PML Lecture Notes for Lectures by Oswin

Oswin Krause

Probabilistic Machine Learning, November 2022

Contents

On this Document

This document is intended as a short script to provide a) the necessary background for the course and b) to provide lecture notes for the lectures given by Oswin. The background sections are 1 and 2 followed by the sections for individual lecture content.

The background part is written in a concise and compressed way to quickly look up basic relations and properties. It assumes that the reader has some familiarity with Machine Learning in general. For a less concise description, we refer to Bishop, Pattern Recognition and Machine-Learning, Chapters 1-4. Each section in this document contains references to the chapters in the book, even though our content has a different order than Bishop and is somewhat more formal.

Background material As probability theory stands on the shoulder of giants, we need a rich knowledge of different areas of math. This includes, among others, results from Linear Algebra: properties of matrix determinants, matrix inverses and matrix decompositions. We recommend the matrix cookbook as a quick reference as well as a look into your old material on Linear Algebra.

Structure We write proofs of statements in blocks

> **Proof:** The proof.

These blocks can be skipped on a first read through, even though we encourage the reader to work through them later on. Another important block are Notation blocks:

> **Notation:** We write ...

These should not be skipped as they will often change how we write equations in the following.

Errors This document is version 2.0. Even though we checked derivations and text several times, there can be errors in the document. We are happy to correct any error you find.

Contents Sections 1 and 2 are necessary background of probability distribution with focus on the multivariate normal distribution. We will come back to these properties often throughout the course. Chapter 3 and 4 are content of the first lecture and give an introduction to Bayesian and Frequentist probability as well as Generalized Linear Models.

1 Random Variables

Every algorithm interacting with the real world must be able to handle uncertainty. Uncertainty arises from a number of sources, be it incomplete or missing information, inaccuracies of a measurement or true randomness introduced by physical systems. Probability theory is the branch of mathematics that aims to formally describe uncertainty. A proper introduction to probability theory requires deep knowledge in analysis, topology and measure theory, which we can not provide in these notes. Instead, we will introduce the key concepts in a depth that is sufficient for this course. This means that sometimes we do not treat a topic with the necessary technical rigor required. We therefore assume that all operations (integrals etc.) are well-defined in the context we use them in. Still, these notes assume that the reader is somewhat familiar with the topic and serves mostly as a reminder and to introduce a common notation. An a little bit more basic introduction is given in (Bishop, chapters 1.2, pages 12-20).

The key concept of probability theory is that of a *random variable*. A random variable X models the process of how random observations are generated. Random variables take values on a state space Ω and an *observation* or *realization* of a random variable is a value $x \in \Omega$ that is observed as an outcome of a random experiment. If the set of values of X taken in Ω is countable, we call X a discrete random variable, otherwise, if X takes values that are continuous in Ω, we call X a continuous random variable.

Independent of the type of random variable, the probability that X takes a value in a given set $S \subseteq \Omega$ is $P(X \in S) \in [0, 1]$ and $P(X \in \Omega) = 1$. If X is discrete, the probability of observing value x is given by the *probability mass function* (pmf) $P(X = x) \geq 0$. The probability $P(X \in S)$ can be written in terms of the pmf by summing over all elements in S

$$P(X \in S) = \sum_{x \in S} P(X = x) \ . \tag{1}$$

If X is continuous, the probability of observing a specific event $P(X = x)$ is zero and thus the pmf can not be used to describe continuous random variables. Instead, we use the *probability density function* (pdf) $p(X = x) \geq 0$ and its relation to $P(X \in S)$ is given by integration

$$P(X \in S) = \int_S p(X = x)\, dx \ . \tag{2}$$

Notation: The process of defining a random variable is often written in a shortcut form. Given a pdf $p(x)$ we can write $X \sim p(x)$ to define a random variable X with samples distributed according to $p(x)$. Here, the state space Ω is defined implicitly via the input space of p. This notation is most commonly used in conjunction with known distributions. For example, a common shortcut to write the pdf of a normal distributed variable is $p(x) = \mathcal{N}(x; \mu, \sigma^2)$, where μ and σ^2 are the parameters of the distribution. Then $X \sim \mathcal{N}(x; \mu, \sigma^2)$ defines a random variable that follows a normal distribution with specified mean and variance. Another shortcut is to leave out the value range x and just write $X \sim \mathcal{N}(\mu, \sigma^2)$. The last way has the advantage that we can define random variables that do not have a pdf - something we will encounter just a few pages from now.

In the following, we will assume that random variables are continuous, if not stated otherwise. However, all definitions hold for discrete variables analogously.

The concept of a probability of a single random variable can be extended to a set of random variables. For two random variables X, Y with state spaces Ω_X, Ω_Y, the *joint probability* of observing a specific tuple of outcomes in a set $S \subseteq \Omega_X \times \Omega_Y$ is given by

$$P((X, Y) \in S) = \int_S p(X = x, Y = y)\, dx\, dy \ , \tag{3}$$

where $p(X = x, Y = y)$ stands for the pdf of the joint probability distribution. If only a subset of random variables is of interest, the joint distribution can be *marginalized* to obtain the probability of only the variables of interest. The resulting distribution is called the *marginal*. Sampling from

the marginal distribution is the same as sampling from the joint, while ignoring the values of the variables which are not of interest. In the pdf, marginalization is done via integration of all variables which are not of interest. For example, to obtain the marginal of X from the pair X, Y we compute the marginal pdf

$$p(X = x) = \int_{\Omega_Y} p(X = x, Y = y)\, dy \ . \tag{4}$$

We say that a set of random variables is independent, if their joint pdf can be written as the product of marginals

$$p(X = x, Y = y) = p(X = x)p(Y = y) \ . \tag{5}$$

Finally, we can compute the probability of X taking a specific value conditioned on a specific value of Y that we have already observed, $Y = y$. The *conditional distribution* is given by

$$p(X = x|Y = y) = \frac{p(X = x, Y = y)}{\int_{\Omega_X} p(X = x', Y = y)\, dx'} = \frac{p(X = x, Y = y)}{p(Y = y)} \tag{6}$$

The joint distribution can be written in terms of conditional and marginal distribution

$$p(X = x, Y = y) = p(X = x|Y = y)p(Y = y) = p(Y = y|X = x)p(X = x) \ , \tag{7}$$

and if two variables are independent, then the conditional distribution is the same as the marginal

$$p(X = x|Y = y) = P(X = x) \ . \tag{8}$$

Notation: There are a number of different notations in the literature for the pdf. Common notations include $p_X(x)$, $f_X(x)$ or simply $p(X)$ or $p(x)$. The latter is most commonly used in the machine-learning literature and often the differentiation between the random variable X and an outcome x is not made.
Similarly, there are several shortcuts to describe the relations between multiple random variables. For example the conditional $X|Y$ is itself taken as a random variable (the random variable of X after observing Y).
In the following, we will write $p(x)$ instead of $p(X = x)$ if it is clear from the context, which random variable we mean.

1.1 Transformations and Expectations of random variables

If we observe a random variable X taking values on state-space $\Omega_X = \mathbb{R}^d$ several times, we obtain realizations $x_1, x_2, \ldots, x_N \in \mathbb{R}^d$. Their sample average is given by

$$\hat{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \ . \tag{9}$$

The mean of X, $E[X]$ is the integral

$$E[X] = \int_{\Omega_X} p(x) x dx \ . \tag{10}$$

The relation between the sample average and the mean is given by the *weak law of large numbers*: If X has finite variance, then in the limit of infinite samples the sample average converges to the mean

$$E[X] = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} x_i \ . \tag{11}$$

We can transform a random variable given a function $f : \Omega_X \to \Omega_Z$. Applying f to an observation x of X gives a new random variable Z with realizations, $z = f(x)$. As shortcut, we write

$$Z = f(X) \ . \tag{12}$$

In the case that $\Omega_X = \Omega_Z = \mathbb{R}^d$ and f is a bijective function, we can write the probability density of Z in terms of the inverse function $f^{-1} : \Omega_Z \to \Omega_X$ and the determinant of its Jacobian $J_{f^{-1}}(z)$

$$p(z) = p(X = f^{-1}(z))|\det J_{f^{-1}}(z)| \ . \tag{13}$$

The Jacobian of a function g is the matrix of partial derivatives:

$$J_g(x) = \begin{pmatrix} \frac{\partial g_1(x)}{\partial x_1} & \cdots & \frac{\partial g_1(x)}{\partial x_d} \\ \vdots & \cdots & \vdots \\ \frac{\partial g_d(x)}{\partial x_1} & \cdots & \frac{\partial g_d(x)}{\partial x_d} \end{pmatrix} \tag{14}$$

Proof: This can be derived directly using integration by substitution, which in the multi-dimensional case reads:

$$\int_{g(S)} h(x)\ dx = \int_S h(g(z))|\det J_g(z)|\ dz \ ,$$

assuming that g is bijective and has a continuous derivative. The reader might not be familiar with the multi-variate version and can consider the case one-dimensional case, $d = 1$ instead.

Let $S \subseteq \Omega_Z$. The probability of $P(Z \in S)$ is given by:

$$\begin{aligned} P(Z \in S) &= \int_S p(Z = z)\ dz \\ &= \int_{f^{-1}(S)} p(X = x)\ dx \\ &= \int_S p(X = f^{-1}(z))|\det J_{f^{-1}}(z)|\ dz \end{aligned}$$

In the first step we used, that due to bijectivity, observing a value z of random variable Z entails observation of a unique value x of random variable X. Thus $P(Z \in S) = P(X \in f^{-1}(S))$. The next step is direct application of integration by substitution for multiple variables.
We obtain the final result by comparing the first and last integral point-wise.

With a similar derivation, we can show that the expectation of Z can be written directly in terms of f and X,

$$E[Z] = \int_{\Omega_Z} p(Z = z)z\ dz = \int_{\Omega_X} p(X = x)f(x)\ dx \ . \tag{15}$$

This relation is useful, because often it is much easier to compute the integral of the right hand side, than to start with the left hand side. It is worth noting that this relation holds under much weaker assumptions than required by the change of variable formula and invertibility of f is not required. Indeed, the assumptions are so weak that most statisticians use it without checking the requirements (or even knowing of their existence), which led to its name of *the law of the unconscious statistician* (LOTUS). It is fair to say, that LOTUS will hold for all functions considered in machine-learning.

Notation: We will introduce another helpful shortcut and directly write $E[f(X)]$ instead of $E[Z]$. Further, if more than one random variable appears in an expression and we don't intend to take the expectation over all, we specify the random variable to integrate over. For example, given two random variables X, Y and the transformed variable $Z = f(X, Y)$, then $E_Y[f(X, Y)]$ only computes the expectation over Y, keeping X constant. The result is still a random variable as a transformation of X.

We will state a few important properties of the expectation. Let X, Y be two independent d-dimensional random variables. Then for the sum $X + Y$ it holds:

$$E[X + Y] = E[X] + E[Y] \tag{16}$$

For their product, it holds:

$$E[X \cdot Y] = E[X] \cdot E[Y] \tag{17}$$

For transformed variables, it is difficult to find an expression in the general case. However, if f is an affine linear function, then

$$E[f(X)] = f(E[X]) \ . \tag{18}$$

When f is a convex function, then we have *Jensen's Inequality*:

$$E[f(X)] \geq f(E[X]) \ . \tag{19}$$

While the expectation gives a description for the average location of a sample, the variance is used to obtain an estimate of the spread. For a random variable X with state-space $\mathbb{R}$ the variance can be defined as the expected squared distance from the mean:

$$\mathrm{Var}(X) = E[(X - E[X])^2] \ . \tag{20}$$

A way to measure dependency of X on another real-valued variable Y is the covariance

$$\mathrm{Cov}(X, Y) = E[(X - E[X])(Y - E[Y])] \ . \tag{21}$$

The covariance measures a linear relationship between X and Y: if X becomes larger, then Y tends to change as well and its expected rate of change is given by $\mathrm{Cov}(X, Y)$. Computing the covariance of a variable with itself gives the variance, $\mathrm{Var}(X) = \mathrm{Cov}(X, X)$. Further, if X and Y are independent their covariance is 0, due to property (17) of the mean. The converse does not hold as dependent variables can have covariance 0.

For a random rector $X \in \mathbb{R}^d$, the set of variances and covariances between all pairs of variables in X can be computed using the covariance matrix:

$$\mathrm{Cov}(X) = \begin{pmatrix} \mathrm{Cov}(X_1, X_1) & \dots & \mathrm{Cov}(X_1, X_d) \\ \vdots & \dots & \vdots \\ \mathrm{Cov}(X_d, X_1) & \dots & \mathrm{Cov}(X_d, X_d) \end{pmatrix} = E[(X - E[X])(X - E[X])^T] \ .$$

The diagonal elements of the covariance matrix give the variances of the variables and the off-diagonal elements each give the covariance between pairs of elements.

2 The Normal distribution

Due to it importance to machine-learning, we will quickly reintroduce the key concepts of the normal distribution. We will then extend it to the multivariate case and show some of its most important properties. Many of these properties will be usable throughout the course, even for very advanced non-linear models. As additional reading, we refer to (Bishop, chapter 2.3-2.3.2, pages 78-90). As always, the reader is encouraged to skip the proofs at a first read through.

We call a random variable X with state space $\mathbb{R}$ normal distributed with mean μ and variance σ^2, or in short $X \sim \mathcal{N}(\mu, \sigma^2)$, if it has pdf

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{\sigma^2}} \tag{22}$$

Notation: The normal distribution is an example of a parameterized distribution. If it is clear in the context which parameters and value swe mean, we will just write $p(x)$. If we want to clarify the existence or choice of parameters, we will write them in form of a conditional pdf, for example $p(x|\mu, \sigma^2)$.

The normal distribution has a few important properties:

Affine transformations Let $\epsilon \sim \mathcal{N}(0, 1)$, $a, b \in \mathbb{R}$, then

$$Z = a + b\epsilon \tag{23}$$

is a normal distributed random variable $Z \sim \mathcal{N}(a, b^2)$.

Summation Let $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$, $Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$, then

$$Z = X + Y \tag{24}$$

is a normal distributed random variable $Z \sim \mathcal{N}(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)$.

Notation: For the second result it is important that both variables, X and Y are independent. Following common ML notation, we treat independently defined random variables as independent if not mentioned otherwise. If we assume dependence, we denote this by making the parameters of one variable a function of the other, e.g., $Y \sim \mathcal{N}(\mu_Y(X), \sigma_Y^2(X))$.

A result of these two properties is that any linear combination of normal distributed random variables is normally distributed. We extend the definition to the multivariate distribution by focusing on this property:

Definition: Let $\epsilon \in \mathbb{R}^N$ be a random variable with elements distributed as $\epsilon_i \sim \mathcal{N}(0, 1)$, $i = 1, \ldots, N$. Further, let $A \in \mathbb{R}^{d \times N}$, $\mu \in \mathbb{R}^d$. A random variable of the form

$$X = \mu + A\epsilon \tag{25}$$

is called Multivariate Normal Distributed with mean $E[X] = \mu$ and variance $\Sigma = AA^T$, or in short

$$X \sim \mathcal{N}(\mu, \Sigma) \ .$$

This definition is indirect as it gives us neither a density nor a distribution function, but it provides a direct way to create normal distributed random variables. A more commonly used definition is the following:

PDF of the Multivariate Normal Distribution: Let $X \sim \mathcal{N}(\mu, \Sigma)$, where $\mu \in \mathbb{R}^d$ and $\Sigma \in \mathbb{R}^{d\times d}$ is a symmetric positive definite matrix. Then X has pdf

$$p(x) = \frac{1}{\sqrt{2\pi}^d \sqrt{\det \Sigma}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \quad . \tag{26}$$

These two definitions are not equivalent, because the second definition only encompasses a subset of normal distributed variables: the first definition allows a choice of A such, that $\Sigma = AA^T$ is only positive semi-definite, which means that $\det \Sigma = 0$. In this case, equation (26) includes a division by zero. This is why typically in a machine-learning course only the second version is presented: handling of normal distributions with semi-definite covariance matrix is more difficult. We give the full definition here because a proper handling of Gaussian Processes later in this course requires the full definition, but unless explicitly stated, we will assume that X has a pdf. For completeness, we need to show that the pdf described by (26) belongs to a random variable that can be created via (25). The statement we want to show is:

Let X be a multivariate normal distributed random variable following (25). If A is invertible, then X has pdf (26). Conversely, if X has pdf (26), then it can be written in form (25) with A invertible.

Proof: We will begin with the first part of the statement. First of all, we note that invertibility of A requires that $N = d$ and thus A is a $d \times d$ matrix. Since the elements of ϵ are independent normally distributed variables, its pdf is

$$p(\epsilon) = \prod_{i=1}^{d} p(\epsilon_i) = \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\epsilon_i^2} = \frac{1}{\sqrt{2\pi}^d} \exp\left(-\frac{1}{2}\epsilon^T \epsilon\right)$$

By (25), X can be written as transformation of a random variable using the affine function $f(x) = \mu + Ax$ and $X = f(\epsilon)$. Since A is invertible, f is an invertible function with inversion $f^{-1}(x) = A^{-1}(x-\mu)$ and Jacobian $J_{f^{-1}}(x) = A^{-1}$. This allows us to use the change of variables formula (13)

$$\begin{aligned} p(X = x) &= p(\epsilon = f^{-1}(x))|\det J_{f^{-1}}(x)| \\ &= p(\epsilon = A^{-1}(x-\mu))|\det(A^{-1})| \\ &= \frac{\det A^{-1}}{\sqrt{2\pi}^d} \exp\left(-\frac{1}{2}(x-\mu)^T (A^{-1})^T A^{-1}(x-\mu)\right) \end{aligned}$$

To write this in terms of Σ, we first note that $\Sigma^{-1} = (A^{-1})^T A^{-1}$. To write $|\det J_{f^{-1}}(x)|$ in terms of Σ, we use properties of the determinant and obtain $\det \Sigma = \det(AA^T) = (\det A)^2$ and $|\det J_{f^{-1}}(x)| = 1/|\det A| = 1/\sqrt{\det \Sigma}$. Inserting this in the previous result gives (26).

To show the opposite direction, we need to show that for every Σ a matching A exists. This can be done using the eigenvalue decomposition. Since Σ is symmetric positive definite, it has a symmetric eigenvalue composition

$$\Sigma = \sum_{i=1}^{d} \lambda_i v_i v_i^T \quad ,$$

where λ_1 are the eigenvalues and v_i the corresponding eigenvectors with property $v_i^T v_i = 1$ and $v_i^T v_j = 0$, $i \neq j$. Since Σ is positive definite, $\lambda_i > 0$. We now construct A as

$$A = \sum_{i=1}^{d} \sqrt{\lambda_i} v_i v_i^T \quad .$$

It can be verified easily, that $AA^T = \Sigma$ and A is invertible as $\sqrt{\lambda_i} > 0$

The relation between (25) and the pdf (26) is important, because in practice it it often easier to show that a random variable has a multivariate normal distribution via (25), than to manipulate the pdf and its integrals. We will use this approach in the following to derive a few of the most important properties of multivariate normal random variables: Their mean and variance and rules for calculating the marginals, conditionals, projections and summation.

Standard Multivariate normal Let $\epsilon \in \mathbb{R}^d$ be a random vector with elements $\epsilon_i \sim \mathcal{N}(0,1)$. Then, ϵ follows a multivariate normal distribution, $\epsilon \sim \mathcal{N}(0, I_d)$, where 0 is the d-dimensional zero vector and I_d the d-dimensional identity matrix. This distribution is called he standard multivariate normal distribution.

Proof: This follows directly from property (25) with $\mu_X = 0$ and $A = I_d$.

Mean and Covariance Let $X \sim \mathcal{N}(\mu_x, \Sigma_X)$. Its mean is

$$E[X] = \mu \tag{27}$$

and its covariance matrix is

$$\text{Cov}(X) = \Sigma_X \tag{28}$$

Proof: We start by the observation of (25), that there exists an A, such, that $X = \mu_X + A\epsilon$, where $\epsilon \sim \mathcal{N}(0, I_d)$.
Then using property (18) of the mean we obtain:

$$E[X] = E[\mu_X + A\epsilon] = \mu_X + AE[\epsilon] = \mu_X$$

The last step holds, since the mean of a univariate standard normal variable is 0 and thus the vector is the zero-vector.

$$\begin{aligned}
\text{Cov}[X] &= E[(X - E[X])(X - E[X])^T] \\
&= E[(\mu_X + A\epsilon - \mu_X)(\mu_X + A\epsilon - \mu_X)^T] \\
&= E[A\epsilon\epsilon^T A^T] \\
&= A \underbrace{E[\epsilon\epsilon^T]}_{I_d} A^T = \Sigma_X
\end{aligned}$$

In the first step, we used equation (25) and the value for the mean of X we computed earlier. Then we simplified and applied property (18) of the mean. Finally, we used that the variance of standard normally distributed random variables is 1 and covariance between independent variables is 0 and thus $E[\epsilon\epsilon^T]$ is the identity matrix.

Marginal and Conditional distribution Let $X \sim \mathcal{N}(\mu_x, \Sigma_X)$ be a d-dimensional multivariate random variable. We consider marginalizing a subset of these variables, or computing the conditional distribution of a subset of variables. For simplicity we split X in two blocks: $X_1 \in \mathbb{R}^k$ and $X_2 \in \mathbb{R}^{d-k}$. With this block split, we can write both Σ_X and μ_X in block notation

$$X = \left[\begin{array}{c} X_1 \\ \hline X_2 \end{array}\right], \quad \mu_X = \left[\begin{array}{c} \mu_1 \\ \hline \mu_2 \end{array}\right], \quad \Sigma_X = \left[\begin{array}{c|c} \Sigma_{11} & \Sigma_{12} \\ \hline \Sigma_{21} & \Sigma_{22} \end{array}\right]$$

Then, the marginal distribution of X_1 is $\mathcal{N}(\mu_1, \Sigma_{11})$ and the conditional distribution $X_2|X_1$ with pdf $p(x_2|x_1)$ is $\mathcal{N}(\mu_{2|1}, \Sigma_{2|1})$, where

$$\mu_{2|1} = \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(X_1 - \mu_1), \qquad \Sigma_{2|1} = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{21}^T \tag{29}$$

The results for the marginal of X_2 and the conditional $X_1|X_2$ can be obtained by permuting the order of X_1 and X_2 in the random vector and exchanging the blocks of μ_X and Σ_X accordingly.

