$\mathrm{EI}_t(\boldsymbol{x}) = \int_{z_t(\boldsymbol{x})}^{+\infty} (\mu_t(\boldsymbol{x}) + \sigma_t(\boldsymbol{x})\epsilon - \hat{f}_t) \cdot \phi(\epsilon)\, d\epsilon. \tag{B.13}$$

2. By splitting the integral from eq. (B.13) into two distinct terms, we obtain

$$\begin{aligned}
\mathrm{EI}_t(\boldsymbol{x}) = (\mu_t(\boldsymbol{x}) - \hat{f}_t) \int_{z_t(\boldsymbol{x})}^{+\infty} \phi(\epsilon)\, d\epsilon \\
- \sigma_t(\boldsymbol{x}) \int_{z_t(\boldsymbol{x})}^{+\infty} (-\epsilon) \cdot \phi(\epsilon)\, d\epsilon.
\end{aligned}$$

For the first term, we use the symmetry of $\mathcal{N}(0,1)$ around 0 to write the integral in terms of the CDF. For the second term, we notice that $(-\epsilon)\cdot\phi(\epsilon) = \frac{1}{\sqrt{2\pi}}\frac{d}{d\epsilon}e^{-\epsilon^2/2}$. Thus, we can derive this integral directly,

$$\begin{aligned}&= (\mu_t(\boldsymbol{x}) - \hat{f}_t)\Phi(-z_t(\boldsymbol{x})) \\ &\quad - \sigma_t(\boldsymbol{x})\Big(\lim_{\epsilon\to\infty}\phi(\epsilon) - \phi(z_t(\boldsymbol{x}))\Big).\end{aligned}$$

Using the symmetry of ϕ around 0, we obtain

$$\mathrm{EI}_t(\boldsymbol{x}) = (\mu_t(\boldsymbol{x}) - \hat{f}_t)\Phi\left(\frac{\mu_t(\boldsymbol{x}) - \hat{f}_t}{\sigma_t(\boldsymbol{x})}\right) + \sigma_t(\boldsymbol{x})\phi\left(\frac{\mu_t(\boldsymbol{x}) - \hat{f}_t}{\sigma_t(\boldsymbol{x})}\right).$$

Solution to exercise 9.16 (Regret of IDS).

1. As $\boldsymbol{x}_t^{\mathrm{UCB}}$ is the UCB action,

$$\hat{\Delta}_t(\boldsymbol{x}_t^{\mathrm{UCB}}) = u_t(\boldsymbol{x}_t^{\mathrm{UCB}}) - l_t(\boldsymbol{x}_t^{\mathrm{UCB}}) = 2\beta_{t+1}\sigma_t(\boldsymbol{x}_t^{\mathrm{UCB}}).$$

2. We first bound $I_t(\boldsymbol{x}_t^{\mathrm{UCB}})$:

$$\begin{aligned}I_t(\boldsymbol{x}_t^{\mathrm{UCB}}) &= \mathrm{I}\Big(f_{\boldsymbol{x}_t^{\mathrm{UCB}}}; y_{\boldsymbol{x}_t^{\mathrm{UCB}}} \;\Big|\; \boldsymbol{x}_{1:t}, y_{1:t}\Big) \\ &= \frac{1}{2}\log\left(1 + \frac{\sigma_t^2(\boldsymbol{x}_t^{\mathrm{UCB}})}{\sigma_n^2}\right). && \text{using eq. (8.15)}\end{aligned}$$

Note that $\sigma_t^2(\boldsymbol{x})/\sigma_n^2 \leq C$ for some constant C since variance is decreasing monotonically. So, applying the hint,

$$\geq \frac{\sigma_t^2(\boldsymbol{x}_t^{\mathrm{UCB}})}{2C\sigma_n^2}.$$

Combining this with (1), we obtain

$$\widehat{\Psi}_t(\boldsymbol{x}_t^{\mathrm{UCB}}) = \frac{\hat{\Delta}_t(\boldsymbol{x}_t^{\mathrm{UCB}})^2}{I_t(\boldsymbol{x}_t^{\mathrm{UCB}})} \leq 8C\sigma_n^2\beta_{t+1}^2.$$

3. With high probability,

$$\Psi_t(\boldsymbol{x}_{t+1}) \leq \widehat{\Psi}_t(\boldsymbol{x}_{t+1}) \leq \widehat{\Psi}_t(\boldsymbol{x}_t^{\mathrm{UCB}}) \leq 8C\sigma_n^2\beta_{t+1}^2,$$

where the first inequality is due to $\Delta(\boldsymbol{x}) \leq \hat{\Delta}_t(\boldsymbol{x})$ with high probability, the second inequality due to the definition of the IDS algorithm (9.35), and the third inequality is from (2). Invoking theorem 9.13 with $\overline{\Psi}_T \doteq 8C\sigma_n^2\beta_T^2$, we obtain that

$$R_T \leq \sqrt{\gamma_T\overline{\Psi}_T T} = \beta_T\sqrt{8C\sigma_n^2\gamma_T T}$$

with high probability.

Markov Decision Processes

Solution to exercise 10.8 (Value functions). We can use eq. (10.15) to write the state-action values as a linear system of equations (i.e., as a "table"). This linear system can be solved, for example, using Gaussian elimination to yield the desired result.

Solution to exercise 10.13 (Greedy policies). It follows directly from the definition of the state-action value function (10.9) that

$$\arg\max_{a \in A} q(x, a) = \arg\max_{a \in A} r(x, a) + \gamma \sum_{x' \in X} p(x' \mid x, a) \cdot v(x').$$

Solution to exercise 10.16 (Optimal policies).

1. Recall from Bellman's theorem (10.32) that a policy is optimal iff it is greedy with respect to its state-action value function. Now, observe that in the "poor, unknown" state, the policy π is not greedy.
2. Analogously to exercise 10.8, we write the state-action values as a linear system of equations and solve the system using, e.g., Gaussian elimination.
3. Observe from the result of (2) that π' is greedy with respect to its state-action value function, and hence, it follows from Bellman's theorem that π' is optimal.

Solution to exercise 10.23 (Linear convergence of policy iteration). Using the hint and $\boldsymbol{v}^\star \geq \boldsymbol{v}^\pi$ for any policy π,

$$\begin{aligned}
\left\|\boldsymbol{v}^{\pi_t} - \boldsymbol{v}^\star\right\|_\infty &\leq \left\|\boldsymbol{B}^\star \boldsymbol{v}^{\pi_{t-1}} - \boldsymbol{v}^\star\right\|_\infty \\
&= \left\|\boldsymbol{B}^\star \boldsymbol{v}^{\pi_{t-1}} - \boldsymbol{B}^\star \boldsymbol{v}^\star\right\|_\infty && \text{using that } \boldsymbol{v}^\star \text{ is a fixed-point of } \boldsymbol{B}^\star, \text{ that is, } \boldsymbol{B}^\star \boldsymbol{v}^\star = \boldsymbol{v}^\star \\
&\leq \gamma \left\|\boldsymbol{v}^{\pi_{t-1}} - \boldsymbol{v}^\star\right\|_\infty && \text{using that } \boldsymbol{B}^\star \text{ is a contraction, see theorem 10.21} \\
&\leq \gamma^t \left\|\boldsymbol{v}^{\pi_0} - \boldsymbol{v}^\star\right\|_\infty . && \text{by induction}
\end{aligned}$$

Solution to exercise 10.24 (Reward modification).

1. Recall that the value function v_M^π for an MDP M is defined as $v_M^\pi(x) = \mathbb{E}_\pi\left[\sum_{t=0}^\infty \gamma^t R_t \mid X_0 = x\right]$. Given an optimal policy $\pi^\star$ for M and any policy π, we know that for any $x \in X$,

$$\begin{aligned}
& v_M^{\pi^\star}(x) \geq v_M^\pi(x) \\
\iff \quad & \mathbb{E}_{\pi^\star}\left[\textstyle\sum_{t=0}^\infty \gamma^t R_t \mid X_0 = x\right] \geq \mathbb{E}_\pi\left[\textstyle\sum_{t=0}^\infty \gamma^t R_t \mid X_0 = x\right] \\
\iff \quad & \mathbb{E}_{\pi^\star}\left[\textstyle\sum_{t=0}^\infty \gamma^t \alpha R_t \mid X_0 = x\right] \geq \mathbb{E}_\pi\left[\textstyle\sum_{t=0}^\infty \gamma^t \alpha R_t \mid X_0 = x\right] && \text{multiplying both sides by } \alpha \\
\iff \quad & \mathbb{E}_{\pi^\star}\left[\textstyle\sum_{t=0}^\infty \gamma^t R'_t \mid X_0 = x\right] \geq \mathbb{E}_\pi\left[\textstyle\sum_{t=0}^\infty \gamma^t R'_t \mid X_0 = x\right] \\
\iff \quad & v_{M'}^{\pi^\star}(x) \geq v_{M'}^\pi(x).
\end{aligned}$$

   Thus, $\pi^\star$ is an optimal policy for M'.

2. We give an example where the optimal policies differ when rewards are shifted.
   Consider an MDP with three states $\{1, 2, 3\}$ where 1 is the initial state and 3 is a terminal state. If one plays action A in states 1 or 2 one transitions directly to the terminal state. Additionally, in state 1 one can play action B which leads to state 2. Let every transition give a deterministic reward of $r \doteq -1$. Then it is optimal to traverse the shortest path to the terminal state, in particular, to choose action A when in state 1.
   If we consider the reward $r' \doteq r + 2 = 1$, then it is optimal to traverse the longest path to the terminal state, in particular, to choose action B when in state 1.
3. For an MDP M, we know that is optimal state-action value function satisfies Bellman's optimality equation (10.35),

$$q^\star_M(x,a) = \mathbb{E}_{x'|x,a}\left[r(x,x') + \gamma \max_{a' \in A} q^\star_M(x',a')\right].$$

   For the MDP M', we have

$$\begin{aligned} q^\star_{M'}(x,a) &= \mathbb{E}_{x'|x,a}\left[r'(x,x') + \gamma \max_{a' \in A} q^\star_{M'}(x',a')\right] \\ &= \mathbb{E}_{x'|x,a}\left[r(x,x') + f(x,x') + \gamma \max_{a' \in A} q^\star_{M'}(x',a')\right] \\ &= \mathbb{E}_{x'|x,a}\left[r(x,x') + \gamma\phi(x') - \phi(x) + \gamma \max_{a' \in A} q^\star_{M'}(x',a')\right]. \end{aligned}$$

   Reorganizing the terms, we obtain

$$q^\star_{M'}(x,a) + \phi(x) = \mathbb{E}_{x'|x,a}\left[r(x,x') + \gamma \max_{a' \in A}\big(q^\star_{M'}(x',a') + \phi(x')\big)\right].$$

   If we now define $q(x,a) \doteq q^\star_{M'}(x,a) + \phi(x)$, we have

$$q(x,a) = \mathbb{E}_{x'|x,a}\left[r(x,x') + \gamma \max_{a' \in A} q(x',a')\right].$$

   This is exactly Bellman's optimality equation for the MDP M with reward function r, and hence, $q \equiv q^\star_M$.
   If we take $\pi^\star$ to be an optimal policy for M, then it satisfies

$$\begin{aligned} \pi^\star(x) &\in \arg\max_{a \in A} q^\star_M(x,a) \\ &= \arg\max_{a \in A} q^\star_{M'}(x,a) + \phi(x) && \text{using the above characterization of } q^\star_M \\ &= \arg\max_{a \in A} q^\star_{M'}(x,a). && \text{using that } \phi(x) \text{ is independent of } a \end{aligned}$$

Tabular Reinforcement Learning

Solution to exercise 11.14 (Q-learning).

1. $Q^\star(A,\downarrow) = 1.355, Q^\star(G_1, \text{exit}) = 5.345, Q^\star(G_2, \text{exit}) = 0.5$

2. Repeating the given episodes infinitely often will not lead to convergence to the optimal Q-function because not all state-action pairs are visited infinitely often.
   Let us assume we observe the following episode instead of the first episode.

   Episode 3

x	a	x'	r
A	$\rightarrow$	B	0
B	$\downarrow$	G_2	0
G_2	exit		1

   If we repeat episodes 2 and 3 infinitely often, Q-learning will converge to the optimal Q-function as all state-action pairs will be visited infinitely often.
3. First, recall that Q-learning is an off-policy algorithm, and hence, even if episodes are obtained off-policy, Q-learning will still converge to the optimal Q-function (if the other convergence conditions are met). Note that it only matters which state-action pairs are observed and not which policies were followed to obtain these observations.
   The "closer" the initial Q-values are to the optimal Q-function, the faster the convergence of Q-learning. However, if the convergence conditions are met, Q-learning will converge to the optimal Q-function regardless of the initial Q-values.
4. $v^\star(A) = 10$, $v^\star(B) = 10$, $v^\star(G_1) = 10$, $v^\star(G_2) = 1$

Model-free Reinforcement Learning

Solution to exercise 12.2 (Q-learning and function approximation).