Proof: We start by the observation of (25), that there exists an A, such, that $X = \mu_X + A\epsilon$ where $\epsilon \sim \mathcal{N}(0, I_d)$. Moreover, it can be shown that there exist an A that fulfills the form

$$A = \left[\begin{array}{c|c} A_{11} & 0 \\ \hline A_{21} & A_{22} \end{array}\right], \text{ with } A_{22} \text{ invertible.}$$

(This can be shown constructively using the Cholesky decomposition algorithm. We skip this part for brevity of presentation.)

To obtain a more concrete description of A we relate the values using $\Sigma_X = AA^T$ and using blockwise matrix multiplication, we have

$$\begin{aligned}\left[\begin{array}{c|c} \Sigma_{11} & \Sigma_{12} \\ \hline \Sigma_{21} & \Sigma_{22} \end{array}\right] &= \left[\begin{array}{c|c} A_{11}A_{11}^T + 0 \cdot 0 & A_{11}A_{21}^T + 0 \cdot A_{22}^T \\ \hline A_{21}A_{11}^T + A_{22} \cdot 0 & A_{22}A_{22}^T + A_{21}A_{21}^T \end{array}\right] \\ &= \left[\begin{array}{c|c} A_{11}A_{11}^T & A_{11}A_{21}^T \\ \hline A_{21}A_{11}^T & A_{22}A_{22}^T + A_{21}A_{21}^T \end{array}\right]\end{aligned}$$

By comparing both sides block by block, we obtain the relations

$$\begin{aligned} A_{11}A_{11}^T &= \Sigma_{11} \\ A_{21} &= \Sigma_{21}(A_{11}^T)^{-1} \\ A_{22}A_{22}^T &= \Sigma_{22} - A_{21}A_{21}^T = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{21}^T \end{aligned}$$

With that, $X = \mu_X + A\epsilon$ can be written in block notation using the same split of ϵ in ϵ_1 and ϵ_2:

$$\left[\begin{array}{c} X_1 \\ \hline X_2 \end{array}\right] = \left[\begin{array}{c} \mu_1 \\ \hline \mu_2 \end{array}\right] + \left[\begin{array}{c|c} A_{11} & 0 \\ \hline A_{21} & A_{22} \end{array}\right] \left[\begin{array}{c} \epsilon_1 \\ \hline \epsilon_2 \end{array}\right] \tag{30}$$

Comparing the left and right side reveals $X_1 = \mu_1 + A_{11}\epsilon_1$ and thus $X_1 \sim \mathcal{N}(\mu_1, \Sigma_{11})$ by equation (25). This shows the marginalization part.

For the conditional distribution $X_2|X_1$, we assume that X_1 is observed and therefore fixed. Thus $X_2|X_1$ are all samples from the joint that lead to this value of X_1. When considering this in the generating transformation (30), this means as X_1 appears on the left side, we have to compute its effect on the right hand side. We have shown already, that $X_1 = \mu_1 + A_{11}\epsilon_1$, thus with X_1 fixed, ϵ_1 is fixed as well and

$$\epsilon_1 = A_{11}^{-1}(X_1 - \mu_1) \ .$$

The same equation for X_2 is given by the second row of (30) and

$$X_2 = \mu_2 + A_{21}\epsilon_1 + A_{22}\epsilon_2 = \mu_2 + A_{21}A_{11}^{-1}(X_1 - \mu_1) + A_{22}\epsilon_2 \ .$$

Thus, X_2 follows a multivariate distribution by (25) with mean $\mu_{2|1} = \mu_2 + A_{21}A_{11}^{-1}(X_1 - \mu_1)$ and covariance $\Sigma_{2|1} = A_{22}A_{22}^T$. Using the three value relations between A and Σ_X above, we obtain the result.

Joint of dependent normals with shifted means Let $X \sim \mathcal{N}(\mu_X, \Sigma_X)$ be a d-dimensional multivariate random variable and $Y|X \sim \mathcal{N}(\mu_Y + AX, \Sigma_Y)$. Then the joint distribution of both variables is

$$\left[\begin{array}{c} X \\ \hline Y \end{array}\right] \sim \mathcal{N}\left(\left[\begin{array}{c} \mu_X \\ \hline \mu_Y + A\mu_X \end{array}\right], \left[\begin{array}{c|c} \Sigma_X & \Sigma_X A^T \\ \hline A\Sigma_X & \Sigma_Y + A\Sigma_X A^T \end{array}\right]\right) \ . \tag{31}$$

This statement is the inverse of the previous statement: In (29) we obtained that $X_2|X_1$ is normally distributed with its only dependence on the value of X_1 being the shifted mean. Now we show,

that two multivariate normal distributed random variables that depend on each other by a mean shift are jointly multivariate normal.

Proof: The proof follows by stating (25) for the joint distribution.

Closed under linear Transformations Let $X \sim \mathcal{N}(\mu_x, \Sigma_X)$ be a d-dimensional multivariate random variable. Further, let $Q \in \mathbb{R}^{k \times d}$, then

$$Z = QX$$

is a multivariate normal random variable with $Z \sim \mathcal{N}(Q\mu_X, Q\Sigma_X Q^T)$

Proof: Let $\epsilon \sim \mathcal{N}(0, I_d)$. Further, let A_X be a matrix such, that $\Sigma_X = A_X A_X^T$. Then $X = \mu_X + A_X\epsilon$ and

$$Z = QX = Q(\mu_X + A_X\epsilon) = \underbrace{Q\mu_X}_{\mu_Z} + \underbrace{QA_X}_{A_Z}\epsilon = \mu_Z + A_Z\epsilon.$$

This meets the definition of a multivariate normal distribution by (25) and Z has covariance

$$\Sigma_Z = A_Z A_Z^T = QA_X A_X^T Q^T = Q\Sigma_X Q^T \quad .$$

Sum of Multivariate Random Variables Let $X \sim \mathcal{N}(\mu_x, \Sigma_X)$, $Y \sim \mathcal{N}(\mu_y, \Sigma_Y)$ be d-dimensional multivarate random variables. Then,

$$Z = X + Y \tag{32}$$

is a multivariate normal random variable with $Z \sim \mathcal{N}(\mu_X + \mu_Y, \Sigma_X + \Sigma_Y)$.

Proof: We can construct the result directly from the previous result on linear transformations. We treat X, Y as two blocks of a $2d$ dimensional random variable T and it is straight forward to show that T is a multivariate normal distributed random variable (e.g., as special case of the joint of dependent normals property) with

$$T = \left[\begin{array}{c} X \\ \hline Y \end{array}\right] \sim \mathcal{N}\left(\left[\begin{array}{c} \mu_X \\ \hline \mu_Y \end{array}\right], \left[\begin{array}{c|c} \Sigma_X & 0 \\ \hline 0 & \Sigma_Y \end{array}\right]\right) \quad .$$

We define as linear transformation the block-matrix constructed by concatenating two d-dimensional identity matrices I_d, $Q = [I_d | I_d] \in \mathbb{R}^{d \times 2d}$ and $Z = Q \cdot T = X + Y$. Then, using the projection property, we obtain the end result.

3 Interpretations of Probability

While probability theory introduces the formal foundations for handling uncertainty and randomness, mathematics is not directly concerned with the real world. The formal definition of probability does not tell us how to interpret the values we compute. Two fundamentally different interpretations of probabilities are actively used in Machine Learning: the Frequentist and the Bayesian perspective. The choice of perspective can lead to quite different methods for modeling data.

Let us start with two simple examples to illustrate the problem. Consider first that we want to throw a fair coin 100 times. Before throwing the coin, we can ask "What is the probability that a fair coin will come up heads at least 60 times?". We then throw the coin 100 times and observe that it comes up heads 63 times. This is an unlikely event, so an immediate question arising is: "After observing the outcome, what is the probability that the coin is fair?".

In both questions, we use the word probability with a different meaning. In the first question, probability refers to the *frequency* of an outcome. If we repeated this experiment 1000 times, we would expect that the coin comes up heads more than 60 times in around 2.8% of repetitions and the weak law of large numbers tells us that as the number of repetitions increase, we expect less of a deviation from this number. In this sense, this interpretation of probability is a verifiable fact (at least in our experiment, where we have the luxury that we can repeat it numerous times).

In the second question, we interpret probability as an *uncertainty over a held belief*. The coin might be fair but we have observed an unlikely event providing evidence that the coin gives heads more often than tails and this observation changes our belief over the coins fairness. Most importantly, this interpretation of probability does not refer to verifiable facts - the coin is either fair or biased, independent of our beliefs.

In the real world, these two interpretations of probability exist side by side in our decision processes. A recent scientific example was the possible observation of neutrinos travelling faster than light by the OPERA collaboration in 2011. Neutrinos were produced at CERN in Geneva and sent towards a detector at Gran Sasso in Italy. The time-difference between time of production and time of observation provides an estimate of the speed at which the particles travel. This observation is noisy, since at both ends the instruments only have limited precision. Therefore the experiment was repeated a large number of times. The result was that neutrinos moved too quickly: they arrived on average around 60.7ns faster than they would if they traveled with speed of light and the time difference was several time larger than what was expected due to random measurement imprecision.

The first reaction of the scientific community was, that this was most likely a data fluke or an error in the experiment. So the collaboration tested everything, repeated the experiment and gathered enough data so that the probability of observing a random fluctuation this large was less than 1 in 10 Million - the common threshold for an effect to be considered a scientific discovery in particle physics. This only lead many scientists to *belief stronger* that there is an error in the experiment. And they were right. Half a year later, a loose cable was found that slowed down the time signal of the high-precision clock. A repetition of the experiment then verified that the new expected travel time was within the error margin of the experiment assuming that neutrinos are not traveling faster than the speed of light.

This example sheds light on the two interpretations of probability. In the first, we only consider uncertainty over verifiable facts: is the difference in travel time larger than we would expect? It does neither provide evidence towards the reason why these differences are observed, nor does it provide direct guidance towards which decision to take: it could be either a real effect or the experiment is faulty and there is no information that can discern between the two.

In the second interpretation, our beliefs of the truth are affected by our prior belief: evidence towards faster than light neutrinos lead scientists to belief that the experiment is faulty, because their *prior* belief of the world is that the probability of an error in the experiment is much larger than a particle traveling faster than light. This is also why gathering more experimental data using the same experimental equipment did not sway their opinion: gathering more data only rules out the chance of observing a random fluctuation, but does not rule out persistent errors in the measurement equipment.

3.1 Bayesianism and Frequentism

The two aforementioned interpretations each have a name in the scientific community. Approaches that interpret probabilities as the frequency of an event are called *Frequentist* and statisticians who favour this interpretation are called *Frequentists*. Approaches using the second interpretation are called *Bayesian* and statisticians using these approaches are called *Bayesians*. This name refers to the importance of Bayes' Theorem to incorporating the prior belief into the evidence. Still, the use of Bayes' theorem does not make a theory Bayesian, as it is a fact derived from probability theory and used by Frequentists as well - it is the interpretation of the resulting probabilities as a belief, which is the important distinction.

There are very strong opinions in the statistics community about which approach is the "better" one. This is because both approaches can be used to make decisions: While Frequentism does not model decisions directly, a Frequentist can still decide to adapt the option that is supported most by the data. For example, we can decide that a coin should be discarded and replaced by a new if it lands heads more than 60 times out of a 100. This decision would at worst discard 1.7% of fair coins on average, while discarding most of strongly biased coins (e.g. a coin with probability of heads $\geq 65\%$ has at least a 82.7% chance of being discarded) . This decision process does not entail a belief over whether a discarded coin is biased or not: depending on the actual frequency of biased coins, almost none or all discarded coins might be fair.

A Bayesian would incorporate their prior belief about the relative proportions of fair and unfair coins into the model to obtain a final belief that the decision is based on. The decision would then be to only keep coins which are believed to be fair (e.g. $P(\text{Coin is fair}|\text{Data}) > 0.8$). The Frequentist would argue that by incorporating the prior belief, the decision becomes subjective: it is not only the evidence that partakes in the decision, but also the unverifiable prior. The Bayesian would in turn argue that not using prior knowledge sometimes leads to nonsensical decisions, like adapting that Neutrinos might travel faster than light (to which the Frequentist would argue that of course they would not favour that and instead would propose to gather data via an independent replication to rule out an error in this experiment. By this point, the reader might guess, that these discussions between Bayesians and Frequentists are lively, but rarely fruitful).

In the following, we will try to keep a neutral view on the two approaches. While this topic is also covered in (Bishop, chapters 1.2.3, 1.5, 3.4), it is important to keep in mind that Bishop declares himself a Bayesian and he is arguing for the adaptation of the Bayesian approach. Similarly, while this text tries to stay neutral, it is written by someone who was educated in the Frequentist view. We will introduce the two approaches in a formal way and shed light on their predictions. As a running example, we will use the case of logistic regression in a binary classification problem. We assume, we are given a Dataset $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(\ell)}, y^{(\ell)})\}$, where the $x^{(i)} \in \mathbb{R}^2$ are the inputs and $y^{(i)} \in \{0, 1\}$ are the labels assigned to point $x^{(i)}$. Our goal is to find the parameters θ of a linear model $f_\theta(x) = \theta_1 x_1 + \theta_2 x_2 + \theta_3$ that separates between points of both classes. The probability of a label y given a point x in this model is

$$p(y = 1|x, \theta) = \text{sigmoid}(\theta_1 x_1 + \theta_2 x_2 + \theta_3) \tag{33}$$

A visualisation of this data and task is given in Figure 1.

3.2 The Bayesian view

In the Bayesian view we assume that the unknown parameter vector θ is a random variable symbolizing the distribution of our beliefs. Our initial beliefs about the parameter before seeing any data are encoded via the *prior distribution* $p(\theta)$. The dataset $\mathcal{D}$ is considered fixed: this is the evidence that we use to update our prior belief $p(\theta)$ to the *posterior* belief $p(\theta|\mathcal{D})$.

We assume that $\mathcal{D}$ is produced by a data generative process. For example, in logistic regression, we assume that a learning problem and its dataset are generated by first sampling the unknown model parameters θ from $p(\theta)$, then sample datapoints $x^{(i)}$ from a distribution $p(x)$ and then acquire labels using a distribution $p(y|x, \theta)$, or

$$p(\mathcal{D}, \theta) = p(\theta)p(\mathcal{D}|\theta) = p(\theta) \prod_{i=1}^{\ell} p(x^{(i)})p(y^{(i)}|x, \theta) \ . \tag{34}$$

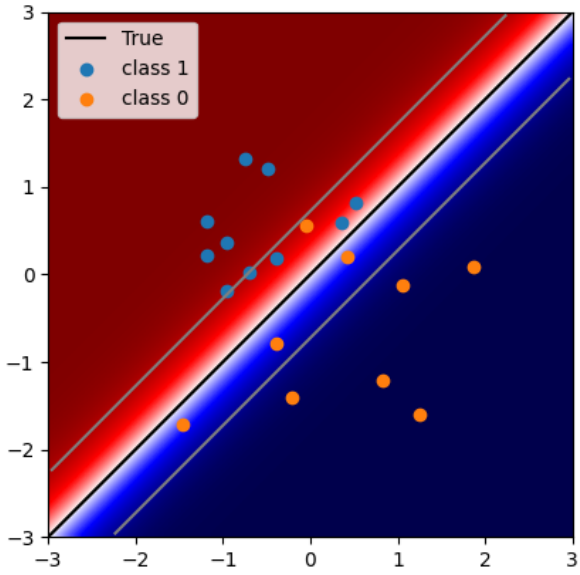


Figure 1: Visualization of our logistic regression toy experiment. The samples drawn (points) are assigned labels (colors) based on the class-probability $p(y = 1|x, \theta)$ (background coloring) for a chosen value of θ. The decision boundary of the classifier is shown as the black line labeled "True". Blue background means $p(y = 0|x, \theta) > p(y = 1|x, \theta)$, red means the opposite.

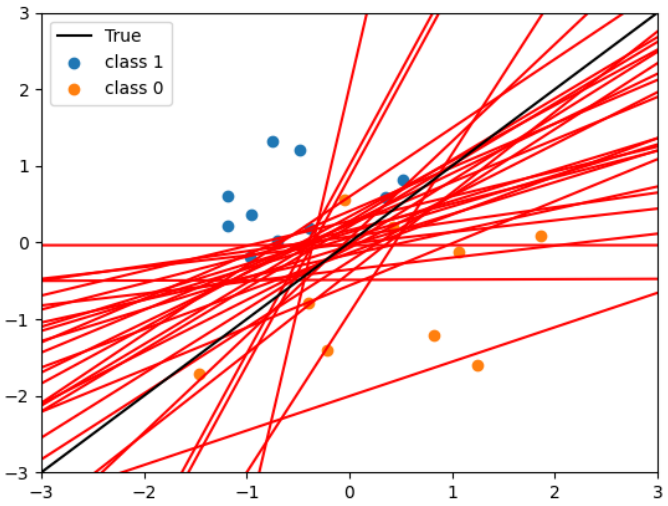
Since we do not observe θ in this generative process and are only left with the dataset, we can only compute our belief over the distribution of the parameters after observing the data. Assuming that our model (34) is correct, we compute $p(\theta|\mathcal{D})$ using Bayes' Theorem :

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}, \theta)}{\int_{\Omega_\theta} p(\mathcal{D}, \theta')d\theta'} = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \quad . \tag{35}$$

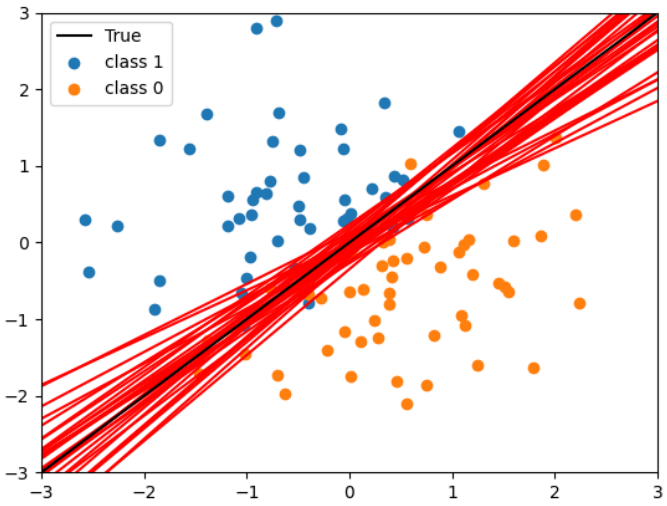
The quantity $p(\mathcal{D})$ is constant in the model. It is the normalisation constant that ensures that $\int p(\theta|\mathcal{D})d\theta = 1$. This quantity is often very difficult to compute.

To visualize the distribution using logistic regression, we can obtain samples from $p(\theta|\mathcal{D})$ (with algorithms covered later in the course) and plot the resulting models. As prior $p(\theta)$ we use a standard normal distribution for each parameter. In Figure 2a we show the decision boundaries obtained for different sampled values of θ given a dataset with $\ell = 20$ elements. The red lines mark the points where $p(y = 1|x, \theta) = 1/2$. There is quite some variation in the obtained samples, even though a majority is pointing in the correct direction and pass in-between the data points. This variation is reduced as we increase the size of the dataset to $\ell = 100$ (Figure 2b) at which point most of the obtained boundaries separate the two classes well. This is an important property of Bayesian approaches. As the number of points is low, the prior dominates the samples and thus, if the prior has a large variation, the posterior distribution will retain some of it and will still be affected by it.

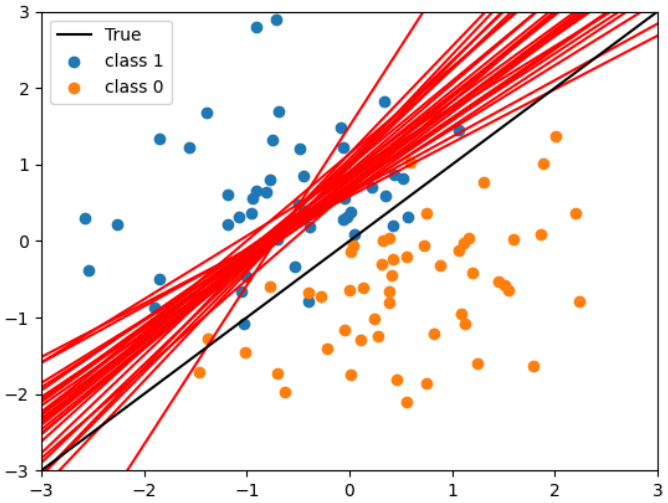
This is true independently of the chosen prior. If we change the choice of prior to a less fitting strong prior belief, the quality of the obtained models decreases. In Figure 2c we have changed the prior distribution of the offset θ_3 to a normal distribution with mean 5 and variance 0.1. In this case, even with $\ell = 100$ samples in the dataset, the decision boundaries still don't separate the two classes at a good location and the obtained decision boundaries are shifted upwards. Moreover, the posterior distribution shows low variation, similar to the case in Figure 2b. Thus, the variance of the posterior does not give information about the quality of the model. If the prior distribution is confidently wrong, the posterior is likely confidently wrong as well, unless a large amount of data is provided. Of course, if the provided prior is confident and right, then the posterior will be



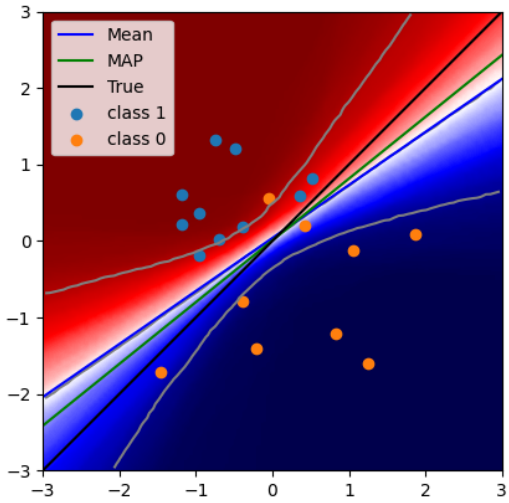
(a) $\ell = 20$



(b) $\ell = 100$



(c) Bad prior, $\ell = 100$



(d) Model selection

Figure 2: a+b) samples from the posterior distribution using datasets with different number of points ℓ. The training data is plotted as points, colors indicate the label of the dots. c) Same as b) but with a different prior distribution. d) Plot of the Mean (blue, eq (37)), MAP (green, eq. (36)) and a-posteriori distribution (background, grey lines, eq. (38))

much better and samples will approximate the true parameters with high precision, even if only little data is available.