1. We have to show that

$$v^\star(x) = \max_{a \in A} r(x, a) + \gamma \mathbb{E}_{x'|x,a}\left[v^\star(x')\right]$$

   for every $x \in \{1, 2, \ldots, 7\}$. We give a derivation here for $x = 1$ and $x = 2$.
   - For $x = 1$,

$$\begin{aligned} v^\star(1) &= -3 \\ \max_{a \in A} r(1, a) + \gamma \mathbb{E}_{x'|1,a}\left[v^\star(x')\right] &= -3 \end{aligned}$$

     since

$$r(1, a) + \gamma \mathbb{E}_{x'|1,a}\left[v^\star(x')\right] = \begin{cases} -1 + -2 = -3 & \text{if } a = 1 \\ -1 + -3 = -4 & \text{if } a = -1. \end{cases}$$

- Likewise, for $x = 2$,

$$\begin{aligned} v^\star(2) &= -2 \\ \max_{a \in A} r(2,a) + \gamma \mathbb{E}_{x'|2,a}\big[v^\star(x')\big] &= -2 \end{aligned}$$

since

$$r(2,a) + \gamma \mathbb{E}_{x'|2,a}\big[v^\star(x')\big] = \begin{cases} -1 + -1 = -2 & \text{if } a = 1 \\ -1 + -3 = -4 & \text{if } a = -1. \end{cases}$$

2. We have

$$\begin{aligned} Q(3,-1) &= 0 + \frac{1}{2}\left(-1 + \max_{a' \in A} Q(2,a')\right) = \frac{1}{2}(-1+0) = -\frac{1}{2} \\ Q(2,1) &= 0 + \frac{1}{2}\left(-1 + \max_{a' \in A} Q(3,a')\right) = \frac{1}{2}(-1+0) = -\frac{1}{2} \\ Q(3,1) &= 0 + \frac{1}{2}\left(-1 + \max_{a' \in A} Q(4,a')\right) = \frac{1}{2}(-1+0) = -\frac{1}{2} \\ Q(4,1) &= 0 + \frac{1}{2}\left(0 + \max_{a' \in A} Q(4,a')\right) = \frac{1}{2}(0+0) = 0. \end{aligned}$$

3. We compute

$$\begin{aligned} \nabla_{\boldsymbol{w}} \ell(\boldsymbol{w};\tau) &= -\left(r + \gamma \max_{a' \in A} Q(x',a';\boldsymbol{w}^{\text{old}}) - Q(x,a;\boldsymbol{w})\right) \begin{bmatrix} x \\ a \\ 1 \end{bmatrix} && \text{using the derivation of eq. (12.15)} \\ &= -\left(-1 + \max_{a' \in A}\{1 - a' - 2\} - (-2 - 1 + 1)\right) \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} -4 \\ 2 \\ -2 \end{bmatrix}. \end{aligned}$$

This gives

$$\begin{aligned} \boldsymbol{w}' &= \boldsymbol{w} - \alpha \nabla_{\boldsymbol{w}} \ell(\boldsymbol{w};\tau) \\ &= \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} - \frac{1}{4} \begin{bmatrix} -4 \\ 2 \\ -2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1/2 \\ 3/2 \end{bmatrix}. \end{aligned}$$

Solution to exercise 12.6 (Eligibility vector). We have

$$\begin{aligned} \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(a \mid \boldsymbol{x}) &= \frac{\nabla_{\boldsymbol{\varphi}} \pi_{\boldsymbol{\varphi}}(a \mid \boldsymbol{x})}{\pi_{\boldsymbol{\varphi}}(a \mid \boldsymbol{x})} && \text{using the chain rule} \\ &= \boldsymbol{\phi}(\boldsymbol{x},a) - \frac{\sum_{b \in A} \boldsymbol{\phi}(\boldsymbol{x},b) \exp(\boldsymbol{\varphi}^\top \boldsymbol{\phi}(\boldsymbol{x},b))}{\sum_{b \in A} \exp(\boldsymbol{\varphi}^\top \boldsymbol{\phi}(\boldsymbol{x},b))} && \text{using elementary calculus} \\ &= \boldsymbol{\phi}(\boldsymbol{x},a) - \sum_{b \in A} \pi_{\boldsymbol{\varphi}}(b \mid \boldsymbol{x}) \cdot \boldsymbol{\phi}(\boldsymbol{x},b). \end{aligned}$$

Solution to exercise 12.8 (Variance of score gradients with baselines).

1. The result follows directly using that

$$\operatorname{Var}[f(\mathbf{X}) - g(\mathbf{X})] = \operatorname{Var}[f(\mathbf{X})] + \operatorname{Var}[g(\mathbf{X})] - 2\operatorname{Cov}[f(\mathbf{X}), g(\mathbf{X})]. \qquad \text{using eq. (1.48)}$$

2. Denote by $r(\tau)$ the discounted rewards attained by trajectory τ. Let $f(\tau) \doteq r(\tau)\nabla_{\boldsymbol{\varphi}} \Pi_{\boldsymbol{\varphi}}(\tau)$ and $g(\tau) \doteq b\nabla_{\boldsymbol{\varphi}} \Pi_{\boldsymbol{\varphi}}(\tau)$. Recall that $\mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}[g(\tau)] = 0$, implying that

$$\begin{aligned}\operatorname{Var}[g(\tau)] &= \operatorname{Var}\left[b\nabla_{\boldsymbol{\varphi}} \Pi_{\boldsymbol{\varphi}}(\tau)\right] \\ &= \mathbb{E}\left[\left(b\nabla_{\boldsymbol{\varphi}} \Pi_{\boldsymbol{\varphi}}(\tau)\right)^2\right]. && \text{using the definition of variance (1.44)}\end{aligned}$$

On the other hand,

$$\begin{aligned}\operatorname{Cov}[f(\tau), g(\tau)] &= \mathbb{E}[(f(\tau) - \mathbb{E}[f(\tau)])g(\tau)] && \text{using the definition of covariance (1.35)} \\ &= \mathbb{E}[f(\tau)g(\tau)] - \mathbb{E}[f(\tau)]\underbrace{\mathbb{E}[g(\tau)]}_{0} && \text{using linearity of expectation (1.24)} \\ &= \mathbb{E}\left[b \cdot r(\tau) \cdot \left(\nabla_{\boldsymbol{\varphi}} \Pi_{\boldsymbol{\varphi}}(\tau)\right)^2\right].\end{aligned}$$

Therefore, if $b^2 \leq 2b \cdot r(\boldsymbol{x}, \boldsymbol{a})$ for every state $\boldsymbol{x} \in \mathcal{X}$ and action $\boldsymbol{a} \in \mathcal{A}$, then the result follows from eq. (12.40).