So far we have only computed a distribution of parameters via the posterior distribution, but in practice we often have to decide on a concrete model to use. There are a few common options. The most well-known is the maximum-a-posteriori estimate (MAP), which picks the θ that maximizes $p(\theta|\mathcal{D})$:

$$\theta^{MAP} = \arg\max_{\theta} p(\theta|\mathcal{D}) = \arg\max_{\theta} p(\mathcal{D}|\theta)p(\theta) \ . \tag{36}$$

This approach does not require computing $p(\mathcal{D})$ as it is a constant over all queried values of θ. However, picking the point with the largest value of the posterior pdf is not equivalent to picking a point that is representative of the distribution. This point might be far away from the majority of sampled points. Another estimate is the mean

$$\hat{\theta} = E[\theta|\mathcal{D}] = \int p(\theta|\mathcal{D})\theta \ d\theta \ . \tag{37}$$

In practice this value can not be computed and is instead replaced by taking the sample mean over a set of sampled parameter vectors. Of course it is debatable, whether taking the mean of sampled parameter values is a sensible operation in our model. The solution that many Bayesians favour is defining a new distribution, called the *posterior predictive*

$$p(y|x, \mathcal{D}) = \int p(\theta|\mathcal{D})p(y|x, \theta)d\theta \ . \tag{38}$$

This approach samples a number of parameter values $\theta^{(i)}$ from $p(\theta|\mathcal{D})$ and then for each of them computes the distribution of labels $p(y|x, \theta^{(i)})$ for a given query input x. The prediction is the average of the label distributions. A decision is then taken, for example by picking the label y with the highest probability. For our logistic regression example, we visualize the three different solutions in Figure 2d. Both the MAP and mean solution differ from each other, but they are each a linear model. If we compare the posterior predictive (color background and grey lines in Figure 2d) and compare it to the original model Figure 1, it becomes clear that the distribution of label uncertainty is different. The shape formed by the grey lines in the posterior distribution is closer to an hourglass, than a straight line. This takes into account the spread of the samples in Figure 2a. The uncertainty of the direction of the decision boundary transforms to an uncertainty of its position in sample space. This uncertainty increases as we get further away from the midpoint of the data distribution, giving rise to the hourglass shape.

The biggest criticism of the Bayesian approach is the explicit choice of a prior that affects the final selection of models. This is because in a pure Bayesian approach, it is impossible to verify the quality of a prior. This sounds counterintuitive, because when comparing the results in Figures 2b&2c, it seems obvious which prior leads to a better data fit. To see the difficulty in evaluating priors, we have to look back at our data generative process (34). As long as $p(y|x, \theta)$ never assigns probability 0 to a label, we can obtain a dataset that includes an unlucky pick of labels. Thus, the posterior reflects a balance between picking a model we believe can be true and still somewhat aligns with a dataset that includes a few "unlucky" labels.

Given the difficulty of assessing a prior, how can we pick priors in the real world? After all, in many applications the fitted parameters are abstract and have no obvious meaning. There are several common techniques:

- Complexity arguments. A natural choice of prior in machine learning is one that penalizes model complexity. In the presence of little data, we prefer a simple model and as more data becomes available, we allow the chosen parameters to represent one that is more complex. In our logistic regression example, the complexity of the model can be measured in the norm $\|\theta\|$, as with increasing norm the model becomes more confident in its predictions and the probabilities are closer to 0 and 1. The normal distribution prior with zero mean used in our example does this.

- Weak priors: Sometimes no usable prior knowledge is available. In this case, a weak prior can be used that assigns prior probability to a wide range of parameters. This can be done by picking a normal distribution with a very large variance.

- Sometimes, we have a strong intuition about the general form of a prior, but don't know its parameters. In this case, we can parameterize the prior with its own set of parameters η and then add another prior distribution $p(\eta)$. The resulting prior on θ is then

$$p(\theta) = \int_{\Omega_\eta} p(\theta|\eta)p(\eta)d\eta \tag{39}$$

  This is called a hierarchical prior because as the number of data points increases, we begin to favour specific choices of η which then pick a prior for the parameters θ.

Bayesian model selection The hierarchical prior approach can be used to perform Bayesian model selection: the task of picking the model, or prior, that models the data best. In this, we pick η as an index for the chosen prior and then each $p(\theta|\eta)$ represents a choice of prior. For example, we can pick $p(\theta|\eta = 1)$ as a standard normal distribution to penalize model complexity, while $p(\theta|\eta = 2)$ encodes a different prior which we could have obtained by analysis of the learning problem in question (this analysis might be faulty and thus give rise to the bad prior case we discussed previously). Then, we pick a prior over η, for example a uniform distribution over choices. With this model, computing $p(\theta|\mathcal{D})$ automatically averages over the posterior distribution of η. this happens, since

$$p(\theta|\mathcal{D}) = \int_{\Omega_\eta} p(\theta|\eta, \mathcal{D})p(\eta|\mathcal{D})\ d\eta\ \ , \tag{40}$$

where $p(\theta|\eta, \mathcal{D})$ is the posterior distribution of the model parameters under the choice of prior indexed by η. This way, as more data is used, $p(\eta|\mathcal{D})$ becomes more and more concentrated over a single η: The degree of our belief over which prior is correct increases.

3.3 The Frequentist view

The Frequentist approach takes almost the opposite point of view when it comes to the basic assumptions. Here, we assume that the parameter values θ^* we search in a learning task are unknown, but fixed. Instead, we consider the dataset as a random variable and an estimator (or learning algorithm) is used to estimate the parameter values given the data. The point of view taken here is that for a given task, there exist optimal model parameters and the quality of a) our Dataset $\mathcal{D}$ and b) our estimator $\mathcal{A}$ are deciding factors in how close our estimates can get to the true parameters. Since we rarely have control over a), the focus of Frequentist approaches lies on selection of good estimators b).

Even if the estimator is deterministic, every time we obtain a new dataset for a given task the estimated parameter values will vary slightly and θ is distributed as

$$p(\theta|\theta^*, \mathcal{A}) = \int p(\mathcal{D}|\theta^*)p(\theta|\mathcal{A}, \mathcal{D})\ d\mathcal{D}\ \ . \tag{41}$$

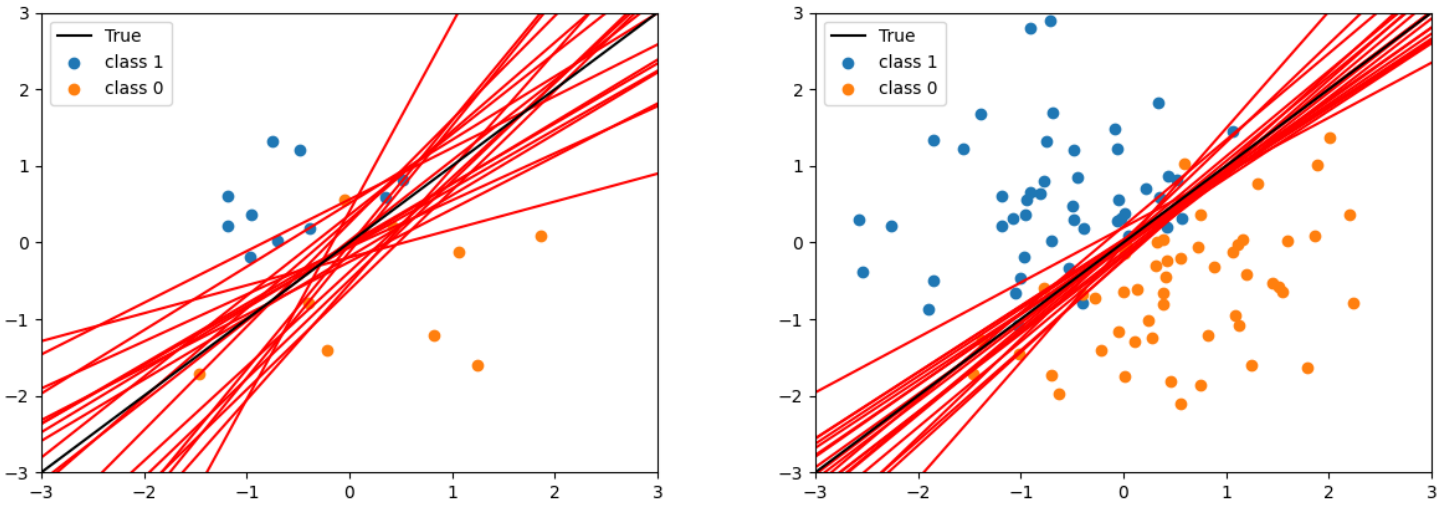
Here, $p(\mathcal{D}|\theta^*)$ uses a generative process similar to (34), except that we assume that θ^* is already drawn and therefore constant. The distribution $p(\theta|\mathcal{A}, \mathcal{D})$ is the distribution of parameter vectors returned by $\mathcal{A}$ given a sampled dataset. If the algorithm is deterministic, the distribution is a delta-peak, putting all probability mass on a single point.

Again, we can visualize this using the logistic-regression toy experiment. We sample a set of datasets $\mathcal{D}^{(i)}$ from our generative process for fixed θ^* and for each dataset, we obtain an estimate $\theta^{(i)}$ by applying the estimator to the dataset. As estimator, we use maximum likelihood:

$$\theta^{(i)} = \arg\max_\theta p(\mathcal{D}^{(i)}|\theta) \tag{42}$$

The results for datasets of sizes $\ell = 20$ and $\ell = 100$ are given in Figures 3a&3b. We can again see that with a small dataset, there is a lot of variation in the obtained models, which decreases as the number of data points increases.

This is a property of the chosen maximum likelihood estimator, but is not true for all estimators. For example consider the estimator that always returns the zero-vector, or the MAP estimate using the "bad" prior used in Figure 2c. Both estimators always show a small (or zero) variance, but



(a) $\ell = 20$ (b) $\ell = 100$

Figure 3: a+b) samples from the distribution $p(\theta|\theta^*, \mathcal{A})$ using datasets with different number of points ℓ. The training data is plotted as points, colors indicate the label of the dots.

model the data poorly. Thus, similar to the Bayesian case, the spread of the distribution $p(\theta|\theta^*, \mathcal{A})$ does not carry information about the quality of the estimator that produced it. If we want to select estimators based on their ability to model the data well, we need a different measure.

Risk measures are used to compare different estimators. They use a loss-function $L(\theta, \theta^*)$ that assign a loss based on the difference between true and estimated value. With this, we define the risk as the expected loss over $p(\theta|\theta^*, \mathcal{A})$, the distribution of estimated parameters using $\mathcal{A}$ as estimator integrated over all possible datasets

$$\mathcal{R}(\mathcal{A}, \theta^*) = \int p(\theta|\theta^*, \mathcal{A}) L(\theta, \theta^*) d\theta \tag{43}$$

A popular choice for the loss is the squared error $L(\theta, \theta^*) = \|\theta - \theta^*\|^2$. Intuitively, with this loss, the risk is the average squared distance between estimated and true parameter value. If the estimator is also unbiased, i.e., $E[\theta] = \theta^*$ the measure directly computes the variance of θ.

Example: We throw a coin N times and record the results as a dataset with outcomes $x_1, \ldots, x\ell$, $x_i \in \{0, 1\}$, where 1 stands for heads. The samples x_i follow a Bernoulli distribution $x_i \sim \text{Bernoulli}(\theta^*)$, where $\theta^* \in [0, 1]$ is the probability of the coin to land heads. As estimator $\mathcal{A}$ of θ^*, we pick the sample mean

$$\theta = \frac{1}{\ell} \sum_{i=1}^{\ell} x_i \ .$$

In this case, the data distribution of the generating process is known and the random variable $Z = \ell\theta$ follows a Binomial distribution. From this we can compute the risk using the squared loss (try deriving this yourself) and obtain:

$$R(\mathcal{A}, \theta^*) = \frac{1}{\ell} \theta^* (1 - \theta^*)$$

The risk as defined above depends on the value θ^*. Since this value is unknown, it seems difficult to compare estimators in practice as different estimators might perform better for different values of θ^*. What we would like to have is a single number. One option is to consider the *minimax risk*, that is the maximum risk over all θ^*,

$$\mathcal{R}_{\max}(\mathcal{A}) = \max_{\theta^*} \mathcal{R}(\mathcal{A}, \theta^*) \ .$$

Example: In the previous coin experiment, the minimax risk over all $\theta^* \in [0, 1]$ is obtained for $\theta^* = 1/2$ and

$$\mathcal{R}_{\max}(\mathcal{A}) = \max_{\theta^*} \frac{1}{\ell}\theta^*(1-\theta^*) = \frac{1}{4\ell} \ .$$

The minimax risk is dominated by the worst-case performance and is thus very pessimistic or even infinite. An average risk would be preferable, but the average risk requires to define a prior distribution on θ^*, $p(\theta^*)$. The resulting risk is called the Bayes-Risk:

$$\mathcal{R}_{\text{Bayes}}(\mathcal{A}) = \int p(\theta^*)\mathcal{R}(\mathcal{A}, \theta^*)d\theta^* \ .$$

The Bayes-Risk is accepted in the Frequentist community if there is a natural choice of $p(\theta^*)$.

Example: Linear Classifiers are estimators. A linear classifier assigns a label to a point based on which side of the decision boundary it lies:

$$h(x) = \text{sign}(\phi^T x)$$

Here, ϕ is the parameter vector selected by the learning procedure. Thus, the model $h(x)$ is itself an estimator of the unknown label y^* of the query point x. We can therefore compute a risk-measure for this estimator.
For classification, we can use the classification loss as risk measure that returns 1 if the wrong label is assigned

$$L(y, y^*) = \begin{cases} 0, \text{ if } y = y^* \\ 1, \text{ if } y \neq y^* \end{cases} \ .$$

The risk is then the expected miss-classification rate averaged over the samples of a specific class

$$\mathcal{R}(f, y^*) = \int p(x|y^*)L(h(x), y^*)\ dx \ .$$

The minimax risk is then just the miss-classification rate of the the class with highest risk. If we use the true label probability as prior, the Bayes risk is the proportion of miss-classified points

$$\mathcal{R}_{\text{Bayes}}(f) = \int p(x, y^*)L(h(x), y^*)\ dx\ dy^*$$

In applications, even if there is a natural choice of prior, the Bayes risk can typically not be computed, since we lack an analytical expression for it. Instead, if pairs $(\theta^*, \mathcal{D})$ sampled from the prior are available, we can compute the *empirical* Bayes risk by using known test sets, i.e., samples of problems with datasets $\mathcal{D}^{(i)}$, $i = 1, \ldots, N$ and parameters $(\theta^*)^{(i)}$. An estimator can generate estimates $\theta^{(i)}$ for each dataset and we can evaluate them using the expected loss

$$R_{\text{emp}}(\mathcal{A}) = \frac{1}{N}\sum_{i=1}^{N} L(\theta^{(i)}, (\theta^*)^{(i)}) \ . \tag{44}$$

Example: When evaluating the risk(=miss-classification rate) of a classifier, the data distribution is typically unknown. Instead, we can obtain a test set of labeled data $x^{(i)}, (y^*)^{(i)}$. The empirical Bayes risk is

$$R_{\text{emp}}(\mathcal{A}) \frac{1}{N} \sum_{i=1}^{N} L(h(x^{(i)}), (y^*)^{(i)}) \ .$$

Statistical Testing a Nutshell. If we have to choose an estimator, we can base this decision on the empirical risk, for example by picking the estimator with the smallest risk. However, even if the estimator is fixed, the empirical risk is a random variable due to its dependence on the test dataset, which in the Frequentist view is random. Thus, it is unclear whether for two estimators $\mathcal{A}_1$, $\mathcal{A}_2$ the observation that $R_{\text{emp}}(\mathcal{A}_1) < R_{\text{emp}}(\mathcal{A}_2)$ also implies $R_{\text{Bayes}}(\mathcal{A}_1) < R_{\text{Bayes}}(\mathcal{A}_2)$. Therefore, the correct approach is to test, whether the observed differences are so large compared to the variance of the empirical risk, that it is unlikely to be a random deviation. In case that the result is unlikely, the better of two models can be adapted, similarly to how we described it in the problem of the fair coin throw in the beginning of the chapter. This strategy is called statistical testing and one of the hallmarks of Frequentist methodology.

Importance to Bayesianism The importance of statistical testing also extends deep into Bayesian methodology, for two reasons. First, at all steps of statistical modeling we have to make assumptions about the model and distributions, for example in the generative model (34). Second, we are often not interested in the likelihood of the data under our model, but instead have fixed performance metrics, for example the miss-classification risk.

Therefore, model evaluation is key to ensure that our model also performs well in reality. Aside of model comparison, there are also more cross-overs between both approaches. For example, instead of a hierarchical prior, we can use the maximum likelihood estimate of the parameters of the prior. This gives rise to *empirical Bayes* methodology.

4 Generalized Linear Models

This chapter covers parts of the following chapters in the Bishop book: 3.1, 3.3, 4.3.

Independent of which statistical view is adapted, most machine learning approaches rely on the assumption of a generative model for the dataset $\mathcal{D}$. The most common is the the generative model in supervised learning for the dataset $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \dots, (x^{(\ell)}, y^{(\ell)})\}$, as we have stated in (34) and repeat here:

$$p(\mathcal{D}|\theta) = \prod_{i=1}^{\ell} p(x^{(i)}) p(y^{(i)}|x^{(i)}, \theta) \tag{45}$$

Notation: In supervised learning, we are interested in learning the relation $p(y^{(i)}|x, \theta)$ while we assume that $p(x)$ is given by an implicit distribution of samples: we want to predict the label for an input but do not want to model the distribution of inputs. Therefore, $p(x)$ is independent of our learned model and in all learning methods $p(x)$ will cancel out at some point. For example, when computing the posterior from (45) using a prior $p(\theta)$, we have

$$\begin{aligned} p(\theta|\mathcal{D}) &= \frac{p(\theta) \prod_{i=1}^{\ell} p(x^{(i)}) p(y^{(i)}|x^{(i)}, \theta)}{\int_{\Omega_\theta} p(\theta') \prod_{i=1}^{\ell} p(x^{(i)}) p(y^{(i)}|x^{(i)}, \theta') d\theta'} \\ &= \frac{p(\theta) \prod_{i=1}^{\ell} p(y^{(i)}|x^{(i)}, \theta)}{\int_{\Omega_\theta} p(\theta) \prod_{i=1}^{\ell} p(y^{(i)}|x^{(i)}, \theta') d\theta'} \end{aligned}$$

therefore, we will drop $p(x)$ in the following. This is equivalent to conditioning on $x^{(1)}, \dots, x^{(\ell)}$ as first step.

With logistic regression, we have already considered an example for binary labels $y^{(i)} \in \{0, 1\}$

$$p(y = 1|x, \theta) = \text{sigmoid}(f_\theta(x)) \ , \tag{46}$$

where $f_\theta(x) = \theta^T x$ is a linear model. Here, we used the sigmoid function

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}$$

to map between the model prediction and the probability of a label. But we could have used any function that maps from a real input to the interval $(0, 1)$. The question is therefore: What is the significance of the choice and how does it affect our model?

Linear Regression To answer this question, we look first at a different case, linear regression. Here, our labels are real valued, $y \in \mathbb{R}$ and our task is to find the parameters of a linear function f_θ such, that the predictions $f_\theta(x^{(i)})$ are as close as possible to $y^{(i)}$. We assume that the $y^{(i)}$ are noisy observations. This means that the labels assigned to a random point X are the response of the true underlying model $g(X)$, corrupted by random noise, which is typically assumed to be additive:

$$Y = g(X) + \epsilon \tag{47}$$

Here, ϵ is some random variable, modeling the measurement noise. Consequently, our labels are samples from this generative model

$$y^{(i)} = g(x^{(i)}) + \epsilon^{(i)} \ . \tag{48}$$

A frequent assumption is that $\epsilon \sim \mathcal{N}(0, \sigma_Y^2)$ and therefore

$$p(y|x) = \mathcal{N}(y; g(x), \sigma_Y^2) \ . \tag{49}$$

Of course we don't know $g(x)$, but assuming it is a linear function, our generative model takes the form

$$p(y|x, \theta) = \mathcal{N}(y; \theta^T x, \sigma_Y^2) \ . \tag{50}$$

Binary Classification For the binary classification case, we can try to adapt the model of linear regression and investigate possible alternatives to generalize (47) to binary labels. A possible generalization is as follows: We take $g(x)$ and view it as a function of class-membership: if $g(X) > 0$, then the label is 1, otherwise 0. Of course, since our predictions are affected by incomplete information and measurement noise, labels are not assigned based on $g(X)$ but $g(X) + \epsilon$, where ϵ now models our uncertainty about the class membership. Thus, labels are Bernoulli distributed random variables generated by

$$Y = \begin{cases} 1, & \text{if } g(X) + \epsilon > 0 \\ 0, & \text{otherwise} \end{cases} \quad . \tag{51}$$

The probability that $Y = 1$ can be computed easily, since $Y = 1$ entails $g(X) + \epsilon > 0$ and thus

$$P(y = 1|x) = P(\epsilon > -g(X)) = 1 - P(\epsilon \leq -g(X)) = 1 - \int_{-\infty}^{-g(X)} p(\epsilon)\, d\epsilon \ .$$

The function $P(\epsilon \leq t)$ for a univariate random variable is the cumulative distribution function (cdf).