Solution to exercise 12.9 (Score gradients with state-dependent baselines). First, observe that

$$\mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\left[G_0 \nabla_{\boldsymbol{\varphi}} \log \Pi_{\boldsymbol{\varphi}}(\tau)\right] = \mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\left[\sum_{t=0}^{T-1} G_0 \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t)\right], \qquad \text{using eq. (12.30)}$$

and hence, it suffices to show

$$\begin{aligned}&\mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\left[\sum_{t=0}^{T-1} G_0 \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t)\right] \\ &= \mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\left[\sum_{t=0}^{T-1} (G_0 - b(\tau_{0:t-1}))\nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t)\right].\end{aligned} \tag{B.14}$$

We prove eq. (B.14) with an induction on T. The base case ($T = 0$) is satisfied trivially. Fixing any T and assuming eq. (B.14) holds for T, we have,

$$\begin{aligned}&\mathbb{E}_{\tau\sim\Pi_{\boldsymbol{\varphi}}}\left[\sum_{t=0}^{T} (G_0 - b(\tau_{0:t-1}))\nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t)\right] \\ &= \mathbb{E}_{\tau_{0:T-1}}\left[\mathbb{E}_{\tau_T}\left[\sum_{t=0}^{T} (G_0 - b(\tau_{0:t-1}))\nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t) \;\middle|\; \tau_{0:T-1}\right]\right] && \text{using the tower rule (1.33)}\end{aligned}$$

$$= \mathbb{E}_{\tau_{0:T-1}}\Bigg[\sum_{t=0}^{T-1} G_0 \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t) \qquad \text{using the induction hypothesis}$$

$$+ \mathbb{E}_{\tau_T}\big[(G_0 - b(\tau_{0:T-1}))\nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_T \mid \boldsymbol{x}_T) \;\big|\; \tau_{0:T-1}\big]\Bigg].$$

Using the score function trick for the score function $\nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t)$ analogously to the proof of lemma 12.7, we have,

$$\begin{aligned}
&\mathbb{E}_{\tau_T}\big[b(\tau_{0:T-1})\nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_T \mid \boldsymbol{x}_T) \;\big|\; \tau_{0:T-1}\big] \\
&= \mathbb{E}_{\boldsymbol{a}_T}\big[b(\tau_{0:T-1})\nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_T \mid \boldsymbol{x}_T) \;\big|\; \tau_{0:T-1}\big] \\
&= b(\tau_{0:T-1}) \int \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_T \mid \boldsymbol{x}_T)\nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_T \mid \boldsymbol{x}_T)\, d\boldsymbol{a}_T \\
&= b(\tau_{0:T-1}) \int \nabla_{\boldsymbol{\varphi}} \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_T \mid \boldsymbol{x}_T)\, d\boldsymbol{a}_T \qquad \text{using the score function trick,} \\
&= 0. \qquad \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t) = \nabla_{\boldsymbol{\varphi}} \pi_{\boldsymbol{\varphi}}(a_t|x_t) / \pi_{\boldsymbol{\varphi}}(a_t|x_t)
\end{aligned}$$

Thus,

$$\begin{aligned}
&= \mathbb{E}_{\tau_{0:T-1}}\left[\mathbb{E}_{\tau_T}\left[\sum_{t=0}^{T} G_0 \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t) \;\middle|\; \tau_{0:T-1}\right]\right] \\
&= \mathbb{E}_{\tau \sim \Pi_{\boldsymbol{\varphi}}}\left[\sum_{t=0}^{T} G_0 \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t)\right]. \qquad \text{using the tower rule (1.33) again}
\end{aligned}$$

Solution to exercise 12.12 (Policy gradients with downstream returns). Each trajectory τ is described by four transitions,

$$\tau = (x_0, a_0, r_0, x_1, a_1, r_1, x_2, a_2, r_2, x_3, a_3, r_3, x_4).$$

Moreover, we have given

$$\begin{aligned}
\pi_\theta(2 \mid x) &= \theta, & \frac{\partial \pi_\theta(2 \mid x)}{\partial \theta} &= +1 \\
\pi_\theta(1 \mid x) &= 1 - \theta, & \frac{\partial \pi_\theta(1 \mid x)}{\partial \theta} &= -1.
\end{aligned}$$

We first compute the downstream returns for the given episode,

$$\begin{aligned}
G_{0:4} &= r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r^3 = 1 + \frac{1}{2}\cdot 0 + \frac{1}{4}\cdot 1 + \frac{1}{8}\cdot 1 = \frac{11}{8} \\
G_{1:4} &= r_1 + \gamma r_2 + \gamma^2 r_3 = 0 + \frac{1}{2}\cdot 1 + \frac{1}{4}\cdot 1 = \frac{3}{4} \\
G_{2:4} &= r_2 + \gamma r_3 = 1 + \frac{1}{2}\cdot 1 = \frac{3}{2} \\
G_{3:4} &= r_3 = 1.
\end{aligned}$$

Lastly, we can combine them to compute the policy gradient,

$$\nabla_\theta j(\theta) \approx \sum_{t=0}^{3} \gamma^t G_{t:4} \nabla_\theta \log \pi_\theta(a_t \mid x_t) \qquad \text{using Monte Carlo approximation of eq. (12.45) with a single sample}$$

$$= 1 \cdot \frac{11}{8} - 1 \cdot \frac{1}{2} \cdot \frac{3}{4} + 1 \cdot \frac{1}{4} \cdot \frac{3}{2} - 1 \cdot \frac{1}{8} \cdot 1 = \frac{5}{4}.$$

Solution to exercise 12.15 (Policy gradient with an exponential family).

1. The distribution on actions is given by

$$\pi_{\boldsymbol{\varphi}}(a \mid x) = \sigma(f_{\boldsymbol{\varphi}}(x))^a \cdot (1 - \sigma(f_{\boldsymbol{\varphi}}(x)))^{1-a}.$$

To simplify the notation, we write $\mathbb{E}$ for $\mathbb{E}_{(\boldsymbol{x},a)\sim\pi_{\boldsymbol{\varphi}}}$ and q for $q^{\pi_{\boldsymbol{\varphi}}}$. We get

$$\begin{aligned}
&\nabla_{\boldsymbol{\varphi}} j(\boldsymbol{\varphi}) \\
&= \mathbb{E}\left[q(\boldsymbol{x},a) \nabla_{\boldsymbol{\varphi}} \left(a \log \sigma(f_{\boldsymbol{\varphi}}(\boldsymbol{x})) + (1-a) \log(1 - \sigma(f_{\boldsymbol{\varphi}}(\boldsymbol{x})))\right)\right] && \text{using the policy gradient theorem (12.58)} \\
&= \mathbb{E}\left[q(\boldsymbol{x},a) \nabla_f \left(a \log \sigma(f) + (1-a) \log(1 - \sigma(f))\right) \nabla_{\boldsymbol{\varphi}} f_{\boldsymbol{\varphi}}(\boldsymbol{x})\right] && \text{using the chain rule} \\
&= \mathbb{E}\left[q(\boldsymbol{x},a) \nabla_f \left(-a \log(1 + e^{-f}) + (1-a) \log\left(\frac{e^{-f}}{1+e^{-f}}\right)\right) \nabla_{\boldsymbol{\varphi}} f_{\boldsymbol{\varphi}}(\boldsymbol{x})\right] && \text{using the definition of the logistic function (5.10)} \\
&= \mathbb{E}\left[q(\boldsymbol{x},a) \nabla_f \left(-f + af - \log(1 + e^{-f})\right) \nabla_{\boldsymbol{\varphi}} f_{\boldsymbol{\varphi}}(\boldsymbol{x})\right] \\
&= \mathbb{E}\left[q(\boldsymbol{x},a) \left(a - 1 + \frac{e^{-f}}{1+e^{-f}}\right) \nabla_{\boldsymbol{\varphi}} f_{\boldsymbol{\varphi}}(\boldsymbol{x})\right] \\
&= \mathbb{E}\left[q(\boldsymbol{x},a)(a - \sigma(f)) \nabla_{\boldsymbol{\varphi}} f_{\boldsymbol{\varphi}}(\boldsymbol{x})\right]. && \text{using the definition of the logistic function (5.10)}
\end{aligned}$$

The term $a - \sigma(f)$ can be understood as a residual as it corresponds to the difference between the target action a and the expected action $\sigma(f)$.

2. We have

$$\begin{aligned}
\nabla_{\boldsymbol{\varphi}} j(\boldsymbol{\varphi}) &= \mathbb{E}\left[q(\boldsymbol{x},a) \nabla_f \log \pi_f(a \mid \boldsymbol{x}) \nabla_{\boldsymbol{\varphi}} f(\boldsymbol{x})\right] && \text{using eq. (12.58) and the chain rule} \\
&= \mathbb{E}\left[q(\boldsymbol{x},a) \nabla_f \left(\log h(a) + af - A(f)\right) \nabla_{\boldsymbol{\varphi}} f(\boldsymbol{x})\right] \\
&= \mathbb{E}\left[q(\boldsymbol{x},a)(a - \nabla_f A(f)) \nabla_{\boldsymbol{\varphi}} f(\boldsymbol{x})\right].
\end{aligned}$$

3. We have $\nabla_f A(f) = \sigma(f)$. We are therefore looking for a function $A(f)$ whose derivative is $\sigma(f) = \frac{1}{1+e^{-f}} = \frac{e^f}{1+e^f}$. With this equality of the sigmoid we can find the integral, and we have $A(f) = \log(1 + e^f) + c$. Let us confirm that this gives us the Bernoulli distribution with $c = 0$:

$$\begin{aligned}
\pi_f(a \mid \boldsymbol{x}) &= h(a) \exp(af - \log(1 + e^f)) \\
&= h(a) \frac{e^{af}}{1 + e^f} \\
&= h(a) \begin{cases} \sigma(f) & \text{if } a = 1 \\ 1 - \sigma(f) & \text{if } a = 0. \end{cases}
\end{aligned}$$

This is the Bernoulli distribution with parameter $\sigma(f)$ where we have $h(a) = 1$.

4. Using that $\nabla_f A(f) = f$, we immediately get

$$\nabla_{\boldsymbol{\varphi}} j(\boldsymbol{\varphi}) = \mathbb{E}\big[q(\boldsymbol{x}, a)(a - f)\nabla_{\boldsymbol{\varphi}} f(\boldsymbol{x})\big].$$

5. No, we cannot use the reparameterization trick since we do not know how the states $\boldsymbol{x}$ depend on action a. These dependencies are determined by the unknown dynamics of the environment. Nonetheless, we can apply it after sampling an episode according to a policy and then updating policy parameters in hindsight. This is for example done by the soft actor-critic (SAC) algorithm (cf. section 12.5).

Solution to exercise 12.19 (Soft value function).

1. Equation (12.90) can be written as

$$\begin{aligned}
&\mathrm{KL}\big(\Pi_{\boldsymbol{\varphi}} \| \Pi_\star\big) \\
&= \sum_{t=1}^{T} \mathbb{E}_{(\boldsymbol{x}_t, \boldsymbol{a}_t) \sim \Pi_{\boldsymbol{\varphi}}} \big[\mathrm{S}[\beta(\boldsymbol{a}_t \mid \boldsymbol{x}_t)] - \mathrm{H}\big[\pi_{\boldsymbol{\varphi}}(\cdot \mid \boldsymbol{x}_t)\big]\big].
\end{aligned}$$

Adding and subtracting $\log Z(\boldsymbol{x}_t)$ gives

$$\begin{aligned}
&= \sum_{t=1}^{T} \mathbb{E}_{(\boldsymbol{x}_t, \boldsymbol{a}_t) \sim \Pi_{\boldsymbol{\varphi}}} \big[\mathrm{S}[\hat{\pi}(\boldsymbol{a}_t \mid \boldsymbol{x}_t)] - \log Z(\boldsymbol{x}_t) - \mathrm{H}\big[\pi_{\boldsymbol{\varphi}}(\cdot \mid \boldsymbol{x}_t)\big]\big] \\
&= \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{x}_t \sim \Pi_{\boldsymbol{\varphi}}} \big[\mathrm{H}\big[\pi_{\boldsymbol{\varphi}}(\cdot \mid \boldsymbol{x}_t) \| \hat{\pi}(\cdot \mid \boldsymbol{x}_t)\big] - \log Z(\boldsymbol{x}_t) - \mathrm{H}\big[\pi_{\boldsymbol{\varphi}}(\cdot \mid \boldsymbol{x}_t)\big]\big] && \text{using the definition of cross-entropy (5.40)} \\
&= \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{x}_t \sim \Pi_{\boldsymbol{\varphi}}} \big[\mathrm{KL}\big(\pi_{\boldsymbol{\varphi}}(\cdot \mid \boldsymbol{x}_t) \| \hat{\pi}(\cdot \mid \boldsymbol{x}_t)\big) - \log Z(\boldsymbol{x}_t)\big]. && \text{using the definition of KL-divergence (5.44)}
\end{aligned}$$

2. We prove the statement by (reverse) induction on t. For the base case, note that the term

$$\mathbb{E}_{\boldsymbol{x}_T \sim \Pi_{\boldsymbol{\varphi}}} \big[\mathrm{KL}\big(\pi_{\boldsymbol{\varphi}}(\cdot \mid \boldsymbol{x}_T) \| \hat{\pi}(\cdot \mid \boldsymbol{x}_T)\big) - \log Z(\boldsymbol{x}_T)\big]$$

is minimized for $\pi_{\boldsymbol{\varphi}} \equiv \hat{\pi}$. The KL-divergence then evaluates to zero, and we are left only with the $\log Z(\boldsymbol{x}_T)$ term.
For the inductive step, fix any $1 \leq t < T$ and $\pi^\star(\boldsymbol{a}_t \mid \boldsymbol{x}_t)$ must minimize the two terms