Logistic Regression To derive logistic regression, we need to pick ϵ as distributed according to the standard logistic distribution with state space $\mathbb{R}$ and pdf

$$p(\epsilon) = \frac{\exp(-\epsilon)}{(1 + \exp(-\epsilon))^2} \ .$$

The cdf is given by

$$P(\epsilon \leq t) = \frac{1}{1 + \exp(-x))} = \text{sigmoid}(t)$$

And using properties of the sigmoid, we obtain:

$$P(y = 1|x) = 1 - P(\epsilon \leq -g(x)) = 1 - \text{sigmoid}(-g(x)) = \text{sigmoid}(g(x))$$

Replacing $g(X)$ by our model, we obtain the original result in equation (46).

Probit regression The choice of the logistic distribution for ϵ was arbitrary. After all, why shouldn't we use $\epsilon \sim \mathcal{N}(0, 1)$? If we do this, we can not write down the solution in such a nice closed form anymore, since

$$P(\epsilon < t) = \Phi(x) = \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}x^2)\, dx$$

does not have a closed form solution. The resulting model is called probit regression, based on the name for $\Phi(t)$, probit function. Even though, we can not compute the integral analytically, there are still efficient algorithms to compute $\Phi(t)$.

If that works, why do most people prefer logistic regression? Next to the fact that the sigmoid function can be written down easily using known primitives, there is an important statistical difference between the two models. The logistic distribution has much heavier tails than the normal distribution. That means that even at the same variance, the samples spread out over a larger area. Thus, the logistic regression is much more resilient to outliers, as it expects labels from the opposite class even further away from the boundary.

Generalized Linear Models We want to generalize the models we have discussed so far. We will call a model a generalized linear model, if it has the form

$$f_\theta(x) = \theta^T \phi(x) \ ,$$

where $\phi : \mathbb{R}^d \to \mathbb{R}^N$ is a feature mapping from the d dimensional input points x to a set of N features. Further, the linear model must be based on a likelihood function

$$p(y|x, \theta) = p(y|f_\theta(x)) \ .$$

The simplest example for ϕ is

$$\phi(x) = \begin{pmatrix} x \\ 1 \end{pmatrix} \quad ,$$

which simply adds a constant value to the input. this allows incorporating a bias term in the linear model. But any set of non-linear features are allowed, as long as they are independent of θ.

Finally, lets take a look at a Frequentist and Bayesian example.

4.1 Logistic Regression Revisited

We have already taken a look at Frequentist Maximum Likelihood estimation in the logistic regression example where we maximized

$$\arg\max_\theta p(\mathcal{D}|\theta) = \arg\max \prod_{i=1}^{\ell} p(y^{(i)}|x^{(i)},\theta) = \arg\min_\theta \underbrace{-\sum_{i=1}^{\ell} \log p(y^{(i)}|x^{(i)},\theta)}_{E(\theta)} \quad .$$

This searches the model that describes the data best, independent of any prior considerations about good values of θ. To find the maximum algorithmically, we perform gradient descent on the gradient of $E(\theta)$

$$\nabla E(\theta) = -\sum_{i=1}^{\ell} \nabla_\theta \log p(y^{(i)}|x^{(i)},\theta) \quad .$$

or use any other algorithm that finds minima of the error function.

4.2 Example: Bayesian Linear Regression

Returning to Linear regression, we can take a full Bayesian approach. We assume that our model has the form $f_\theta(x) = \theta^T x$ and its parameters have a prior distribution of $\theta \sim \mathcal{N}(0, \Sigma_\theta)$. This amounts to a choice of $\phi(x) = x$ in the previous chapter and we make this explicit choice just for the sake of simplicity of presentation: all derivations are the same when replacing x by $\phi(x)$. As before, we assume that the noise is normal distributed and $\epsilon \sim \mathcal{N}(0, \sigma_y^2)$. This leads to the likelihood function

$$p(y|x,\theta) = \mathcal{N}(y; \theta^T x, \sigma_y^2) \quad .$$

For simplicity, we will store the datapoints in a design matrix $\mathcal{X} \in \mathbb{R}^{\ell \times d}$ where the ith row stores the ith datapoint $x^{(i)}$. Similarly we have $\mathcal{Y} \in \{0,1\}^N$ as a vector storing the labels $y^{(i)}$. We can rewrite the likelihood in this notation and

$$p(\mathcal{D}|\theta) = \prod_{i=1}^{\ell} p(y^{(i)}|x^{(i)},\theta) = p(\mathcal{Y}|\mathcal{X},\theta) = \mathcal{N}(\mathcal{Y}; \mathcal{X}\theta, \sigma_y^2 I_N) \tag{52}$$

is a multivariate normal distribution.

To compute the posterior likelihood $p(\theta|\mathcal{D})$ (or $p(\theta|\mathcal{Y},\mathcal{X})$), we first need to compute $p(\mathcal{D},\theta)$. This means we need to compute the joint distribution of $p(\mathcal{Y}|\mathcal{X},\theta)$ and $p(\theta)$. Since both distributions are normal by our assumption and the conditional distribution (52) depends on the prior only in terms of a mean-shift, we can look up the property for the joint distribution from two dependent normal distributions, given in equation (31):

$$\left[\begin{array}{c} \theta \\ \hline \mathcal{Y} \end{array}\right] \sim \mathcal{N}\left(\left[\begin{array}{c} 0 \\ \hline 0 \end{array}\right], \left[\begin{array}{c|c} \Sigma_\theta & \Sigma_\theta \mathcal{X}^T \\ \hline \mathcal{X}\Sigma_\theta & \sigma_y^2 I_N + \mathcal{X}\Sigma_\theta \mathcal{X}^T \end{array}\right]\right) \quad . \tag{53}$$

Next, we want to condition $p(\theta, \mathcal{Y}|\mathcal{X})$ on $\mathcal{Y}$ using (29). To do this, we first reorder the random variables:

$$\left[\begin{array}{c} \mathcal{Y} \\ \hline \theta \end{array}\right] \sim \mathcal{N}\left(\left[\begin{array}{c} 0 \\ \hline 0 \end{array}\right], \left[\begin{array}{c|c} \sigma_y^2 I_N + \mathcal{X}\Sigma_\theta \mathcal{X}^T & \mathcal{X}\Sigma_\theta \\ \hline \Sigma_\theta \mathcal{X}^T & \Sigma_\theta \end{array}\right]\right) \quad . \tag{54}$$

And then conditioning leads to $p(\theta|\mathcal{D}) = p(\theta|\mathcal{X}, \mathcal{Y}) = \mathcal{N}(\theta; \mu_{\theta|\mathcal{D}}, \Sigma_{\theta|\mathcal{D}})$ with mean and variance

$$\mu_{\theta|\mathcal{D}} = \Sigma_\theta X^T (\sigma_y^2 I_N + \mathcal{X}\Sigma_\theta \mathcal{X}^T)^{-1} \mathcal{Y} \tag{55}$$

$$\Sigma_{\theta|\mathcal{D}} = \Sigma_\theta - \Sigma_\theta X^T (\sigma_y^2 I_N + \mathcal{X}\Sigma_\theta \mathcal{X}^T)^{-1} X \Sigma_\theta \ . \tag{56}$$

Remark: This result differs from the results presented in Bishop, section 3.3, equations 3.53 and 3.54. Numerically both results lead to the same values of mean and covariance, but the form we derived here is more useful for Gaussian Processes which we introduce later in the course.

For the normal distribution, the mode and mean are the same, so both the MAP and Mean model parameters are $\mu_{\theta|\mathcal{D}}$. For the posterior predictive distribution, we have to compute (38). We again need to apply (31) and obtain

$$p(y|x, \mathcal{D}) = \mathcal{N}(y; x^T \mu_{\theta|\mathcal{D}}, \sigma_y^2 + x^T \Sigma_{\theta|\mathcal{D}} x) \ . \tag{57}$$

With this model, we can not only give an estimate of the y but also a measure of uncertainty based on our prior and the inherent variance of the noise.

5 From Random Functions to Random Processes

In Bayesian linear regression, we chose a parameterized family of functions, and then used probability theory to find the posterior probability of parameters given the data. This is an indirect process, insofar as our goal was to find the function that fits the data best. Defining a parameterisation was only a means to an end. But can we get around this step and directly define a probability distribution on some space of functions?

We will first start with an example, showing that random functions are more difficult to discuss than the random variables we have looked at before. Then, we introduce the notion of random processes. A word of warning: random processes are a difficult area of math and require substantial amounts of measure theory to define correctly. We will not go this route and instead try to give an intuition about, what random processes are and how to work with them.

To begin, consider a set of functions $f : \mathbb{N} \to \mathbb{R}$. As the argument is a natural number, these functions are equivalent to sequences $f(i) = f_i$, $i = 1, 2, \dots$. We can define a random function belonging to this family by picking a distribution for each f_i, for example

$$f_i \sim \mathcal{N}(0, 0.01) \ .$$

This gives us an immediate way to compute the probability of the function f, via the probability of the sequence $p(f_1, f_2, \dots)$. If we only consider sequences of length ℓ, we know how to compute the probability, as

$$p(f_1, f_2, \dots, f_\ell) = \prod_{i=1}^{\ell} \mathcal{N}(f_i; 0, 0.01) \ .$$

But what happens as $\ell \to \infty$? Let us look at two cases, $f_i = 0$, $i = 1, \dots, \ell$ and $f_i = 1$, $i = 1, \dots, \ell$. For $f_i = 1$, we obtain

$$p(1, 1, \dots) = \lim_{\ell \to \infty} \prod_{i=1}^{\ell} \underbrace{\mathcal{N}(1; 0, 0.01)}_{<1} = 0 \ ,$$

while for $f_i = 0$, we get

$$p(0, 0, \dots) = \lim_{\ell \to \infty} \prod_{i=1}^{\ell} \mathcal{N}(0; 0, 0.01) = \lim_{\ell \to \infty} \left(\underbrace{\frac{10}{\sqrt{2\pi}}}_{>1} \right)^{\ell} = \infty \ .$$

The first case is still rather benign, as probability 0 just means that we will never draw this sample. In the second case, we obtain a pdf of infinity, which will hold true when slightly varying the values of f_i around 0. However, if we compute the cdf

$$P(F_1 < u_i, F_2 < u_2, \dots) = \lim_{\ell \to \infty} \prod_{i=1}^{\ell} \underbrace{\int_{-\infty}^{u_i} \mathcal{N}(f_i; 0, 0.01) \, df_i}_{<1} = 0 \ .$$

which means that even though we integrate over volumes with infinite pdf, the resulting integrals are 0 everywhere. Thus, we can conclude that our notion of integral and probability does not make sense in these spaces. We need to find a different way to work with these objects.

5.1 Random Fields & Random Processes

As a way around the problems of defining the probability of a random function, the notion of a random field was developed.

Let Ω be an event space and let $\mathcal{X}$ be an index set (e.g. $\mathbb{N}$ or $\mathbb{R}^d$). A random field is a collection of random variables

$$F_x \in \Omega, \ \forall x \in \mathcal{X} \ .$$

Intuitively, a random field is a function that assigns a random variable to each point $x \in \mathcal{X}$. An alternative intuition is to see a random field as a (potentially infinite) set of dependent random variables, and the index set allows us to identify and select them. In the literature, if the index set is $\mathbb{R}^d$, a field is also called a random process.

The notion of a random field is very broad. For example, if we take a n-dimensional multivariate normal distributed random variable $g \in \mathbb{R}^d \sim \mathcal{N}(\mu, \Sigma)$, then, if we pick an index-set $\mathcal{X} = \{1, 2, \ldots, d\}$, the elements of the vectors g_i form a random field.

As a real world example, we can measure the height of the ocean at a set of chosen points on earth. In this example, the index set could be the longitude and latitude of the measurement position, and the measured ocean height at a given point is a random variable. when measuring at two points close to each other, the result will be highly correlated, as the same wave tends to pass through both points, while for far away points, the correlation will become less and less strong (but once you measure on opposite sides of the earth, you will be able to measure a new correlation due to the gravitational pull of the moon!). This application also introduces a new type of question that we could not ask previously: given a set of measurements, what is the distribution of ocean height at a point we have not measured yet? And where can we add a measurement position such, that it minimizes the uncertainty over all points of the ocean?

As a final example, as we can pick $\mathbb{N}$ as index set, the sequence example we discussed at the beginning of the chapter is also a random field. Thus, random fields can not solve our problems of assigning probabilities to the full collection of random variables. We must accept that even if we have an infinite set of random variables, we can only look at a finite subset $S_\ell = \{x_1, \ldots, x_\ell\} \subseteq \mathcal{X}$, $\ell \in \mathbb{N}$ at a time. This means that at any point, we can only reason about the random process random variables F_x via its marginals

$$p(F_{x_1}, F_{x_2}, \ldots, F_{x_\ell}) \ .$$

Notation: Given a set $S = \{x_1, \ldots, x_\ell\}$, $\ell \in \mathcal{N}$, we use the following shorthand notation for the marginals:

$$p(f_1, \ldots, f_\ell | S) = p(F_{x_1} = f_{x_1}, F_{x_2} = f_{x_2}, \ldots, F_{x_\ell} = f_{x_\ell}) \ .$$

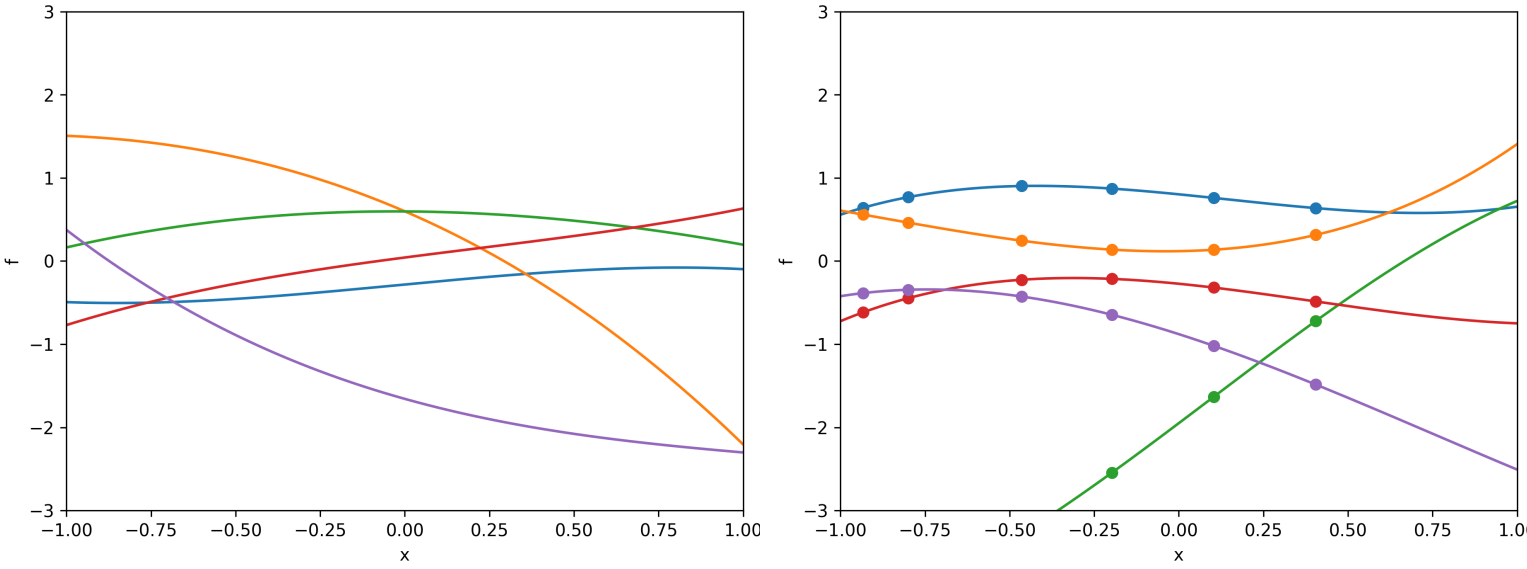
We sometimes write $p(f_S|S)$, or $p(f|S)$ as shorthand if it is clear that we mean an f of size ℓ. Moreover, we will drop the distinction between random variables F_x and the observations f_x.

By only looking at marginals, we no longer compute probabilities of functions, but probabilities of point-observations: the probability $p(f|S)$ measures the probability of drawing a function that has value f on the observed subset S and for any S and f, there could be infinitely many functions that could have lead to this value.

The question arises, whether we lose anything, if we discuss a random process only in terms of its marginal distributions with finite number of variables. This is answered by Kolmogorovs Consistency Theorem. We will only paraphrase it in the following as its full statement requires proper handling of measures and topology. Kolmogorovs Consistency Theorem states that we can indeed represent a random process fully via a family of marginal distributions $p(f_1, \ldots, f_\ell|S)$, $S_\ell \subseteq X$, $\ell \in \mathcal{N}$. Likewise, if two random fields have the same marginal distributions, they are the same. The result is called a consistency theorem as it implies a consistency property between the marginal distribution of different subsets and it can be used to test, whether a family of marginal distributions $p(f|S)$ indeed represents a random field.

Let $T \subset S_\ell$. We can partition the vector of selected random variables $f = (f_1, \ldots, f_\ell)$ as $f = (f_T, f_C)$, where f_T are the variables indexed by T and f_C are the variables indexed by S but not T. Then, Kolmogorovs consistency theorem requires

$$p(f_T|T) = \underbrace{\int p(f_T, f_C|S) \, df_C}_{p(f_T|S)} \ . \tag{58}$$



(a) Random Function view

(b) Random Field View

Figure 4: Bayesian linear regression: Sampled functions from the function prior over third degree polynomials. Left: Just the polynomials. Right: When viewing this as a random process, we only observe the values of the functions at a few selected points.

Thus, marginalizing variables in $p(f|S)$ must lead to the same result as picking the function $p(f_T|T)$ from the family of marginals. Intuitively, this makes sense: including an unobserved random variable in the model must not affect the distribution of the observed variables, as otherwise we are forced to include all unobserved random variables which makes computing probabilities impossible.

With this property, we can check, whether the given family of marginals forms a random field simply by testing, whether this property holds for all S_ℓ and T. Of course, this does not give us a way to define random fields, only to check that what we came up with is correct.

5.2 Example: Bayesian Linear Regression as Random Process

In this section, our aim is to view Bayesian Linear Regression through the lens of random processes. We will not focus on the prediction, but purely on a different interpretation of prior. In Bayesian linear regression, we first choose a linear regression function

$$f_\theta(x) = \theta^T \phi(x)$$

and then add a prior distribution on θ, which we choose here as some multivariate normal distribution, $\theta \sim \mathcal{N}(0, \Sigma_\theta)$. We will now change our point of view by considering that each observed value of θ leads to a function f_θ and we can thus reinterpret our prior not as a prior over the parameters of the function, but the compound of prior and basis function as a prior over a chosen set of functions.

Example: In a 1-dimensional regression problem, we can pick ϕ as a set of polynomial basis functions, $\phi(x) = (1, x, x^2, x^3, \ldots, x^N)$. We can now assign a prior probability distribution to each θ_i, for example $\theta_1 \sim \mathcal{N}(0, 1)$ and $\theta_i \sim \mathcal{N}\left(0, \frac{1}{(i-1)^2}\right)$, for $i > 1$.
Now, sampling a θ from our prior gives rise to a polynomial function $f_\theta(x) = \theta_1 + \theta_2 x + \theta_3 x^3, \ldots$, which we can plot instead of plotting θ. An example for this for $N = 3$ is given in Figure 4a.

In reality we can not plot a function at each point, but we can only do so at a certain set of chosen locations, while for all values in-between we can only make educated guesses. From the point of view of a random process this becomes even more strict: we can never observe a function, but only observe its function values at a given set of points. To visualize this better, we give another example (Figure 4b) where we chose a set of points $S = \{x_1, \dots x_\ell\}$ and marked the observed value of $f_i = f_\theta(x_i)$.

Seeing S as a subset of an index set of evaluation locations, for example $x \in [-1, 1]$ turns the observations into a random field: depending on the evaluated positions, we get a different set of observations f_x from the random variables F_x with values given by the function values of the underlying drawn function. The marginals of this random process can be directly computed from our given prior process. As first step, we write f for the vector of observations at points in S for some f_θ:

$$f = \begin{bmatrix} f_1 \\ f_2 \\ \dots \\ f_\ell \end{bmatrix} = \begin{bmatrix} \phi(x_1)^T\theta \\ \phi(x_2)^T\theta \\ \dots \\ \phi(x_\ell)^T\theta \end{bmatrix} = \underbrace{\begin{bmatrix} \phi(x_1)^T \\ \phi(x_2)^T \\ \dots \\ \phi(x_\ell)^T \end{bmatrix}}_{\Phi(S)\in\mathbb{R}^{\ell\times N}} \theta = \Phi(S)\theta \tag{59}$$

Each row of $\Phi(S)$ contains the values of the different feature maps at the evaluated points and thus the value of f can be computed as a matrix-vector product. As θ is a multivariate normal distributed random variable, we can use the property that multivariate normal distributions are closed under linear transformations, and the distribution of f can be directly written as

$$f \sim \mathcal{N}(0, \Phi(S)\Sigma_\theta\Phi(S)^T) \ .$$

Consistency can be easily shown based on the construction of f $(= f_S)$ in (59). In this equation, we obtain one row for each element in S and each row is independent of all other elements in S. Following the construction, we can construct f_T the same way. A quick check shows that the result is the same as removing the rows in (59) that are in S but not in T. But exactly this is marginalization! One can check this analytically by using that marginalisation in normal distributions amounts to removing the rows and columns of the covariance matrix (and rows in the mean) that belong to the variables that are integrated over.

5.3 Example: Ornstein-Uhlenbeck

Let us introduce a random process that is commonly observed in many different areas of science: The Ornstein-Uhlenbeck process. It is of important for Machine-Learning as it forms the basis of diffusion processes, but it also appears in various models of correlated noise, for example in quantum physics. We will first give its construction, then compute the marginals and finally show that it fulfills Kolmogorovs consistency theorem.

As index set, we take an ordered set of points $S = \{t_1, \dots, t_\ell\} \subseteq [0, \infty)$, $-\infty = t_0 < t_1 < t_2 < \dots < t_\ell$. The ordering is no limitation as we can pick any subset of points and reorder their indices. The Ornstein-Uhlenbeck process has two parameters, the marginal variance σ^2 and the time correlation parameter θ.