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{x}_t \sim \Pi_{\boldsymbol{\varphi}}} \big[\mathrm{KL}\big(\pi_{\boldsymbol{\varphi}}(\cdot \mid \boldsymbol{x}_t) \| \hat{\pi}(\cdot \mid \boldsymbol{x}_t)\big) - \log Z(\boldsymbol{x}_t)\big] \\
&\quad + \mathbb{E}_{(\boldsymbol{x}_t, \boldsymbol{a}_t) \sim \Pi_{\boldsymbol{\varphi}}} \Big[\mathbb{E}_{\boldsymbol{x}_{t+1} \sim p(\cdot \mid \boldsymbol{x}_t, \boldsymbol{a}_t)} [-\log Z(\boldsymbol{x}_{t+1})]\Big]
\end{aligned}$$

where the first term stems directly from the objective (12.92) and the second term represents the contribution of $\pi_{\boldsymbol{\varphi}}(\boldsymbol{a}_t \mid \boldsymbol{x}_t)$ to all subsequent terms. Letting $\beta^\star(\boldsymbol{a} \mid \boldsymbol{x}) \doteq \exp(q^\star(\boldsymbol{x}, \boldsymbol{a}))$, $Z^\star(\boldsymbol{x}) \doteq \int_{\mathcal{A}} \beta^\star(\boldsymbol{a} \mid \boldsymbol{x})\, d\boldsymbol{a}$, and recalling that we denote by $\pi^\star(\cdot \mid \boldsymbol{x})$ the policy $\beta^\star(\cdot \mid \boldsymbol{x}) / Z^\star(\boldsymbol{x})$, this objective can be reexpressed as

$$= \mathbb{E}_{\boldsymbol{x}_t \sim \Pi_{\boldsymbol{\varphi}}} \left[\mathrm{KL}\big(\pi_{\boldsymbol{\varphi}}(\cdot \mid \boldsymbol{x}_t) \| \pi^\star(\cdot \mid \boldsymbol{x}_t)\big) - \log Z^\star(\boldsymbol{x}_t) \right]$$

which is minimized for $\pi_{\boldsymbol{\varphi}} \equiv \pi^\star$, leaving only the $\log Z^\star(\boldsymbol{x}_t)$ term. It remains only to observe that $\beta^\star(\boldsymbol{a}_T \mid \boldsymbol{x}_T) = \beta(\boldsymbol{a}_T \mid \boldsymbol{x}_T)$, so in the final state $\boldsymbol{x}_T$, $\log Z^\star(\boldsymbol{x}_T) = \log Z(\boldsymbol{x}_T)$ and the policies $\pi^\star$ and $\hat{\pi}$ coincide.

Model-based Reinforcement Learning

Solution to exercise 13.14 (Bounding the regret of H-UCRL).

1. To simplify the notation, we use $z_{t,k}$ as shorthand for $(x_{t,k}, \pi_t(x_{t,k}))$ (and similarly $\widehat{z}_{t,k}$ for $(\widehat{x}_{t,k}, \pi_t(\widehat{x}_{t,k}))$). The base case is implied trivially. For the induction step, assume that eq. (13.28) holds at iteration k. We have

$$\begin{aligned} \left\|\widehat{x}_{t,k+1} - x_{t,k+1}\right\| &= \left\|\widehat{f}_{t-1}(\widehat{z}_{t,k}) - f(z_{t,k})\right\| \\ &\leq \left\|\widehat{f}_{n-1}(\widehat{z}_{t,k}) - f(\widehat{z}_{n,k})\right\| + \left\|f(\widehat{z}_{t,k}) - f(z_{t,k})\right\| \\ &\leq 2\beta_t \sigma_{t-1}(\widehat{z}_{t,k}) + L_1 \left\|\widehat{x}_{t,k} - x_{t,k}\right\| \end{aligned}$$

adding and subtracting $f(\widehat{z}_{n,k})$ and using Cauchy-Schwarz

where the final inequality follows with high probability from the assumption that the confidence intervals are well-calibrated (cf. eq. (13.22)) and the assumed Lipschitzness.

$$\begin{aligned} &= 2\beta_t [\sigma_{t-1}(z_{t,k}) + \sigma_{t-1}(\widehat{z}_{t,k}) - \sigma_{t-1}(z_{t,k})] \\ &\qquad + L_1 \left\|\widehat{x}_{t,k} - x_{t,k}\right\| \end{aligned}$$

Once more using Lipschitz continuity, we obtain

$$\begin{aligned} &\leq 2\beta_t \left[\sigma_{t-1}(z_{t,k}) + L_2 \left\|\widehat{x}_{t,k} - x_{t,k}\right\|\right] \\ &\qquad + L_1 \left\|\widehat{x}_{t,k} - x_{t,k}\right\| \\ &= 2\beta_t \sigma_{t-1}(z_{t,k}) + \alpha_t \left\|\widehat{x}_{t,k} - x_{t,k}\right\| \end{aligned}$$

where $\alpha_t \doteq (L_1 + 2\beta_t L_2)$. By the induction hypothesis,

$$\leq 2\beta_t \sum_{l=0}^{k} \alpha_t^{k-l} \sigma_{t-1}(z_{t,l}).$$

This is identical to the analysis of UCB from exercise 9.7, only that here errors compound along the trajectory.

2. The assumption that $\alpha_t \geq 1$ implies that

$$\left\|\widehat{x}_{t,k} - x_{t,k}\right\| \leq 2\beta_t \alpha_t^{H-1} \sum_{l=0}^{k-1} \sigma_{t-1}(z_{t,l}). \tag{B.15}$$

Moreover, by definition of π_t, we have with high probability that $J_H(\pi_t; \widehat{f}) \geq J_H(\pi^\star; f)$. This is because π_t maximizes reward under *optimistic* dynamics. Thus,

$$\begin{aligned}
r_t &= \sum_{k=0}^{H-1} r(x_{t,k}, \pi^\star(x_{t,k})) - r(x_{t,k}, \pi_t(x_{t,k})) \\
&\leq \sum_{k=0}^{H-1} r(\widehat{x}_{t,k}, \pi_t(\widehat{x}_{t,k})) - r(x_{t,k}, \pi_t(x_{t,k})) \\
&\leq L_3 \sum_{k=0}^{H-1} \left\|\widehat{x}_{t,k} - x_{t,k}\right\| && \text{using Lipschitz-continuity of } r \\
&\leq 2\beta_t \alpha_t^{H-1} L_3 \sum_{k=0}^{H-1} \sum_{l=0}^{k-1} \sigma_{t-1}(z_{t,l}) && \text{using eq. (B.15)} \\
&\leq 2\beta_t H \alpha_t^{H-1} L_3 \sum_{k=0}^{H-1} \sigma_{t-1}(z_{t,k}).
\end{aligned}$$

3. Let us first bound R_T^2. By the Cauchy-Schwarz inequality,

$$\begin{aligned}
R_T^2 &\leq T \sum_{t=1}^{T} r_t^2 \\
&\leq \mathcal{O}\left(T \sum_{t=1}^{T} \beta_t^2 H^2 \alpha_t^{2(H-1)} \left(\sum_{k=0}^{H-1} \sigma_{t-1}(z_{t,k}) \right)^2 \right) && \text{using (2)} \\
&\leq \mathcal{O}\left(T \beta_T^2 H^3 \alpha_T^{2(H-1)} \sum_{t=1}^{T} \sum_{k=0}^{H-1} \sigma_{t-1}^2(z_{t,k}) \right). && \text{using Cauchy-Schwarz and assuming w.l.o.g. that } \beta_t \text{ is monotonically increasing}
\end{aligned}$$

Taking the square root, we obtain

$$\begin{aligned}
R_T &\leq \mathcal{O}\left(\beta_T H^{\frac{3}{2}} \alpha_T^{H-1} \sqrt{T \sum_{t=1}^{T} \sum_{k=0}^{H-1} \sigma_{t-1}^2(z_{t,k})} \right) \\
&\leq \mathcal{O}\left(\beta_T H^{\frac{3}{2}} \alpha_T^{H-1} \sqrt{T \Gamma_T} \right).
\end{aligned}$$

Bibliography

Eitan Altman. *Constrained Markov decision processes: stochastic modeling*. Routledge, 1999.