We then sample a set of random variables $f_1, \dots, f_\ell$ using the following relation:

$$f_1 = \epsilon_1 \tag{60}$$
$$f_i = a_i f_{i-1} + \epsilon_i \tag{61}$$
$$\epsilon_i \sim \mathcal{N}(0, \sqrt{1 - a_i^2}\sigma^2) \tag{62}$$
$$a_i = e^{-\theta(t_i - t_{i-1})} \in [0, 1] \tag{63}$$

Note, that in the equation above, $a_1 = 0$ due to the definition of $t_0 = -\infty$. Thus, the samples are created via a Markov chain with conditional distributions

$$p(f_i|f_{i-1}, S) = \mathcal{N}(f_i; a_i f_{i-1}, \sigma^2(1 - a_i^2))$$

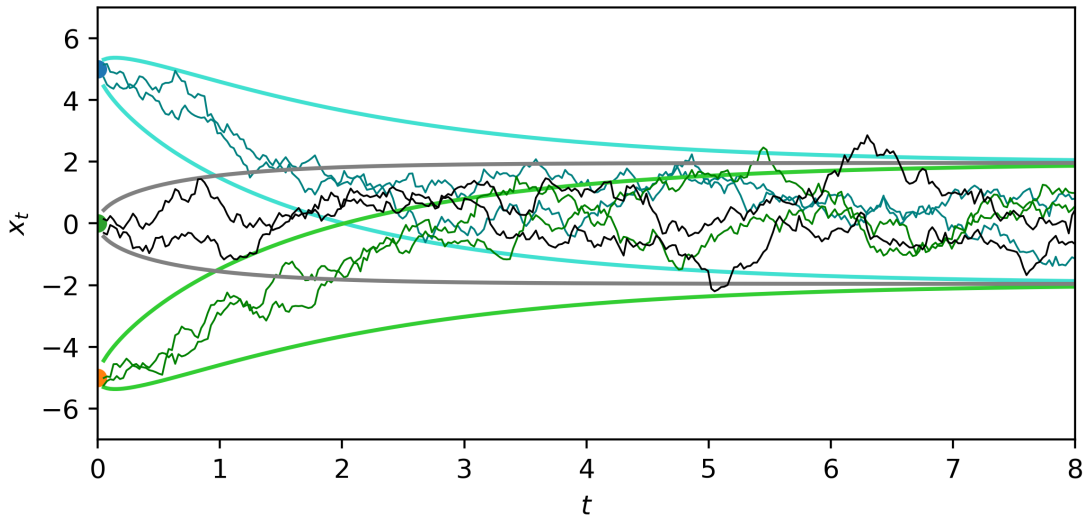


Figure 5: Visualisation of the Ornstein-Uhlenbeck process for $t \in [0, 8]$. Wiggly lines are samples from the process by following (61) starting from $\{-5, 0, 5\}$. The other liens depict 95% confidence intervals of the distrivbutrion $p(f_t|f_0)$. TODO: y-axis in plot should be f_t.

and marginal

$$p(f_0, \ldots, f_\ell|S) = p(f_0) \prod_{i=0}^{\ell-1} p(f_{i+1}|f_i, S) \ .$$

the resulting samples drawn for $\ell = 100$ are shown in Figure 5 for $\theta = 1$ and $\sigma = 1$. For visualisation purposes we picked the initial values from $\{-5, 0, 5\}$. The points that start further away from the mean have an initial trend towards smaller values and then begin to wiggle around the mean in complex Zig-Zag lines. We also depict the 95% interval of the conditional $p(x_t|x_0)$, which we will derive later in this example.

To check whether this definition of marginal forms a random process, we have to find a better formulation of the marginal first and then show that it fulfills the consistency theorem.

The first thing we note is that we can write the random variable f_i solely as a sum of ϵ_k, $k = 1 \ldots, i$:

$$\begin{aligned} f_i &= a_i f_{i-1} + \epsilon_i \\ &= a_i a_{i-1} f_{i-2} + a_i \epsilon_{i-1} + \epsilon_i \\ &= \ldots \\ &= \sum_{k=1}^{i} \left[\prod_{j=k+1}^{i} a_j \right] \epsilon_k, \end{aligned}$$

where we set $a_1 = 0$ for simplicity. By writing out the set of all $f_1 \ldots, f_\ell$ in matrix form, we can see that they are related by a linear transformation of w_i:

$$\underbrace{\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \ldots \\ f_\ell \end{bmatrix}}_{f} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & \ldots & 0 \\ a_2 & 1 & 0 & 0 & \ldots & 0 \\ a_2 a_3 & a_3 & 1 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ldots & 0 \\ \prod_{j=2}^{\ell} a_j & \prod_{j=3}^{\ell} a_j & \prod_{j=4}^{\ell} a_j & & \prod_{j=5}^{\ell} a_j \ldots & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \ldots \\ \epsilon_\ell \end{bmatrix}}_{\epsilon} ,$$

where ϵ is a multivariate normal random variable

$$\epsilon \sim \mathcal{N}\left(0, \underbrace{\begin{bmatrix} \sigma^2(1-a_1^2) & 0 & 0 & \dots & 0 & \\ 0 & \sigma^2(1-a_2^2) & 0 & 0 & \dots & 0 \\ 0 & 0 & \sigma^2(1-a_3^2) & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots\dots & 0 \\ 0 & 0 & 0 & 0 & \dots & \sigma^2(1-a_\ell^2) \end{bmatrix}}_{D}\right) .$$

This means that f is a linear transformation of a multivariate normal distributed random variable ϵ and thus is itself is multivariate random variable (property: closed under linear transformations) that follows:

$$f \sim \mathcal{N}(0, \underbrace{ADA^T}_{K(S)}) \ . \tag{64}$$

Next, we will compute the values of single entries of $K(S)$. Using that D is a diagonal matrix, we can write a single element is

$$K(S)_{ij} = \sum_{k=1}^{\ell} A_{ik} D_{kk} A_{jk}$$

Since A is a lower triangular matrix, in row A_i only the first i elements are non-zero and the same holds for the first j element of A_j. Thus, assuming $j \leq i$ (and computing the rest via the symmetry of $K(S)$) we can limit the sum to

$$K(S)_{ij} = \sum_{k=1}^{j} A_{ik} D_{kk} A_{jk}$$

as for all other cases one of the factors will be zero.

We begin by computing the diagonal elements. It holds

$$\begin{aligned}
K(S)_{ii} &= \sum_{k=1}^{i} A_{ik}^2 D_{kk} = \sum_{k=1}^{i} \prod_{r=k+1}^{i} a_r^2 \sigma^2 (1-a_k^2) \\
&= \sigma^2 \left((1-a_i^2) + a_i^2(1-a_{i-1}^2) + a_i^2 a_{i-1}^2 (1-a_{i-2}^2) + \dots + a_i^2 a_{i-1}^2 \cdots a_2 (1-a_1^2) \right) \\
&= \sigma^2 \left((1-a_i^2) + a_i^2 \underbrace{\left((1-a_{i-1}^2) + a_i^2 \underbrace{\left(\dots (1-a_2^2) + a_2^2 \underbrace{(1-a_1^2)}_{1} \right)}_{1} \right)}_{1} \right) \\
&= \sigma^2 \cdot 1 \ .
\end{aligned}$$

In the derivation, we began by inserting the elements of A_{ik} and pulling the square inside the product. In the next step we wrote the sum of products term by term and factored out σ^2. We can see that once an element a_k appears in one of the terms, it also appears in all terms on its right hand side. We use this to factorize the terms. This leads to a lot of constructs of the form $(1-a_k)+a_k b_k$, where b_k is a placeholder for the factor behind a_k after factorization. On the inner most term, we have $(1-a_2)+a_2 b_2$, where $b_2 = 1-a_1$. Since by definition $a_1 = 1$, and therefore we have $b_2 = 1$. From this follows that all $b_k = 1$.

For the off-diagonal elements we assume again for simplicity that $j < i$. We obtain

$$\begin{aligned}
K(S)_{ij} &= \sum_{k=1}^{j} A_{ik} A_{jk} D_{kk} = \sum_{k=1}^{j} \prod_{r=k+1}^{j} a_r \prod_{r'=k+1}^{i} a_{r'} \sigma^2 (1 - a_k^2) \\
&= \sum_{k=1}^{j} \prod_{r=k+1}^{j} a_r \prod_{r'=j+1}^{i} a_{r'} \sigma^2 (1 - a_k^2) \\
&= \prod_{r'=j+1}^{i} a_{r'} \underbrace{\sum_{k=1}^{j} \prod_{r=k+1}^{j} \sigma^2 a_r^2 \sigma^2 (1 - a_k^2)}_{K(S)_{jj}} \\
&= \sigma^2 \prod_{r'=j+1}^{i} a_{r'} \ .
\end{aligned}$$

The key observation is that $A_{ik} = A_{jk} a_{j+1} \cdots a_i$ and thus we can factor out the common factors.

We can simplify the elements of $K(S)$ one final time. Remember that $a_i = e^{-\theta |t_i - t_{i-1}|}$ and $t_i > t_{i-1}$. Then for $j < i$:

$$\begin{aligned}
K(S)_{ij} &= \sigma^2 \prod_{r'=j+1}^{i} a_{r'} \\
&= \sigma^2 \prod_{r'=j+1}^{i} e^{-\theta (t_r - t_{r-1})|} \\
&= \sigma^2 e^{-\theta \sum_{r'=j+1}^{i} t_r - t_{r-1}} \\
&= \sigma^2 e^{-\theta (t_i - t_j)}
\end{aligned}$$

For $j > i$ we obtain the same result with a sign change in the exponential function (to compensate for $t_j > t_i$) and by noticing that when we plug in $j = i$ in the equation above, we obtain $K(S)_{ii}$ we arrive at:

$$K(S)_{ij} = \sigma^2 e^{-\theta |t_i - t_j|} \tag{65}$$

With this, we can write the matrix $K(S)$ simply as

$$K(S) = \sigma^2 \begin{bmatrix}
1 & e^{-\theta(t_2 - t_1)} & e^{-\theta(t_3 - t_1)} & \dots & e^{-\theta(t_\ell - t_1)} \\
e^{-\theta(t_2 - t_1)} & 1 & e^{-\theta(t_3 - t_2)} & \dots & e^{-\theta(t_\ell - t_2)} \\
e^{-\theta(t_3 - t_1)} & e^{-\theta(t_3 - t_2)} & 1 & \dots & e^{-\theta(t_\ell - t_3)} \\
\vdots & \vdots & \vdots & \dots & \vdots \\
e^{-\theta(t_\ell - t_1)} & e^{-\theta(t_\ell - t_2)} & e^{-\theta(t_\ell - t_3)} & \dots & 1
\end{bmatrix} . \tag{66}$$

As a special case, we can now compute the marginal distributions for single X_t. Independently of t it holds $X_t \sim \mathcal{N}(0, \sigma^2)$. Random Processes with the property that all marginals are identical are called *stationary*, since they model no global trends in the function. Moreover, for the conditional distribution, we have

$$f_t | f_{t'} \sim \mathcal{N}\left(e^{-\theta |t - t'|}, \sigma^2 \left(1 - e^{-2\theta |t - t'|} \right) \right) \tag{67}$$

Thus, as $t - t'$ becomes large, the mean of $f_t | f_{t'}$ approaches 0 and the variance approaches σ^2. We have seen a depiction of this already in Figure 5. As a result, as $t - t' \to \infty$, the random variables f_t and $f_{t'}$ become mutually independent of each other. In practice, this process can be rather quick. With $\theta |t - t'| > 5$, the exponential term in front of the mean is smaller than 0.01. This reveals the role of θ as correlation parameter: it directly governs how quickly two samples become independent of each other. In some sense, it is the same as defining the unit of time and is therefore often kept as $\theta = 1$.

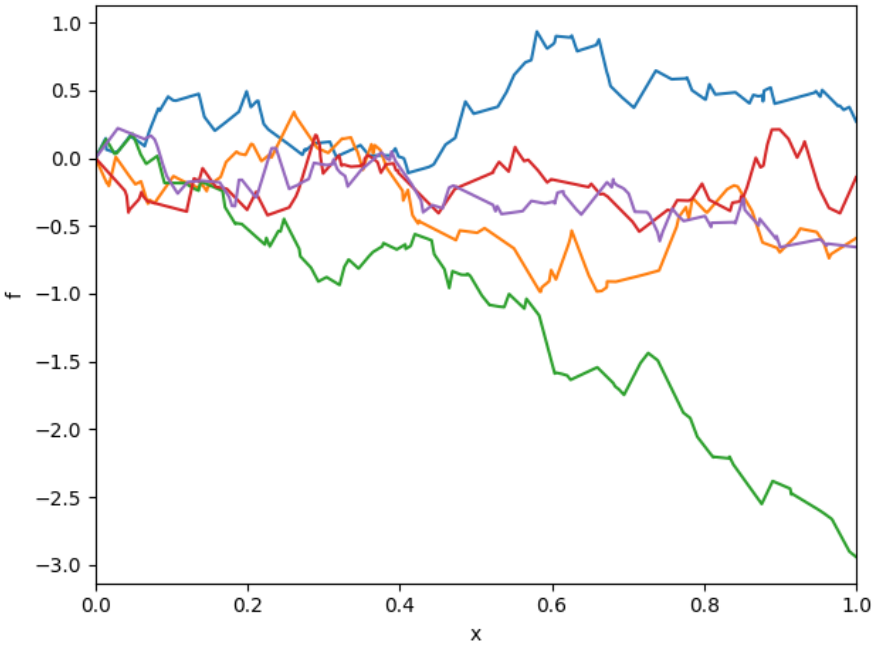


Figure 6: Samples from the Wiener process with S being a set of 100 equally spaced points

Our goal was to show that Kolmogorov consistency holds. We now have all the tools needed, to check whether our marginals $p(S|f)$ are consistent as required by (58). For this, we must take a subset of T of any S and show that marginalizing $p(f_S|S)$ leads to the same distribution as $p(f_T|T)$. i.e., $p(f_T|S) = p(f_T|T)$. Since we have shown that the marginals are multivariate normal distributions with mean 0, the only thing we have to show is that the covariance matrices match.

First, take T as a subset of S with one element less, for example by removing x_2. Then, marginalizing means to remove the second row and column from $K(S)$ in (70). Since the elements $K(S)_ij$ only depend on i and j, once we removed the second row and column, all terms depending on x_2 are removed as well. The remaining terms are identical to the ones in $K(T)$. Thus the marginals are consistent. Now, we can iteratively apply this process for any T by crossing out rows and columns one-by-one. This shows that the marginals are consistent, and the Ornstein-Uhlenbeck Process is indeed a random process.

For the moment, it is time to leave the Ornstein-Uhlenbeck process behind. We will return to it later, when we talk about Diffusion in Section 7.

5.4 Additional Example: Wiener Process

Note: This is an older Example used in previous iterations of the course. This example performs the exact same derivations as the Ornstein-Uhlenbeck example before, and thus can serve as an additional example for the reader.

Let us introduce the Wiener Process as a non-trivial example for a random process. We will first give its construction, then compute the marginals and finally show that it fulfills Kolmogorovs consistency theorem.

As index set, we take an ordered set of points $S = \{x_1, \dots, x_\ell\} \subseteq [0, 1]$, $0 = x_0 \leq x_1 < x_2 < \cdots < x_\ell$. The ordering is no limitation as we can pick any subset of points and assign their indices via their order. We then sample a set of random variables $f_1, \dots, f_\ell$ using

the following relation:

$$\begin{aligned} f_0 &= 0 \\ f_{i+1} &= f_i + W_{i+1},\ W_{i+1} \sim \mathcal{N}(0, x_{i+1} - x_i) \end{aligned}$$

Thus, the samples are created via a Markov chain with conditional distributions

$$p(f_{i+1}|f_i, S) = \mathcal{N}(f_{i+1}; f_i, x_{i+1} - x_i)$$

and marginal

$$p(f_0, \ldots, f_\ell|S) = \prod_{i=0}^{\ell-1} p(f_{i+1}|f_i, S) \ .$$

the resulting samples drawn for $\ell = 100$ are shown in Figure 6. All samples start at 0 by construction and then fan out in different directions with a larger spread the larger x is. Further, all samples are very rough and form a zick-zack path.

To check whether this definition of marginal forms a random process, we have to find a better formulation of the marginal first and then show that it fulfills the consistency theorem.

The first thing we note is that we can write the random variable f_i solely as a sum of W_i:

$$f_{i+1} = f_i + W_{i+1} = f_{i-1} + W_{i+1} + W_i = \cdots = \underbrace{f_0}_{0} + \sum_{j=1}^{i+1} W_j$$

By writing out the set of all $f_1 \ldots, f_\ell$ in matrix form, we can see that they are related by a linear transformation of W_i:

$$\underbrace{\begin{bmatrix} f_1 \\ f_2 \\ \cdots \\ f_\ell \end{bmatrix}}_{f} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ 1 & 1 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ldots & 0 \\ 1 & 1 & 1 & \ldots & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} W_1 \\ W_2 \\ \cdots \\ W_\ell \end{bmatrix}}_{W} ,$$

where W is a multivariate normal random variable

$$W \sim \mathcal{N}\left(0, \underbrace{\begin{bmatrix} x_1 - x_0 & 0 & 0 & \ldots & 0 \\ 0 & x_2 - x_1 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ldots & 0 \\ 0 & 0 & 0 & \ldots & x_\ell - x_{\ell-1} \end{bmatrix}}_{D}\right) \ .$$

This means that f is a linear transformation of a multivariate normal distributed random variable and thus it itself is multivariate random variable (property: closed under linear transformations) that follows:

$$f \sim \mathcal{N}(0, \underbrace{ADA^T}_{K(S)}) \ . \tag{68}$$

Next, we will compute the values of single entries of $K(S)$. Using that W is a diagonal matrix, we can write a single element is

$$K(S)_{ij} = \sum_{k=1}^{\ell} A_{ik} D_{kk} A_{jk}$$

Since A is a lower triangular matrix, in row A_i only the first i elements are non-zero and the same holds for the first j element of A_j. Thus we can limit the sum to

$$K(S)_{ij} = \sum_{k=1}^{\min\{i,j\}} A_{ik} D_{kk} A_{jk}$$

as for all other cases one of the factors will be zero. Next, we use that in the definition of A all non-zero elements are 1 and thus we can simplify further to

$$K(S)_{ij} = \sum_{k=1}^{\min\{i,j\}} D_{kk} = \sum_{k=1}^{\min\{i,j\}} (x_k - x_{k-1}) = x_{\min\{i,j\}} - x_0$$

Finally, we can simplify this further as we have defined $x_0 = 0$ and ordered the elements in S such $x_1 < x_2 < \dots$. Thus if $i < j$, $x_i < x_j$. With this we arrive at

$$K(S)_{ij} = \min\{x_i, x_j\} \tag{69}$$

With this and using the known ordering, we can write the matrix $K(S)$ simply as

$$K(S) = ADA^T = \begin{bmatrix} x_1 & x_1 & x_1 & \dots & x_1 \\ x_1 & x_2 & x_2 & \dots & x_2 \\ x_1 & x_2 & x_3 & \dots & x_3 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ x_1 & x_2 & x_3 & \dots & x_\ell \end{bmatrix} . \tag{70}$$

The same argument as for the previous example can now be used to show Kolmogorov consistency.

6 Gaussian Processes

In the last chapter we have seen several examples of random processes in which the marginals were normal distributions, $\mathcal{N}(0, K(S))$, where S is a finite subset of some index set $\mathcal{X}$. In this section we will introduce a generalization of this, the family of Gaussian Processes. But before we can introduce them, we have to introduce the concept of a Hilbert-space kernel (or in short, kernel)

Definition (Kernel): Let $\mathcal{X}$ be some set. Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. If for all $S = \{x_1, \dots, x_\ell\} \subset \mathcal{X}$ and any $\ell \in \mathbb{N}$ it holds

$$K(S) = \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_\ell) \\ \vdots & \ddots & \vdots \\ k(x_\ell, x_1) & \dots & k(x_\ell, x_\ell) \end{bmatrix} \text{ is symmetric positive semi-definite}$$

We call k a kernel.

As a reminder, a matrix is positive semi-definite, if all its eigenvalues are bigger or equal to zero. We will refer to a matrix $K(S)$ defined as above as kernel matrix. The definition of a kernel is at this point purely abstract. The only thing we know is that the set of positive semi-definite matrices is exactly the set of covariance matrices and the goal in the remainder of the chapter is to fill this definition with meaning and to provide an intuition into what a kernel is doing.

Note: Remember that by the definition of multivariate normal distributions, equation (25) and the following page, the covariance matrix of a multivariate normal distribution does not need to be invertible. This is only needed if we also want to compute a pdf.
We have already seen an example where the matrix was not positive definite. In Section 5.2 we analyzed a set of polynomial random functions $\phi(x) = (1, x, x^2, x^3)$, forming a third degree polynomial. A third degree polynomial has 4 parameters, $\theta_1, \dots, \theta_4$. Given 4 pairs of (x_i, f_i) it is possible to uniquely define a polynomial $f(x)$. What happens if we add a fifth pair (x_5, f_5)? Either, it is on the graph of f, that is $f(x_5) = f_5$ or there does not exist a third degree polynomial passing through all 5 points. Thus the covariance matrix computed in Section 5.2 only has rank at most 4. Once the number of observations exceeds 4, the additional points added are forced to lie on the line of the third degree polynomial defined by the first four points.

For now, we will just use the kernel to define Gaussian Processes:

Definition (Gaussian Process): Let $\mathcal{X}$ be an index set

A random field $F_x \in \mathbb{R}$ whose marginals $p(f|S)$ are Multivariate Normal distributions, is called a Gaussian Process.

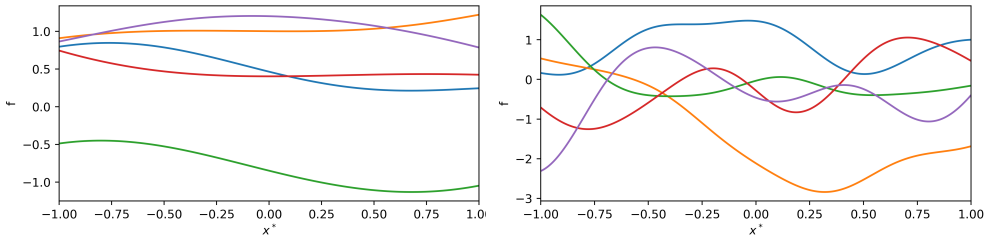
Moreover, there exists a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and a function $m : \mathcal{X} \to \mathbb{R}$ such, that

$$p(f|S) = \mathcal{N}(m(S), K(S)), \forall S = \{x_1, \dots, x_\ell\} \subset \mathcal{X}, \forall \ell \in \mathbb{N}$$

with $m(S) = (m(x_1), \dots, m(x_\ell))$ and $K(S)_{ij} = k(x_i, x_j)$. If m and k are known, we write

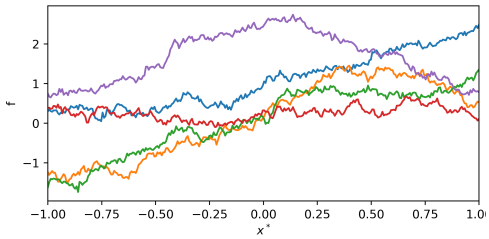
$$f \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot))$$

This relation also holds in reverse: every choice of m and kernel k leads to a random process. In the following, we will first shed light on how kernels affect the Gaussian process as a prior, and we will then use Gaussian processes in a machine-learning task. We end this chapter by discussing how to combine and select kernels.

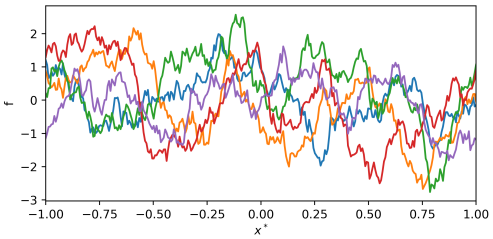


(a) Gaussian Kernel, $\gamma = 0.5$

(b) Gaussian Kernel, $\gamma = 5$



(c) Matern Kernel, $\gamma = 0.5$



(d) Matern Kernel, $\gamma = 5$

Figure 7: Samples taken from a Gaussian Process with $m(x) = 0$ and different choice of kernels and parameters. 300 points are sampled evenly spaced in $[-1, 1]$.

6.1 Kernels and Gaussian Process Priors

As said, the three processes we have investigated before are examples of Gaussian processes:

- Bayesian Linear Regression as random process leads to normal distributed marginals with $m(x) = 0$ and $k(x, x') = \phi(x)\Sigma_\theta\phi(x')$
- The Ornstein-Uhlenbeck process is a Gaussian Process with $m(x) = 0$ and

$$k(x, x') = e^{-\theta|x-x'|}$$

- The Wiener process is a Gaussian Process with $m(x) = 0$ and

$$k(x, x') = \min\{x, x'\}$$

In all cases we have shown the kernel property, by deriving it from the underlying distribution and thus, we have shown directly that these functions lead to symmetric positive definite matrices. Of course, if we are given an arbitrary function k one would need to show first that it is a proper kernel, which means that the matrices $K(S)$ must positive semi-definite, independent of choice of the set S. Showing that an arbitrary function is a kernel can be difficult and in practice kernels are often created from a set of known kernels and are then combined in a way that keeps the kernel property.

As described above, kernels derived from Bayesian linear regression can lead to positive semi-definite kernel matrices and we connected that to the idea that at some point the number of observations available is enough to fully specify each parameter in the model. This begs the question: can we find kernels in which $K(S)$ is always positive definite? This would imply that among the functions drawn from a Gaussian Process with that kernel, we could always find a function that fits the data perfectly. The answer is yes and we call these kernels *universal*:

Definition (Universal Kernel): If k is a kernel and for all $S = \{x_1, \ldots, x_\ell\} \subset \mathcal{X}$ with $x_i \neq x_j, i \neq j$ additionally holds

$$K(S) \text{ is positive definite}$$

Then, we call k universal.

Showing that a kernel is universal is no easy feat and there are very general theorems stating under which conditions a kernel is universal. Two commonly used universal kernels are

1. Gaussian kernel
$$k(x, y) = \exp(-\gamma \|x - y\|^2)$$
2. Matern 3/2 kernel
$$k(x, y) = \left(1 + \gamma\sqrt{3}\|x - y\|\right) \exp\left(-\gamma\sqrt{5}\|x - y\|\right)$$

The choice of kernel has a huge impact on the functions sampled from the Gaussian Process. We can get a better idea of what types of functions the kernel generates by taking S as 300 points on an evenly spaced grid in [-1,1] and plotting the resulting function values, which we show in Figure 7. As we can see, the functions sampled from the Gaussian processes vary a lot. While the Gaussian Kernel generates functions with several smooth hills and valleys, the Matern 3/2 kernel generates much more rugged landscapes. For both kernels the choice of γ affects the width of the valleys.

It is not directly obvious to what degree the choice of kernel matters for the Gaussian process, but we already know that universality does not limit us from sampling diverse shapes of functions: if we sample a set of observations according to $p(f|S)$ of a GP using the Matern kernel, then evaluating its probability under a Gaussian process using a Gaussian kernel — or a Matern kernel with different choice of γ — will lead to a small value. It will turn out that both the choice of kernel and its hyper parameters — here γ— are important for the performance of Gaussian Processes in practice. We will discuss this more in the next sections.

As an end to this section, we introduce two important theorems that bind Gaussian processes, kernels, and Bayesian Linear Regression tightly together. Due to their complexity and reliance on additional abstract concepts of Reproducing Kernel Hilbert Spaces, we show them in a simplified version and without proof.

The first is a characterisation of what a kernel is called Mercer's Theorem

Theorem (Mercer's, simplified): Let $\mathcal{X} \subset \mathbb{R}^d$ be compact and bounded. Let $k : \mathcal{X} \times \mathcal{X}$ be a kernel. Then, there exists a sequence of features $\phi_1, \phi_2, \ldots$ such, that

$$k(x, x') = \sum_{i=1}^{\infty} \phi_i(x)\phi_i(x')$$

For universal kernels the sequence is infinite.

Mercers theorem is a generalization of the result we obtained with Bayesian linear regression. The kernel we obtained there, $k(x, x') = \phi(x)\Sigma_\theta\phi(x')$ can be rewritten in the form of Mercer's theorem. We can write $\Sigma_\theta = AA^T$ and thus $k(x, x') = \phi'(x)\phi'(x') = \sum_{i=1}^{N} \phi'_i(x)\phi'_i(x')$ with $\phi'(x) = A^T\phi(x)$. With this we have established that each kernel can be reinterpreted as an advanced form of Bayesian linear regression, with the additional bonus that for some kernels the corresponding feature map would be infinite dimensional. This connection can be made stronger using the following theorem:

Theorem (Karhunen-Lowe, simplified): Let $f \sim \mathcal{GP}(0, k(\cdot, \cdot))$, ϕ_i given by Mercer's theorem and $x \in \mathcal{X}$, then the random variable

$$\tilde{f}_N = \sum_{i=1}^{N} \theta_i\phi_i(x),\ \theta_i \sim \mathcal{N}(0, 1)$$

converges to f_x as $N \to \infty$, where convergence is measured in squared norm.

This theorem represents the other direction: sampling from a Gaussian Process can be modeled as a prior on some hidden set of parameters θ_i that are used to scale the features $\phi_i(x)$, just that in the case of a universal kernel, we might have an infinite number of parameters in our model.

6.2 Machine-Learning using Gaussian Processes

In the previous section, we have seen an equivalence between the prior of Bayesian linear regression and Gaussian Processes, with the advantage for Gaussian Processes that they can represent functions with possibly infinitely many parameters. What we are missing so far is any practical use of this tool.

For this section, we assume a similar setup as in Bayesian linear regression. We are given a set of data points $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \dots, (x^{(\ell)}, y^{(\ell)})\}$. The labels are given by a noisy observation $y = g(x) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_y^2)$. Our goal is to learn a likely set of functions that approximate $g(x)$ and we aim to predict a function value f^* at location x^* with $f^* \approx g(x^*)$. Unlike Bayesian linear regression, we do not assume a set of features and prior parameter values, but instead that $f \sim \mathcal{GP}(0, k(\cdot,\cdot))$, where k is some kernel[1]. We will refer to the set of known data point locations as $S = \{x^{(1)}, \dots, x^{(\ell)}\}$. With this, we can write the likelihood of the posterior function of y^* after observing $\mathcal{D}$ as

$$p(f^*|x^*, \mathcal{D}) = \frac{\int p(y|f_S)p(f^*, f_S|S \cup x^*)df_S}{p(y|S)} \tag{71}$$

To understand this, we have to understand and derive the different parts. The distribution $p(f^*, f_S|S \cup x^*) = p(f_{x^*}, f_{x^{(1)}}, \dots, f_{x^{(\ell)}})$ is the marginal of the Gaussian process of function values observed at locations in S and x^*. This is the prior distribution of function values we expect to observe. The distribution $p(y|f_S)$ is then the probability of observing the vector of y-values given a chosen set of function-values f_S at the observed locations in S, which is the difference produced by the random noise. Thus $p(y|f_S) = \mathcal{N}(y; f_S, \sigma_y^2 I_\ell)$.

In equation (71) we then integrate over f because we never observed the true function values at S, but only the noisy labels. This gives us the distribution

$$p(y, f^*|S \cup x^*) = \int p(y|f_S)p(f^*, f_S|S \cup x^*)df_S$$

and the last step is to condition on the observations y

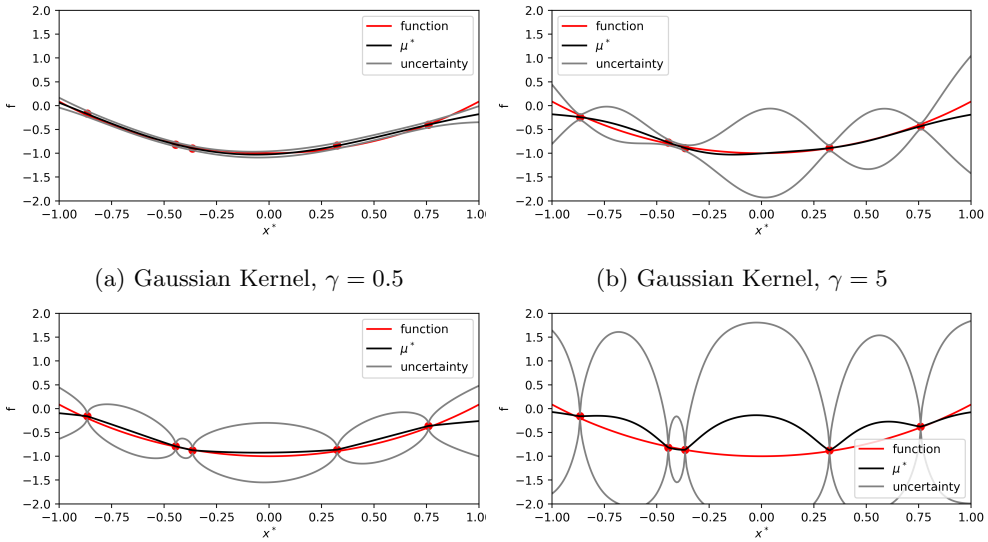
$$\begin{aligned} p(f^*|y, S \cup x^*) &= \frac{p(y, f^*|S \cup x^*)}{\int p(y|S \cup x^*)} \\ &= \frac{\int p(y|f_S)p(f^*, f_S|S \cup x^*)df_s}{p(y|S \cup x^*)} \quad . \end{aligned}$$

This leads to equation (71), except the normalization constant, which we can show does not depend on x^* when performing the integration over f^* carefully:

$$\begin{aligned} p(y|S \cup x^*) &= \int p(y, f^*|S \cup x^*)df^* \\ &= \int \int p(y|f_S)p(f^*, f_S|S \cup x^*)df_S df^* \\ &= \int p(y|f_S) \underbrace{\int p(f^*, f_S|S \cup x^*)df^*}_{p(f_S|S)} df_S \\ &= \int p(y|f)p(f_S|S)df = p(y|S) \quad . \end{aligned} \tag{72}$$

In the third step we reordered the integrals in order to integrate over f^* and the result is due to Kolmogorovs consistency theorem. In itself, this quantity is the probability that our observations are generated from the Gaussian process with added noise, and is called the evidence.

[1] The mean function $m(x)$ can also be included here, but in most applications this is just 0 or some other constant and we ignore it for simplicity. Choosing a mean function is equivalent to fitting a Gaussian Process with 0-mean to $y^{(i)} - m(x^{(i)})$.



(a) Gaussian Kernel, $\gamma = 0.5$

(b) Gaussian Kernel, $\gamma = 5$

(c) Matern Kernel, $\gamma = 0.5$

(d) Matern Kernel, $\gamma = 5$

Figure 8: Visualisation of the posterior $p(f^*|x^*, \mathcal{D})$ and different choice of kernels and parameters. Points and underlying true function $f(x) = e^x + e^{-x} - 3$ in red, $x^* \in [-1, 1]$. The black line visualizes the mean μ^* for chosen x^*. Grey lines mark the 95% confidence interval, $\mu^* \pm 1.96\sigma^*$.

Computing the actual pdf $p(f^*|x^*, \mathcal{D})$ by computing the integrals directly is cumbersome. Moreover, if the kernel is not universal, we run the risk of computing densities of normal distributions that do not have positive definite covariance. Instead we again use the generating model to quickly compute the distribution using the rules we have established in Section 2.

We first begin by writing the distribution of $p(f^*, f|S \cup x^*)$ in block-form:

$$\left[\frac{f}{f^*}\right] \sim \mathcal{N}\left(0, \left[\begin{array}{c|c} K(S) & k(S, x^*) \\ \hline k(S, x^*)^T & k(x^*, x^*) \end{array}\right]\right)$$

The matrix can directly be obtained by insertion into the definition of a Gaussian process where the new position x^* is added as last element to S. In block-notation we added the following quantities: The matrix $K(S)$ is the previously introduced kernel-matrix, $K(S)_{ij} = k(x^{(i)}, x^{(j)})$. We further introduced $k(S, x^*)$ which is a vector with elements $k(S, x^*)_i = k(x^{(i)}, x^*)$ and $k(x^*, x^*)$ is the kernel applied to x^* at both arguments.

Next, in our generative model, the noise variable $\epsilon \sim \mathcal{N}(y; 0, \sigma_y^2 I_\ell)$ is added to the values of f — but not f^* as we are interested in predicting the most likely true function value, and not the value that is disturbed by measurement noise. In block notation, this means:

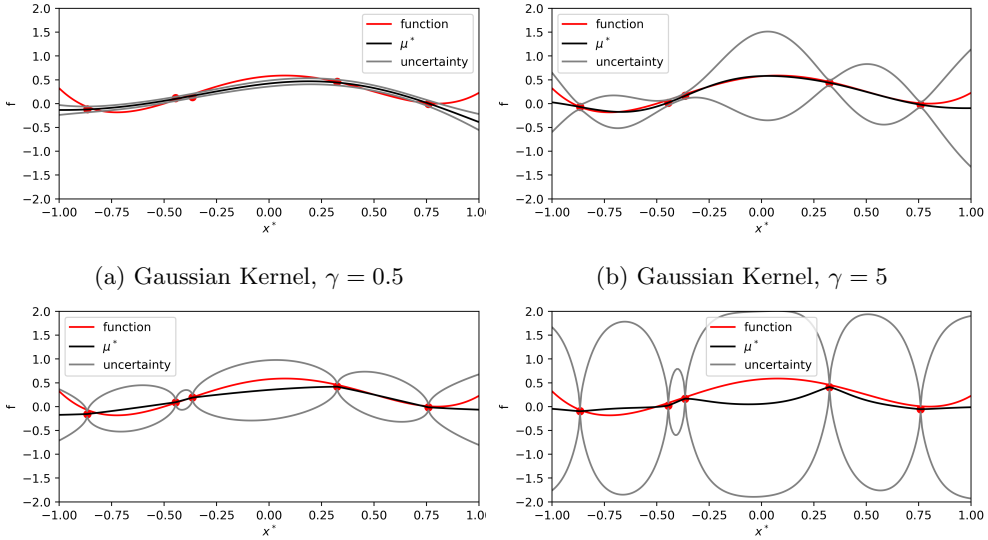
$$\left[\frac{y}{f^*}\right] = \left[\frac{f}{f^*}\right] + \left[\frac{\epsilon}{0}\right] \tag{73}$$

Even though the last element of the second vector is 0, it is still a sample of a multivariate normal distribution (even though it does not have a pdf), with distribution

$$\left[\frac{\epsilon}{0}\right] \sim \mathcal{N}\left(0, \left[\begin{array}{c|c} \sigma_y^2 I_\ell & 0 \\ \hline 0 & 0 \end{array}\right]\right) .$$

Thus, the distribution of the left hand side of equation (73) is a normal distribution with density of a sum of two normal distributions, and we can use rule (32) and obtain

$$\left[\frac{y}{f^*}\right] \sim \mathcal{N}\left(0, \left[\begin{array}{c|c} K(S) + \sigma_y^2 I_\ell & k(S, x^*) \\ \hline k(S, x^*)^T & k(x^*, x^*) \end{array}\right]\right) . \tag{74}$$



(a) Gaussian Kernel, $\gamma = 0.5$

(b) Gaussian Kernel, $\gamma = 5$

(c) Matern Kernel, $\gamma = 0.5$

(d) Matern Kernel, $\gamma = 5$

Figure 9: Visualisation of the posterior $p(f^*|x^*, \mathcal{D})$ and different choice of kernels and parameters. Points and underlying true function $f(x) = 2(x + 0.9)(x + 0.5)(x - 0.8)^2$ in red, $x^* \in [-1, 1]$. The black line visualizes the mean μ^* for chosen x^*. Grey lines mark the 95% confidence interval, $\mu^* \pm 1.96\sigma^*$.

The last step is obtained by conditioning on the observed values of y using (29). This results in an equation very similar to Bayesian linear regression

$$\begin{aligned} f^*|y &\sim \mathcal{N}(\mu^*, \sigma^*) \\ \mu^* &= k(S, x^*)^T((S) + \sigma_y^2 I_\ell)^{-1} y \\ (\sigma^*)^2 &= k(x^*, x^*) - k(S, x^*)^T((S) + \sigma_y^2 I_\ell)^{-1} k(S, x^*) \end{aligned} \tag{75}$$

As in the normal distribution mean and mode are identical, μ_* is the maximum likelihood estimate and the mean for f^*, and thus a good candidate for prediction of the function value. However, the Gaussian Process also provides us with an uncertainty estimate of this prediction, that we can use to quantify uncertainty further, for example by reporting a confidence interval $\mu^* \pm 1.96\sigma^*$, which is the 95% confidence interval for location of f^*, i.e., 95% of sampled values from $p(f^*|x^*, \mathcal{D})$ lie within this interval.

This suggests a very simple algorithm to train and apply a Gaussian Process model:

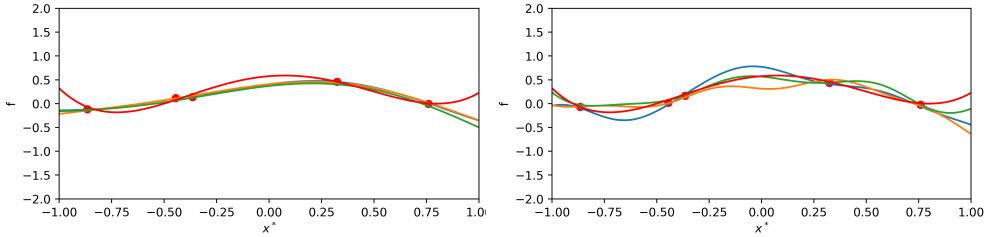
Algorithm (GP, simple):

- Pick kernel $k(\cdot, \cdot)$ and noise variance $\sigma_y > 0$
- Get data $(x_1, y_1), \ldots, (x_1, y_1)$, $S = \{x_1, \ldots x_\ell\}$
- Compute $G = (\sigma_y^2 I_N + K(S))^{-1}$ and $\alpha = Gy$

For a new point x^* to predict:

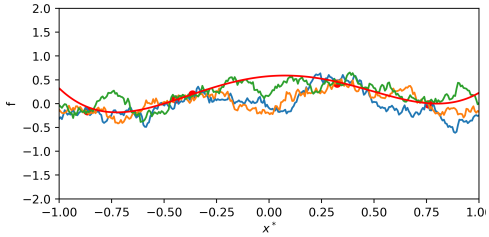
- Compute $\mu_* = k(S, x^*)^T \alpha$
- Compute $(\sigma^*)^2 = k(x^*, x^*) - k(S, x^*)^T G k(S, x^*)$

We can use this algorithm to visualize the behaviour of the GP model for different choices of functions and kernels in a one-dimensional problem (Figures 8& 9). We can see that for different

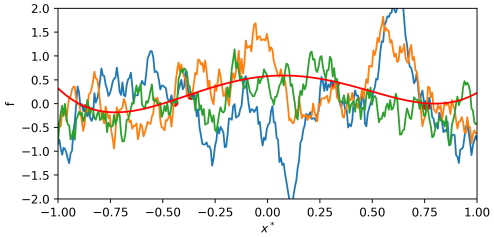


(a) Gaussian Kernel, $\gamma = 0.5$

(b) Gaussian Kernel, $\gamma = 5$



(c) Matern Kernel, $\gamma = 0.5$



(d) Matern Kernel, $\gamma = 5$

Figure 10: Visualisation of the posterior samples and different choice of kernels and parameters. Points and underlying true function $f(x) = 2(x+0.9)(x+0.5)(x-0.8)^2$ in red. Samples are taken by evenly sampling 300 points between $[-1, 1]$ from the posterior.

choices of function the prediction for μ^* vary a lot and also the uncertainties vary by a large amount. This is because Gaussian Processes are a Bayesian model: we chose a prior distribution of f and then predict posterior means and uncertainty. Thus, depending on the kernel our prediction for how the function could continue between observed points will be purely guided by this choice of prior and the uncertainties will follow roughly the behaviour of what we have seen in the prior samples (Figure 7).