Yarden As, Ilnura Usmanova, Sebastian Curi, and Andreas Krause. Constrained policy optimization via bayesian world models. *International Conference on Learning Representations*, 2022.

Dominique Bakry and Michel Émery. Diffusions hypercontractives. In *Séminaire de Probabilités XIX 1983/84: Proceedings*, pages 177–206. Springer, 2006.

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30, 2017.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

Ilija Bogunovic, Jonathan Scarlett, Andreas Krause, and Volkan Cevher. Truncated variance reduction: A unified approach to bayesian optimization and level-set estimation. *Advances in neural information processing systems*, 29, 2016.

Léon Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, pages 273–304, 1995.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691. PMLR, 2014.

Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In *International Conference on Machine Learning*, pages 844–853. PMLR, 2017.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Thomas Cover and Joy Thomas. *Elements of information theory*. Wiley-Interscience, 2006.

Sebastian Curi, Felix Berkenkamp, and Andreas Krause. Efficient model-based reinforcement learning through optimistic policy search and planning. *Advances in Neural Information Processing Systems*, 33:14156–14170, 2020.

Sebastian Curi, Armin Lederer, Sandra Hirche, and Andreas Krause. Safe reinforcement learning via confidence-based filters. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 3409–3415. IEEE, 2022.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472. Citeseer, 2011.

Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. *Mathematics for machine learning*. Cambridge University Press, 2020.

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

David Duvenaud and Ryan P Adams. Black-box stochastic variational inference in five lines of python. In *NIPS Workshop on Black-box Learning and Inference*, 2015.

Eyal Even-Dar and Yishay Mansour. Convergence of optimistic and incremental q-learning. *Advances in neural information processing systems*, 14, 2001.

Eyal Even-Dar, Yishay Mansour, and Peter Bartlett. Learning rates for q-learning. *Journal of machine learning Research*, 5(1), 2003.

Matthew Fellows, Anuj Mahajan, Tim GJ Rudner, and Shimon Whiteson. Virel: A variational inference framework for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Karl Friston. The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11(2):127–138, 2010.

Karl Friston, Francesco Rigoli, Dimitri Ognibene, Christoph Mathys, Thomas Fitzgerald, and Giovanni Pezzulo. Active inference and epistemic value. *Cognitive neuroscience*, 6(4):187–214, 2015.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Insights and applications. In *Deep Learning Workshop, ICML*, volume 1, page 2, 2015.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017.

Roman Garnett. *Bayesian optimization*. Cambridge University Press, 2023.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and David Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1487–1495, 2017.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta,

Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Jouni Hartikainen and Simo Särkkä. Kalman filtering and smoothing solutions to temporal gaussian process regression models. In *2010 IEEE international workshop on machine learning for signal processing*, pages 379–384. IEEE, 2010.

Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. *Advances in neural information processing systems*, 28, 2015.

Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(6), 2012.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. In *Artificial Intelligence and Statistics*, pages 351–360. PMLR, 2015.

José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. *Advances in neural information processing systems*, 27, 2014.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Matthew W Hoffman and Zoubin Ghahramani. Output-space predictive entropy search for flexible global optimization. In *NIPS workshop on Bayesian Optimization*, pages 1–5, 2015.

Thomas Hofmann. *Computational Intelligence Lab*. ETH Zurich, 2022.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *Conference on Uncertainty in Artificial Intelligence*, 2018.

Tommi Jaakkola, Michael Jordan, and Satinder Singh. Convergence of stochastic iterative dynamic programming algorithms. *Advances in neural information processing systems*, 6, 1993.

Edwin T Jaynes. *Probability theory: The logic of science*. Cambridge university press, 2003.

Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabás Póczos. Parallelised bayesian optimisation via thompson sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 133–142. PMLR, 2018.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

Johannes Kirschner and Andreas Krause. Information directed sampling and bandits with heteroscedastic noise. In *Conference On Learning Theory*, pages 358–384. PMLR, 2018.

Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. Learning-based model predictive control for safe exploration. In *2018 IEEE conference on decision and control (CDC)*, pages 6059–6066. IEEE, 2018.

Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability*, 3:71–104, 2014.

Andreas Krause and Fanny Yang. *Introduction to Machine Learning*. ETH Zurich, 2022.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Yi-An Ma, Yuansi Chen, Chi Jin, Nicolas Flammarion, and Michael I Jordan. Sampling can be faster than optimization. *Proceedings of the National Academy of Sciences*, 116(42):20881–20885, 2019.

David JC MacKay. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992.

Beren Millidge, Alexander Tschantz, Anil K Seth, and Christopher L

Buckley. On the relationship between active inference and control as inference. In *Active Inference: First International Workshop, IWAI 2020, Co-located with ECML/PKDD 2020, Ghent, Belgium, September 14, 2020, Proceedings 1*, pages 3–11. Springer, 2020.

Beren Millidge, Alexander Tschantz, and Christopher L Buckley. Whence the expected free energy? *Neural Computation*, 33(2):447–482, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *J. Mach. Learn. Res.*, 21(132):1–62, 2020.

Kevin P Murphy. Conjugate bayesian analysis of the gaussian distribution. *def*, 1(2σ2):16, 2007.

Jayakrishnan Nair, Adam Wierman, and Bert Zwart. *The Fundamentals of Heavy Tails*, volume 53. Cambridge University Press, 2022.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical programming*, 14:265–294, 1978.

Vu Nguyen, Sunil Gupta, Santu Rana, Cheng Li, and Svetha Venkatesh. Regret for expected improvement over the best-observed value and stopping condition. In *Asian Conference on Machine Learning*, pages 279–294. PMLR, 2017.

OpenAI. Gpt-4 technical report, 2023.

Manfred Opper and Cédric Archambeau. The variational gaussian approximation revisited. *Neural computation*, 21(3):786–792, 2009.

Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

Joaquin Quinonero-Candela and Carl Edward Rasmussen. A unifying

view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6, 2005.

Maxim Raginsky, Alexander Rakhlin, and Matus Telgarsky. Nonconvex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis. In *Conference on Learning Theory*, pages 1674–1703. PMLR, 2017.

Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *NIPS*, 2007.

Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial intelligence and statistics*, pages 814–822. PMLR, 2014.

Gareth O Roberts and Jeffrey S Rosenthal. General state space markov chains and mcmc algorithms. 2004.