The Matern 3/2 kernel shows very sudden changes between points in the prior. In the posterior, this is reflected by the uncertainty rising quickly around an observed point. Similarly, the Gaussian kernel for $\gamma = 0.5$ showed very smooth curves, which are perfect for prediction the first function, with tight mean and variance estimates. In contrast, the same kernel for $\gamma = 5$ produces much smaller valleys and thus the uncertainties become larger for the first function. However, when we look at the polynomial function in Figure 9, the $\gamma = 0.5$ kernel does not model this vary well and predicts the variability of the function poorly, while now the $\gamma = 5$ kernel works comparatively well.

A better way to understand this is to use the GP model not to predict a single point, but several points simultaneously. This amounts to a slight change of derivation where instead of prediction for a single x^*, we predict for a whole second set of points $S^* = \{(x^*)^{(1)}, \ldots, (x^*)^{(T)}\}$ use as prior $p(f_{S^*}, f_S | S^* \cup S)$ which introduces additional dependencies between points contained in S^*. We skip the derivation for sake of shortness of the script, but the same steps apply as in the previous derivations and the reader is invited to produce the equations themselves!

Visualizing this using S^* as the same 300 points as for the priors on the polynomial function, we obtain Figure 10. This visualisation confirms the explanation we have found for the previous two figures: samples from the posterior using the Gaussian kernel tend to be smooth functions, with neighbouring function values being close to each other. We also see that again for $\gamma = 0.5$ the function tends to be overly smooth, while samples from $\gamma = 5$ look like possible extensions that closely align with the change of curvature of the real function. While the samples do not follow the true shapes, this indicates that this kernel gives approximately correct uncertainty estimates. In contrast, the Matern kernel predicts completely different function shapes and is only bound by passing nearby the observed function values. Thus, this kernel would be a bad choice for this type of function.

6.3 Kernel combination & Kernel selection

As we have seen, the choice of kernel has a major effect on the predictions of mean and variance of a Gaussian Process. The question arises, how to pick a kernel? Without any tools, we have to guess which shape of kernel works best and use expert knowledge about the task to choose the kernel. Of course, we can remain with our pick of two kernels with parameters, but often we might guess that a function behaves mostly smooth, but also has slightly rugged features. How can we represent that? We can use rules that combine kernels while ensuring that the result is a kernel, for example:

- Let k_1 be a kernel and $\sigma^2 > 0$ a positive constant. Then $k(x, x') = \sigma^2 k_1(x, x')$ is a kernel
- Let k_1 and k_2 be kernel functions. Then, $k(x, x') = k_1(x, x') + k_2(x, x')$ is a kernel
- Let k_1 be a kernel and $a > 0$ a positive constant. Then $k(x, x') = k_1(x, x') + a$ is a kernel

The first rule allows to scale a kernel. This is very important in Gaussian Processes. To understand it, let $K_1(S)$ be the kernel matrix produced by kernel k_1 when applied to set S. Then for the scaled kernel k, we have $p(f|S) = \mathcal{N}(0, \sigma^2 K_1(S))$. In this case, the scaling value takes the role of scaling the variance of the normal distribution. This is important for example for the Gaussian kernel as for it the diagonal $K(S)_{ii} = 1$ and thus it can not model functions with large values very well. The third rule finally is a special case of the second rule. adding a constant is the same as adding the constant kernel $k(x, x') = a$. Its effect is adding a constant to the predicted function values. The three rules together allow to learn arbitrary scale and shifts of function values. (e.g., when learning a function with values between -10 and 100 with most values centered around 40, the standard Gaussian kernel is a bad fit).

Further, the rules allow to combine different kernels with different scales in order to produce kernels which have properties of both. Still, in many cases, we don't have a good idea for the values of the kernel parameters, their scaling values and the added measurement noise σ_y^2. How can we select these values in a systematic manner?

For this, let $p(f|S, \eta)$ be the marginal of the GP with the chosen hyperparameters, where η is a vector representing all parameters of the chosen kernel $k_\eta(x, x')$. A popular approach is to find the choice of η, σ_y^2 that maximizes the evidence (72).

The evidence, depending on the parameters is

$$p(y|S, \eta, \sigma_y^2) = \int p(y|f, \sigma_y^2) p(f|S, \eta) df$$

And the noisy observations have distribution

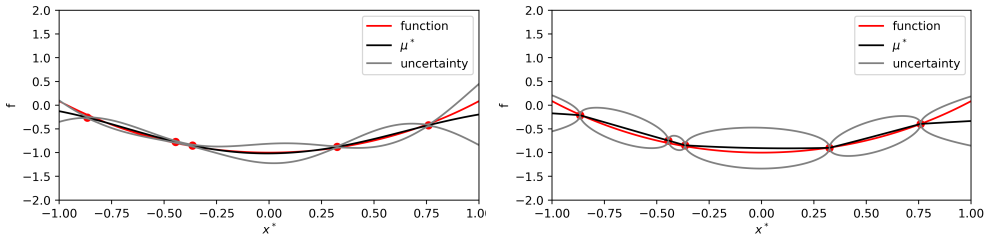
$$y \sim \mathcal{N}\left(0, \sigma_y^2 I_\ell + K_\eta(S)\right) \quad .$$

Here, $K_\eta(S)$ is the kernel matrix of kernel k_η. The pdf of the distribution (equation (26)) computes the likelihood that the Gaussian process with added noise would produce the observation y and thus, finding the parameters that maximize the log-likelihood looks like a good solution. The log-likelihood of the evidence is:

$$\log p(y|S, \eta, \sigma_y^2) = -\frac{1}{2} y^T (\sigma_y^2 I_\ell + K_\eta(S))^{-1} y - \frac{1}{2} \log \det(\sigma_y^2 I_\ell + K_\eta(S)) - \frac{\ell}{2} \log \sqrt{2\pi}$$

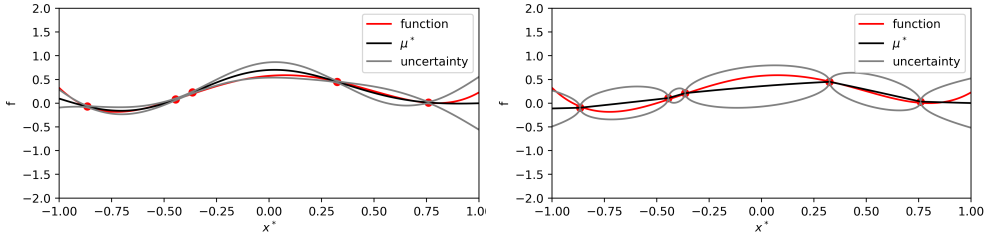
Finding the maximum of this function in terms of η and σ_y^2 is surprisingly difficult as it has several local optima and gradient-descent algorithms often get stuck at a poor local minimum. A good strategy is to use grid-search or random-search to obtain a good starting point. Further, because we optimize the log-likelihood, we have to ensure that the obtained covariance matrices are positive definite, and sufficiently so to not suffer from numerical difficulties. Thus, universal kernels should be favoured or lower bounds put in place that prevent σ_y^2 is chosen too small.

We again visualize the effects of tuning the parameters. This time, we will pick three different functions, $f_1(x) = e^x + e^{-x} - 3$, $f_2(x) = 2(x + 0.9)(x + 0.5)(x - 0.8)^2$ and $f_3(x) = f_1(x) + 3$. The added function is here to visualize the effect of a mean-shift if the underlying kernel can not handle



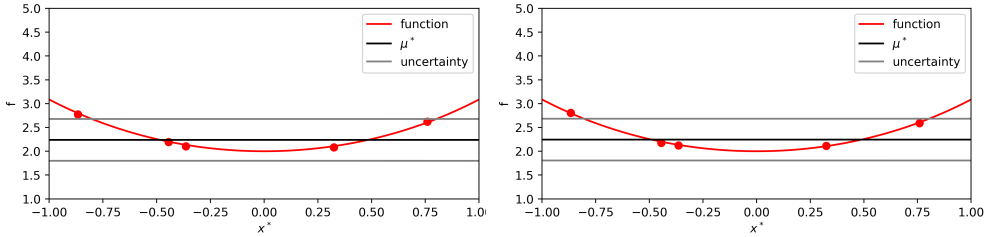
(a) Gaussian Kernel, f_1

(b) Matern Kernel, f_1



(c) Gaussian Kernel, f_2

(d) Matern Kernel, f_2



(e) Gaussian Kernel, f_3

(f) Matern Kernel, f_3

Figure 11: Visualisation of the posterior $p(f^*|x^*, \mathcal{D})$ for different functions and optimized parameters for Gaussian and Matern 3/2 kernels. Points and underlying true functions in red, $x^* \in [-1, 1]$. The black line visualizes the mean μ^* for chosen x^*. Grey lines mark the 95% confidence interval, $\mu^* \pm 1.96\sigma^*$.

arbitrary means. We visualize both the Gaussian and Matern Kernel with their single parameters as well as optimized σ_y^2, without using the additional scaling parameters of the rule introduced before. As grid we use 20 evenly spaced points in $[10^{-4}, 5]$ on both parameters, leading to 400 points evaluated in total. The results are given in Figure 11. The optimized Gaussian Process parameter for f_1 and f_2 in a)-d) make sense and closely follow the function shapes. While the Matern Kernel still shows a much quicker increase of the variance, it does not vastly under or overestimate the uncertainty and the true function is within the learned 95% confidence intervals. For the third function, e)+f), both approaches do not provide reasonable estimates. This is because both kernels can not learn mean-shifts of function values and instead prefer functions with mean close to 0. The result is that after observing a large value at a location, the Gaussian Process with these kernels tends to predict lower values, to get back to the assumed prior assumption of values having 0 mean (an example for this can be seen for the Matern Kernel in Figure 8d)). The only way to prevent this is to choose $\gamma \approx 0$ as this makes the kernels wide, allowing it to predict an almost constant function. And indeed, the result for γ is the smallest value of the grid, 10^{-4}. To model the variations in function values, the noise variance σ_y^2 is increased, modeling all deviations from the mean as noise.

While this explanation sounds hand-wavy, the observation can also be predicted from the math. The mean prediction in our algorithm is written as $\mu^* = k(S, x^*)^T \alpha = \sum_{i=1}^{\ell} \alpha_i k(x^{(i)}, x')\alpha_i$. The mean prediction function is a linear combination of kernel functions. With the Gaussian kernel the function is a linear combination of Gaussian bell functions, each converging to 0 on their tails. This creates the aforementioned trend of converging back to 0 and this is alleviated as the Gaussian kernel becomes wider, and thus locally more constant.

While this is a poor solution, it is the only one that the kernels have available that makes the data likely, thus if the function modeled does not align with the underlying kernel assumptions, the predicted means, variances and learned noise-levels do not necessarily make sense anymore. Does this mean we have to include the scaling and shift parameter mentioned earlier? Not necessarily. In many cases, we can just get by by normalising the observations y to something that fits the kernel. In our case we can scale the labels to have zero mean, unit variance. Then we can learn a Gaussian Process on the normalized data and during prediction undo the normalization for the predicted values. This is standard practice in many applications.

7 Denoising Diffusion Probabilistic Models

This section is intended to give a basic introduction to diffusion models and present the math required to implement a basic diffusion model based on Denoising Diffusion Probabilistic Models[1]. While this is the original idea of diffusion models and there have been many attempts at improving this approach, the basic idea is still used with only very minor modifications, if at all.

Diffusion Models are a subclass of the broad area of generative models. In generative modeling the dataset is a set of points $\mathcal{D} = \{x^{(1)}, \dots, x^{(\ell)}\}$, $x^{(i)} \in \mathbb{R}^d$, and the task is to approximate the underlying data distribution $p_{\mathcal{D}}(x)$ in order to sample new data points from the distribution.

Diffusion Models introduce a diffusion process of time depending random variables $X_0, X_1 \dots, X_T$, where each variable is a vector the size of the original input, $X_t \in \mathbb{R}^d$. In this process, the variables are related by a Markov Chain with conditional distribution $q(X_{t+1}|X_t)$ that, starting with samples from the data distribution $X_0 \sim q_{\mathcal{D}}$, transforms the observations step-by-step into a distribution of pure noise $X_T \sim \mathcal{N}(0, I)$. This is also sometimes called the forward process, and in most diffusion models, this process is fully specified and has no free parameters.

The task of the diffusion model is then to revert this process, that is, the model represents a set of distributions $p(X_t|X_{t+1})$ that each try to learn the reverse distribution of the Markov Chain, $p(X_t|X_{t+1}) \simeq q(X_t|X_{t+1})$. As the forward model repeatedly adds noise, the reverse process is also called a denoising process. With such a model learned, we can generate new samples from $q_{\mathcal{D}}$ by simply sampling $x_T \sim \mathcal{N}(0, I)$ and then sampling $p(x_{T-1}|x_T)$, $p(x_{T-2}|x_{T-1})$, $\dots$, $p(x_0|x_1)$, where x_0 is then an approximate sample of $q_{\mathcal{D}}$.

Similar to variational autoencoders, $p(X_t|X_{t+1})$ can be chosen as $\mathcal{N}(\mu_t(X_{t+1}), \Sigma_t(X_{t+1})$, a normal distribution where both μ_t and Σ_t can be parameterized via deep neural networks. This model has a lot of similarities with stacked, or ladder-VAEs, where each deeper layer learns another layer of the prior. However, as the forward process is fully specified, there is an efficient way to train diffusion models that does not suffer from the many shortcomings of stacked VAEs.

Before we take a look at how the denoising is performed, we have to start with the noise process: how can we systematically define a Markov Chain such, that, starting from an image, we can arrive at pure noise, while we can also easily tune the number of generation steps. A standard choice is to use the Ornstein-Uhlenbeck (OU) process, which we have introduced in Section 5.3. We will reintroduce it in the parameterization of diffusion literature,

We define the forward process in the following way: we first assume that all variables (for example pixels of a grayscale image) in X_t with observations in $\mathbb{R}^d$ are independent of each other and the forward process is defined by an OU process for each variable. The result is that at each step the process defines a multivariate normal distribution with diagonal covariance matrix.

Using the definition of the process defined in Section 5.3, we arrive at the pdf of the forward distribution:

$$q(X_t|X_{t-1}) = \mathcal{N}(X_t; a_t X_{t-1}, (1 - a_t^2) I_d)$$

where I_d is the $d \times d$ identity matrix. In our derivation of the OU process, $a_t \in [0, 1]$ was a function of the time difference between two observations. In diffusion literature, we instead parameterize it as

$$q(X_t|X_{t-1}) = \mathcal{N}(X_t; \sqrt{1 - \beta_t} X_{t-1}, \beta_t I_d) \ ,$$

setting $\beta_t = (1 - a_i^2) \in [0, 1]$. Since we fix the random variables $X_0, \dots, X_T$, we can finally parameterize all steps directly by specifying the sequence of $\beta_1, \dots, \beta_T \in [0, 1]$ used.

This specifies the whole process except X_0 for which we pick the data distribution $q_{\mathcal{D}}$. With all of this, we arrive at the forward process

$$q(X_0, \dots, X_T) = q_{\mathcal{D}}(X_0) \prod_{t=1}^{T} q(X_t|X_{t-1}) \ .$$

This distribution can be visualized in simple cases. In Figure 12 we give the result for the three starting points from Figure 5. In this figure, each column is a visualization of the pdfs $p(X_t)$ taking $p(X_0)$ as the data distribution of three points at $\{-5, 0, 5\}$, where lighter areas means higher probability. For small t the marginal distributions are multi-modal distributions with one peak for each point. As t increases, the peaks get closer and then vanish into a single broad one, that then becomes ever closer to a standard normal distribution.

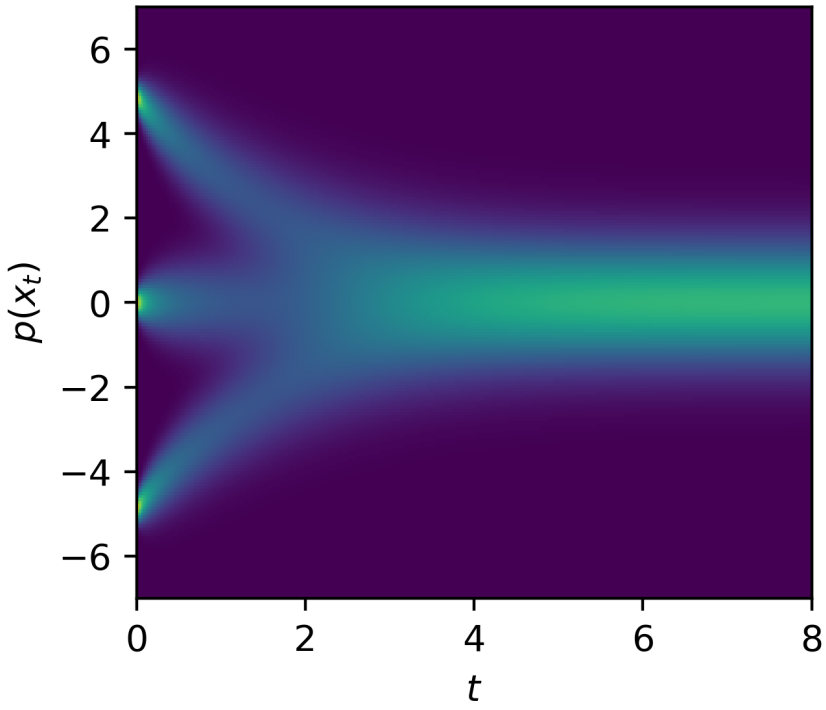


Figure 12: Marginal distributions of the OU process over the different starting points in Figure 5.

7.1 Denoising Diffusion Models

Having specified the forward process, our goal is to learn a model of the reverse process

$$p_\theta(X_0, \dots, X_T) = p_\theta(X_T) \prod_{t=1}^{T} p_\theta(X_{t-1}|X_t) \ ,$$

where θ is the vector of model parameters and each conditional distribution is chosen as a normal distribution

$$p_\theta(X_{t-1}|X_t) = \mathcal{N}(X_{t-1}; \mu_\theta(X_t, t), \beta_t I_d) \ . \tag{76}$$

We fixed the variance here to β_t for simplicity of presentation. While other choices are possible, this is a common choice. Typically, μ_θ is chosen as a deep neural network, but we will take a closer look at the exact form later. For now we treat μ_θ as a black box.

Direct optimization of $p_\theta(X_0)$ via maximum likelihood is not feasible, as it requires marginalization of variables $X_1, \dots, X_T$. However, when we treat the forward process $q(X_0, \dots, X_T)$ as the real data distribution of interest, we can use maximum likelihood to learn $p(X_0, \dots, X_T)$. With our model choice of p_θ, the cross entropy can be split into several independent terms:

$$
\begin{aligned}
&- \int q(X_0, \dots, X_T) \left[\log p_\theta(X_0, \dots, X_T)\right] \, dX_0, \dots, dX_T \\
&= - E_{X_0,\dots,X_T} \left[\log p_\theta(X_0, \dots, X_T)\right] \\
&= - E_{X_0,\dots,X_T} \left[\log p_\theta(X_T) + \sum_{t=1}^{T} \log p_\theta(X_{t-1}|X_t)\right] \\
&= - E_{X_0,\dots,X_T} \left[\log p_\theta(X_T)\right] - \sum_{t=1}^{T} E_{X_0,\dots,X_T} \left[\log p_\theta(X_{t-1}|X_t)\right] \\
&= - E_{X_0,X_T} \left[\log p_\theta(X_T)\right] - \sum_{t=1}^{T} E_{X_0,X_{t-1},X_t} \left[\log p_\theta(X_{t-1}|X_t)\right]
\end{aligned}
$$

In the derivation, we used first that the product of probabilities in p_θ becomes a sum of log-probabilities, which allows splitting the distribution. In the last step, we used that random variables which do not appear in the expectation can be marginalized out. However, we kept X_0 for future derivation steps.

The first term including $p_\theta(X_T)$ can be ignored if we fix $p_\theta(X_T)$. As we have seen before, the OU process will eventually converge to the standard normal distribution as time passes, and thus with enough steps and large enough β_t, $p_\theta(X_T) \sim \mathcal{N}(X_T; 0, I_d)$ is a reasonable choice. The second term over the sum is more interesting. We have

$$
E_{X_0,X_{t-1},X_t} \left[\log p_\theta(X_{t-1}|X_t)\right] = \int q(X_0, X_{t-1}, X_t) \log p_\theta(X_{t-1}|X_t) dX_0, dX_{t-1}, dX_t
$$

The pdf can be rewritten as

$$
q(X_0, X_{t-1}, X_t) = q_{\mathcal{D}}(X_0) q(X_{t-1}|X_0) q(X_t|X_{t-1}) \ .
$$

The first factor is the true data distribution, the third factor is the pdf of the distribution (76) and the middle factor is the conditional of the OU process when marginalizing over variables $X_1, \dots, X_{t-2}$. This marginal is given by

$$
\begin{aligned}
X_{t-1}|X_0 &\sim \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}} X_0, (1 - \bar{\alpha}_{t-1}) I_d\right) \\
\bar{\alpha}_{t-1} &= \prod_{i=1}^{t-1} (1 - \beta_i) \ .
\end{aligned}
$$

Given X_0, all remaining factors of this distribution are normal distributions. Thus, we can hope to compute some of the integrals of the cross-entropy analytically. The naive approach of integrating over X_t does not work, since the dependency of $\mu_\theta(X_t, t)$ on X_t can be arbitrarily complicated. However, we can integrate over X_{t-1}. reordering integrals gives:

$$
E_{X_0,X_{t-1},X_t} \left[\log p_\theta(X_{t-1}|X_t)\right] = E_{X_0,X_t} \left[\int q(X_{t-1}|X_t, X_0) \log p_\theta(X_{t-1}|X_t) dX_{t-1}\right]
$$

The distribution $q(X_{t-1}|X_t, X_0)$ can be computed using (29)

$$
\begin{aligned}
X_{t-1}|X_t, X_0 &\sim \mathcal{N}\left(\tilde{\mu}_{t-1}(X_0, X_t), \tilde{\beta}_t\right) \\
\tilde{\mu}_{t-1}(X_0, X_t) &= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} X_0 + \frac{\sqrt{1 - \beta_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} X_t \\
\tilde{\beta}_t &= \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t
\end{aligned}
\tag{77}
$$

With this distribution, we can finally compute the integral and after some tedious calculations, we arrive at

$$
-E_{X_0,X_{t-1},X_t} \left[\log p_\theta(X_{t-1}|X_t)\right] = E_{X_0,X_t} \left[\frac{1}{2\beta_t} \|\tilde{\mu}_{t-1}(X_0, X_t) - \mu_\theta(X_t, t)\|^2\right] + C_t
$$

where C_t is some constant that only depends on the parameters of the forward process and which can be ignored during training. It is worthwhile to ponder over this result for a second. Since our goal is to learn the optimal μ_θ, what would be the global optimum of this process? Computing the minimum reveals that the optimum is

$$\mu_\theta(X_t, t) = E_{X_0}\left[\tilde{\mu}_{t-1}(X_0, X_t)\right] \quad .$$

This is because the model itself does not know about X_0 and thus has to guess the most likely point, which is the center of mass for all possible X_0. This is not an equally weighted mean as conditioned on X_t, $q(X_0|X_t)$ is not a uniform distribution. This is especially true for small t, as some X_0 are very unlikely to give rise to a particular X_t. Thus the optimal model guesses which X_0 are most likely to have generated an X_t and then makes a step along the most likely diffusion path towards them. This works well in our diffusion process, as the added noise in every step smoothes out the distribution.