Philip A Romero, Andreas Krause, and Frances H Arnold. Navigating the protein fitness landscape with gaussian processes. *Proceedings of the National Academy of Sciences*, 110(3):E193–E201, 2013.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Stuart Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice Hall, 2002.

Daniel Russo and Benjamin Van Roy. Learning to optimize via information-directed sampling. *Advances in Neural Information Processing Systems*, 27, 2014.

Daniel Russo and Benjamin Van Roy. An information-theoretic analysis of thompson sampling. *The Journal of Machine Learning Research*, 17(1):2442–2471, 2016.

Grant Sanderson. But what is the fourier transform? a visual introduction, 2018. URL `https://www.youtube.com/watch?v=spUNpyF58BY`.

Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer, 2001.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based pomdp solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1): 1–51, 2013.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Satinder Singh, Tommi Jaakkola, Michael L Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, 38(3):287–308, 2000.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning*, pages 1015—1022. ICML, 2010.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15 (1):1929–1958, 2014.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Bhavya Sukhija, Matteo Turchetta, David Lindner, Andreas Krause, Sebastian Trimpe, and Dominik Baumann. Gosafeopt: Scalable safe exploration for global optimization of dynamical systems. *Artificial Intelligence*, 320:103922, 2023.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Yee Whye Teh, Alexandre H Thiery, and Sebastian J Vollmer. Consistency and fluctuations for stochastic gradient langevin dynamics. *Journal of Machine Learning Research*, 17, 2016.

Michalis Titsias and Miguel Lázaro-Gredilla. Doubly stochastic variational bayes for non-conjugate inference. In *International conference on machine learning*, pages 1971–1979. PMLR, 2014.

Lenart Treven, Jonas Hübotter, Bhavya Sukhija, Florian Dörfler, and Andreas Krause. Efficient exploration in continuous-time model-based reinforcement learning. *Advances in neural information processing systems*, 2023.

Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. Safe exploration for interactive machine learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Sattar Vakili, Kia Khezeli, and Victor Picheny. On information gain and regret bounds in gaussian process bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 82–90. PMLR, 2021.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Santosh Vempala and Andre Wibisono. Rapid convergence of the unadjusted langevin algorithm: Isoperimetry suffices. *Advances in neural information processing systems*, 32, 2019.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pages 1058–1066. PMLR, 2013.

Yuhui Wang, Hao He, and Xiaoyang Tan. Truly proximal policy optimization. In *Uncertainty in Artificial Intelligence*, pages 113–122. PMLR, 2020.

Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. In *International Conference on Machine Learning*, pages 3627–3635. PMLR, 2017.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

Daniel Widmer, Dongho Kang, Bhavya Sukhija, Jonas Hübotter, Andreas Krause, and Stelian Coros. Tuning legged locomotion controllers via safe bayesian optimization. *arXiv preprint arXiv:2306.07092*, 2023.

Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press, 2006.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Stephen Wright, Jorge Nocedal, et al. Numerical optimization. *Springer Science*, 35(67-68):7, 1999.

Pan Xu, Jinghui Chen, Difan Zou, and Quanquan Gu. Global convergence of langevin dynamics based algorithms for nonconvex optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

Yinyu Ye. The simplex and policy-iteration methods are strongly polynomial for the markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36(4):593–603, 2011.

Acronyms

a.s. almost surely, with high probability, with probability 1

A2C advantage actor-critic

BALD Bayesian active learning by disagreement

BLR Bayesian linear regression

CDF cumulative distribution function

DDPG deep deterministic policy gradients

DDQN double deep Q-networks

DQN deep Q-networks

ECE expected calibration error

EI expected improvement

ELBO evidence lower bound

ES entropy search

FITC fully independent training conditional

GAE generalized advantage estimation

GD gradient descent

GLIE greedy in the limit with infinite exploration

GP Gaussian process

GPC Gaussian process classification

GRV Gaussian random vector

H-UCRL hallucinated upper confidence reinforcement learning

HMC Hamiltonian Monte Carlo

HMM hidden Markov model

i.i.d. independent and identically distributed

IDS information-directed sampling

iff if and only if

KL Kullback-Leibler

LAMBDA Lagrangian model-based agent

LASSO least absolute shrinkage and selection operator

LD Langevin dynamics

LMC Langevin Monte Carlo

LOTE law of total expectation

LOTP law of total probability

LOTUS law of the unconscious statistician

LOTV law of total variance

LSI log-Sobolev inequality

MAB multi-armed bandits

MALA Metropolis adjusted Langevin algorithm

MAP maximum a posteriori

MC Monte Carlo

MCE maximum calibration error

MCMC Markov chain Monte Carlo

MDP Markov decision process

MERL maximum entropy reinforcement learning

MES max-value entropy search

MGF moment-generating function

MI mutual information

MLE maximum likelihood estimate

MLL marginal log likelihood

MPC model predictive control

MSE mean squared error

NLL negative log likelihood

ODE ordinary differential equation

OPES output-space predictive entropy search

PBPI point-based policy iteration

PBVI point-based value iteration

PDF probability density function

PES predictive entropy search

PETS probabilistic ensembles with trajectory sampling

PI probability of improvement

PI policy iteration

PILCO probabilistic inference for learning control

PL Polyak-Łojasiewicz

PMF probability mass function

POMDP partially observable Markov decision process

PPO proximal policy optimization
RBF radial basis function
ReLU rectified linear unit
RHC receding horizon control
RKHS reproducing kernel Hilbert space
RL reinforcement learning
RM Robbins-Monro
SAA stochastic average approximation
SAC soft actor-critic
SARSA state-action-reward-state-action
SDE stochastic differential equation
SG-HMC stochastic gradient Hamiltonian Monte Carlo
SGD stochastic gradient descent
SGLD stochastic gradient Langevin dynamics
SLLN strong law of large numbers
SoR subset of regressors
SVG stochastic value gradients
SVGD Stein variational gradient descent
SWA stochastic weight averaging
SWAG stochastic weight averaging-Gaussian
Tanh hyperbolic tangent
TD temporal difference
TD3 twin delayed DDPG
TRPO trust-region policy optimization
UCB upper confidence bound
ULA unadjusted Langevin algorithm
VI variational inference
VI value iteration
w.l.o.g. without loss of generality
w.r.t. with respect to
WLLN weak law of large numbers

Index

Errata

October 6, 2023	fixed typo in eq. (1.61)
October 19, 2023	revised structure of chapter 5 and section 6.3
October 23, 2023	fixed typo in example 3.5
October 29, 2023	ensure existence of limit in eq. (5.56)
October 31, 2023	clarify index i in theorem 6.21
November 4, 2023	revised structure of chapters 2 and 7
November 11, 2023	fig. 8.4 was wrongly stating that uncertainty sampling picks the point maximizing "total" uncertainty
November 14, 2023	section 7.3.3 was incorrectly referring to "dropout" rather than "dropconnect"
November 14, 2023	fixed typo in eq. (9.14)