Still, while it is quite intuitive, this is not the variant that is used for training a diffusion model. For the next step, we want to use the reparameterisation trick to eliminate X_0 from the term inside the expectation. For this, we write X_0 in terms of X_T using the relation

$$\begin{aligned} X_T &= \sqrt{\bar{\alpha}_t} X_0 + \sqrt{1-\bar{\alpha}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, I_d) \\ \Leftrightarrow X_0 &= \frac{1}{\sqrt{\bar{\alpha}_t}}\left(X_T - \sqrt{1-\bar{\alpha}_t}\epsilon\right) \quad . \end{aligned}$$

Inserting this into $\tilde{\mu}_{t-1}$ simplifies the expression and skipping a large number of tedious steps, we arrive at

$$\begin{aligned} \tilde{\mu}_{t-1}(X_0, X_t) =& \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} X_0 + \frac{\sqrt{1-\beta_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} X_t \\ =& \cdots = \frac{1}{\sqrt{1-\beta_t}}\left(X_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right) \quad . \end{aligned} \tag{78}$$

This is a relatively simple function, that only depends on X_t and the noise ϵ that was used to generate X_t from X_0.

We are now ready to pick μ_θ. There are many different ways to pick the form, but when we look at (78), we see that this term does only depend on X_t and ϵ which are modulated by some pre-factors which are known. Thus, we can make our life easier by picking our model function as

$$\mu_\theta(X_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(X_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(X_t, t)\right) \quad . \tag{79}$$

Here, ϵ_θ is the deep neural network. In this parameterisation, the model does not have to learn to predict the different variances as the pre-factors automatically set the correct scales. Moreover, this parameterisation only requires the model to encode the difference relative to X_t, and it does not need to learn the identity of X_t. Inserting both definitions in the squared distance term, leads to

$$\begin{aligned} &\frac{1}{2\beta_t}\|\tilde{\mu}_{t-1}(X_0, X_t) - \mu_\theta(X_t, t)\|^2 \\ =& \frac{1}{2\beta_t}\left\|\frac{1}{\sqrt{1-\beta_t}}\left(X_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right) - \frac{1}{\sqrt{1-\beta_t}}\left(X_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(X_t, t)\right)\right\|^2 \\ =& \frac{\beta_t}{2(1-\beta_t)(1-\bar{\alpha}_t)}\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}X_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|^2 \quad . \end{aligned}$$

and inserting this into our learning objective, results in:

$$E_{X_0, X_{t-1}, X_t}\left[p_\theta(X_{t-1}|X_t)\right] = \frac{\beta_t}{2(1-\beta_t)(1-\bar{\alpha}_t)} E_{X_0, \epsilon}\left[\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}X_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|^2\right] \quad .$$

Putting everything together, training a diffusion model is the same as minimizing the objective

$$E_t\left[\frac{\beta_t}{2(1-\beta_t)(1-\bar{\alpha}_t)}E_{X_0,\epsilon}\left[\left\|\epsilon-\epsilon_\theta(\sqrt{\bar{\alpha}_t}X_0+\sqrt{1-\bar{\alpha}_t}\epsilon,t)\right\|^2\right]\right] \quad , \tag{80}$$

where the expectation over t is the naive average over all choices of t. Using this objective directly does often perform poorly. The reason is that depending on the sampled t, the pre-factor $1/(1-\bar{\alpha}_t)$ can become very large and thus when sampling t uniformly, the Monte-Carlo estimate of equation (80) can have huge variance. For this reason, the scaling factors are often ignored and the objective then reads

$$E_{t,X_0,\epsilon}\left[\left\|\epsilon-\epsilon_\theta(\sqrt{\bar{\alpha}_t}X_0+\sqrt{1-\bar{\alpha}_t}\epsilon,t)\right\|^2\right] \quad . \tag{81}$$

The final objective is the MSE of the model in predicting the noise that was used to generate X_t from X_{t-1}. We can optimize it with stochastic gradient descent, by picking a t and X_0 uniformly at random and then generating $\epsilon \sim \mathcal{N}(0, I_d)$. As the model only observes X_t and never X_0, its best guess has to be predicting the mean of $\epsilon|X_t$, where X_t is the distribution of the generated X_t marginalized over the complete dataset.

Generating from the learned model After training using (81), we have a trained model for $\epsilon_\theta(X_t, t)$. How can we generate samples form this? We just have to put all the pieces of the model together. We start with sampling $X_0 \sim \mathcal{N}(0, I_d)$. Then, we generate by following the Markov-Chain backwards step by step, using

$$X_{t-1}|X_t \sim \mathcal{N}\left(\mu_\theta(X_t,t), \beta_t I_d\right) \quad , \tag{82}$$

where $\mu_\theta(X_t, t)$ is given by (79). The sample X_0 is then an approximate sample from $q_{\mathcal{D}}$.

7.2 Diffusion in Practice

In the previous section we have shown that training a diffusion model is simple, as only (81) needs to be implemented. However, obtaining good results with diffusion is still challenging. There are a number of problems.

- Generation: Diffusion models typically require a large T, we often see $T > 10000$. Thus, to create a single sample, we have to use equation (82) thousands of times. Each call requires as to evaluate a neural network.
- High dimensional data: The previous problem becomes a lot worse when the data distribution describes high dimensional data like images. Thus if we implement naive diffusion in this setting, each generation step itself becomes a big computational task as a large deep neural network needs to be used.
- Model choices: It is difficult to come up with neural network models that perform well at denoising diffusion. We need a model class that can propose fine grained difference vectors based on a given sample as a function of time. Including the time component turns out to be challenging.

In this section, we will focus on the third point, but will quickly summarize the current status of the field for the first two points.

A large T is required in our model since we choose as variance of the reverse process the covariance matrix $\beta_t I_d$. It can be shown, based on theory of stochastic differential equation that this choice is optimal, if β_t is small. This means that as $T \to \infty$ we require that $\beta_t \to 0$ for all $t = 1, \ldots, T$. Of course this does not mean that $\beta_t \propto 1/T$ is optimal in practice. A schedule that performs empirically well is given by [3]

$$\beta_t = \min\{1-\frac{g(t)}{g(t-1)}, 0.999\},\ g(t) = \cos^2\left(\frac{\pi}{2}\frac{t/T+s}{1+s}\right) \quad ,$$

where s is a free parameter that should be small and the minimum is taken to prevent numerical difficulties. In [3], the parameters used were $s = 0.008$ and $T = 4000$ for an image generation task.

Many successful approaches nevertheless opt to adapt (76) to also predict the variance,

$$p_\theta(X_{t-1}|X_t) = \mathcal{N}(X_{t-1}; \mu_\theta(X_t, t), \Sigma_\theta(X_t, t)) \ .$$

However this is not straight forward and a number of tricks are required to make this work, see [3]. Another alternative is to choose an different forward diffusion process. Since the OU-Process is the only Stationary Gaussian Process with the Markov property, these approaches tend to be non-Markovian. An example are DDIM introduced in [5].

High dimensional data poses practical problems, especially in the image and video domains. A common approach to solve this is by using autoencoders to compress the high dimensional data into lower dimensional representations [4]. This is based on the observation that in practice it is relatively easy to create an autoencoder that maps data to a latent representation and back to the original data without noticeable loss in data fidelity. What is difficult for autoencoders is to perform this mapping in a way that is easy to sample from the latent space. Thus, the pair of diffusion together with autoencoders uses the strengths of both approaches: use diffusion to sample from the latent space and then use the decoder of the autoencoder to generate an image from the latent sample.

When talking about neural network architectures for $\epsilon_\theta(X_t, t)$, we enter the part of machine-learning that is closest to engineering. In practice there are no good ways to develop an architecture, instead trial and error guides the fields towards network architectures that work well in practice. Even the change of something as small as the non-linearity used can change performance drastically. Therefore, we recommend to use an architecture that works well in practice and apply it to the problem as is or with only small numeric adaptations. Of course the optimal architecture is problem dependent. We need different architectures for image or time series data than for unstructured tabular data.

The first network architecture we will introduce is called TabDDPM [2]. While the full architecture is centered around arbitrary data, including categorical (discrete) data, we assume that our data points are given as $X_0 \in \mathbb{R}^d$ and will only describe this part of the architecture. The first thing we have to discuss is how we model time as an argument to the neural network. A Naive approach is to just append t as another entry to X_t and feed that into a neural network. This works poorly in practice, since it is difficult for the neural network to learn that t is a global parameter that affects how each other feature is interpreted. Instead, the following scheme is used to transform the inputs X_t into a time dependent representation Y_t:

$$\text{Encoding}(x, t) = \text{Linear}(x) + \text{Linear}(\text{SiLU}(\text{TimeEmbedding}(t))) \tag{83}$$

Here, we refer to Linear as an affine linear function

$$\text{Linear}(x) = Wx + b$$

where $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ is a matrix and $b \in \mathbb{R}^{\times d_{\text{out}}}$ and $x \in \mathbb{R}^{\times d_{\text{in}}}$is a vector. We assume that each of these linear functions are their own layers with their own parameters (W, b) and d_{in} is chosen to fit the size of the argument. The TabDDPM architecture takes $d_{\text{out}} = 128$ but this is a tunable parameter of the architecture.

With this, $\text{Encoding}(x, t)$ is computed by taking a linear transformation of x and adding to that a linear transformation of some features that only depend on time. The linear functions allow that the time embedding features can be added linearly on each feature of the input space , and this gives the network the freedom to globally transform all inputs depending on the time parameter t. The features are generated the following way: Let d_{time} even be the number of time embedding features. We then have:

$$\text{TimeEmbedding}_i(t) = \begin{cases} \sin\left(2\pi \cdot 1000^{\left(\frac{i-1}{d_{\text{time}}-2}\right)} \cdot \frac{t}{T}\right), & \text{if } i \text{ is odd} \\ \cos\left(2\pi \cdot 1000^{\left(\frac{i-2}{d_{\text{time}}-2}\right)} \cdot \frac{t}{T}\right), & \text{otherwise} \end{cases}$$

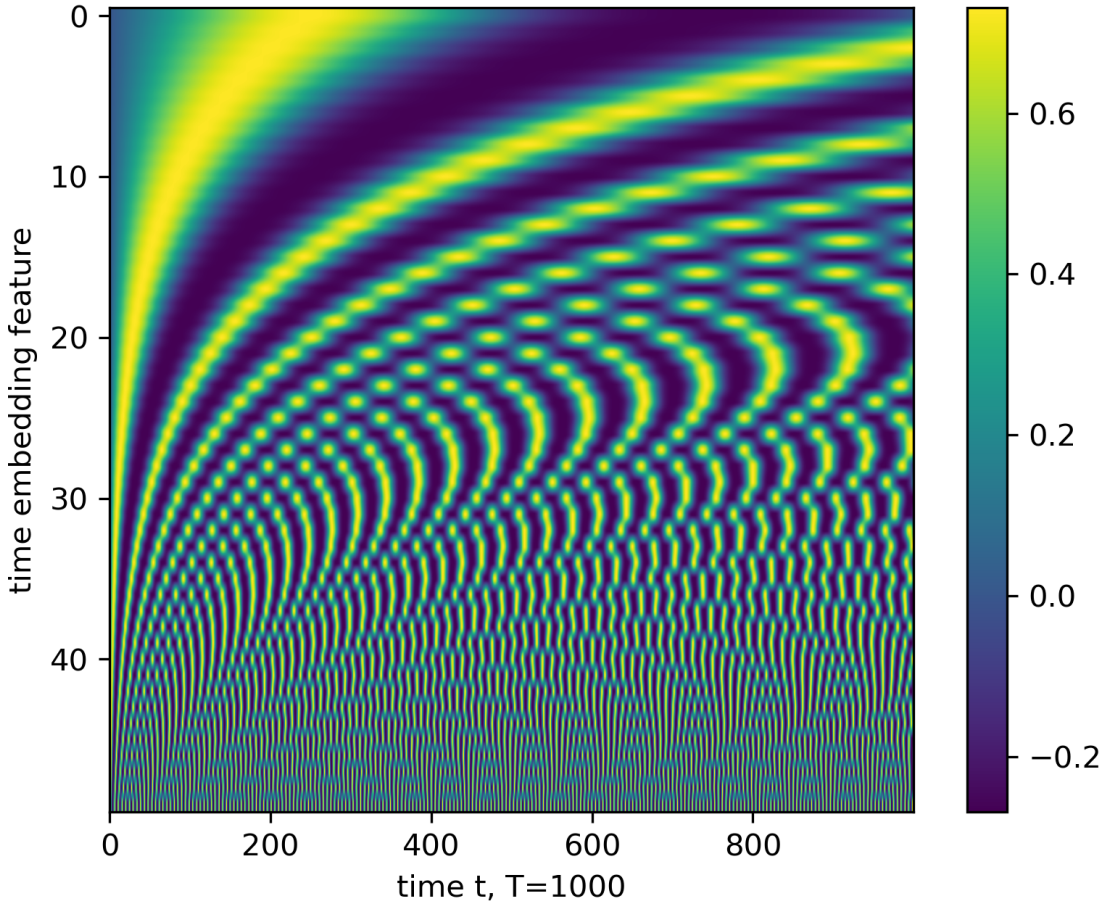


Figure 13: Visualisation of the time embedding features computed by SiLU(TimeEmbedding(t). Only the odd features are shown for $t = 0, 1, \ldots, 1000$. Each column describes one feature vector.

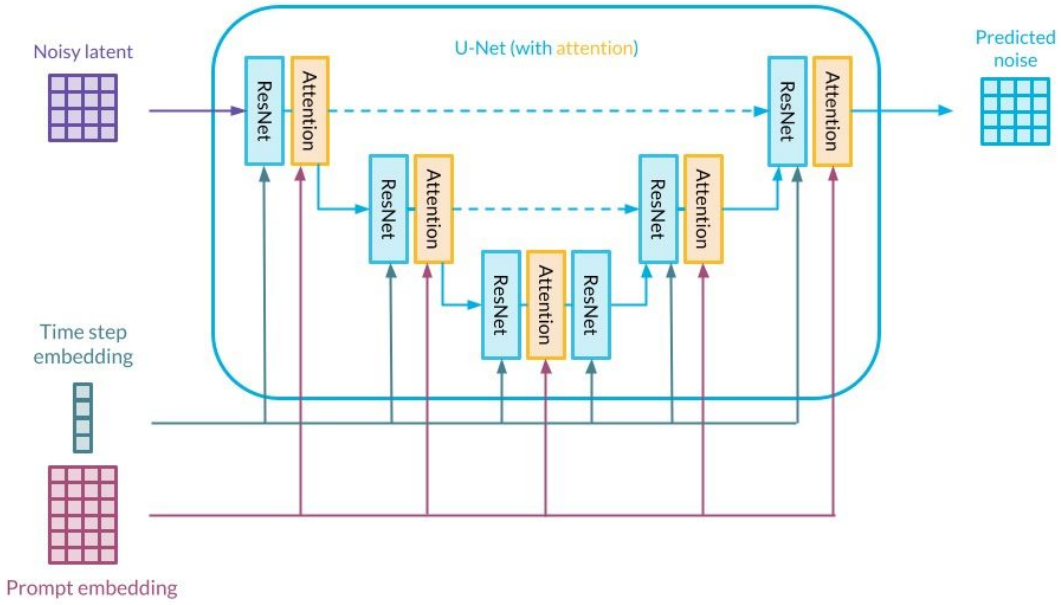


Figure 14: Visualisation of the attention-UNet architecture with time-and prompt-embeddings.

And the features are transformed by a nonlinearity

$$\text{SiLU}(x)_i = \frac{x_i}{1 + \exp -x_i} = x_i \cdot \text{sigmoid}(x) \ .$$

As a result, our time features are squished sinus and cosine functions that are paired to have the same frequency. In essence these create the Fourier basis, except that instead of choosing as frequencies the integers we pick a subset of frequencies that are geometrically spaced between 1 and 1000. The squishing using the SiLU nonlinearity just changes the basis such that time embedding values smaller than 0 are pushed towards 0, which breaks a symmetry in the embedding as only the positive parts of the base are used. See Figure 13 for a visualisation.

The remaining architecture is then a straight forward multilayer perceptron that uses the time transformed features as argument. We have

$$\begin{aligned} \text{MLPBlock}(x) &= \text{DropOut}(\text{ReLu}(\text{Linear}(x)) \\ \epsilon_\theta(X_t, t) &= \text{Linear}(\text{MLPBlock}(\text{MLPBlock}(\dots \text{Encoding}(x, t)))) \ , \end{aligned}$$

where Y_t is given by equation (83), $\text{ReLu}_i(x) = \max\{x, 0\}$, dropout is the dropout regularisation mechanism (can be skipped) and the number of MLPBlock layers is another free parameter.

For image data, the architectures are considerably more advanced due to the high level of specialisation and low-level engineering. A full description is beyond the scope of this course and we will only introduce the basic ideas. The basis of the architecture is a UNet that will predict $\epsilon_\theta(X_t, t)$. The UNet architecture can be seen in Figure 14. The UNet originates from segmentation tasks where the goal is to perform a pixel-wise classification, for example to find the pixels in an image that belong to a tumor. For this task we need on the one hand fine grained local information to accurately segment the boundary of the tumor, but on the other hand we also need a large context around each pixel in order to decide that the structure actually describes a tumor. The UNet provides both information at the same time. The idea can be described best by going right to left in Figure 14. At the top right side, the output is generated as a result of a convolution block. The convolution block takes two input images: first the result of a convolution block that takes the input image as argument and an image provided by a lower layer. The two images are concatenated and processed. The genius idea of the UNet is that this second image is computed based on a *downscaled* version of the input image. This downscaled image is computed by taken the output of the first convolution block on top left side and then downsample this result

by a factor of two, for example via a max pooling operation. This allows the decoder block on the top right side to use both the local information provided by the top left block and combine it with the larger context provided by the downscaled image. Now, we can repeat this idea for the downscaled block again and again until we have a broad enough context window to perform our task.

This idea works similarly for diffusion, since we need local information to obtain fine grained estimates of the noise, but we need global information to get a description of what is actually depicted on the image. Now the time embedding idea of before is added to each of the convolution blocks which are taken from the ResNet architecture. A common block structure is given by

$$\text{ResNet}(x, t) = x + \text{Conv}(\text{ReLu}(\text{Encoding}(x, t))) \ .$$

Here, we use the same time encoding idea as in (83), with the only change that we replace the Linear operations by a 2D convolution operation which we refer to as Conv. Then this is transformed by an MLP block which is given by first computing the nonlinearity and then computing a linear transformation given by a convolution. Finally, this result is added to the original input and returned. The idea of a ResNet block is that at the start of training, when the linear weights are small, it is almost an identity operation that becomes more complex as training progresses. This idea is known to improve training speed and quality of the obtained model. In many UNet architectures, there are multiple ResNet blocks on each level. The attention block is beyond the scope of this material, but it is often introduced in order to incorporate prompting information into the diffusion network. Thus, an output feature of the LLM can be used to inform the diffusion architecture in order to guide the diffusion process towards images that fit to a certain prompt. If this feature is not used, Attention blocks can be replaced by further ResNet blocks with only small losses in performance. But for more info on pompting and attention, see [4]

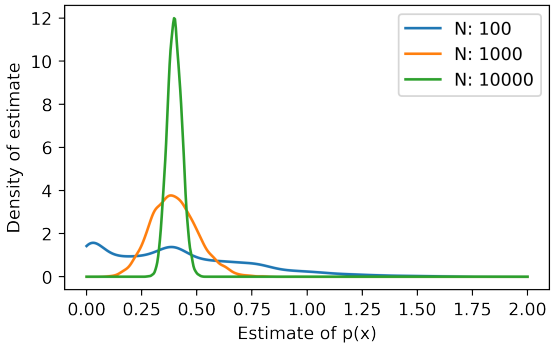


Figure 15: Distribution of empirical estimates of the VAE sample probability using the Naive Monte-Carlo sampler for different numbers of samples $N = 100, 1000, 10000$.

8 Model comparisons

After training of generative models, we need to validate their performance on unseen data. Unlike supervised learning, we often can not compute performance indicators like accuracy, because for unsupervised models these are difficult to define. Thus, for generative models, the log-likelihood of unseen data points is the only objective measure to compare models.

Unfortunately, many generative models (for example variational autoencoders and diffusion models) do not have an analytic log-likelihood. This makes it difficult to compare them. In this section, we introduce algorithms for estimating likelihoods. We do this by rephrasing the problem as estimating a normalization constant of a distribution and then develop algorithms for this, more general problem. As a running example, we will use the problem of estimating the marginal probability $p(x)$ of a variational autoencoder.

Example (Variational Autoencoder): In a VAE the probability of a sample is given by

$$p(x) = \int p(z)p(x|z)dz \ ,$$

where z are the latent variables and $p(x|z)$ is the learned generative model. To obtain a simple example, we assume that $z \in \mathbb{R}$ follows a standard normal distribution and

$$p(x|z) = \mathcal{N}(x; z, 10^{-4})$$

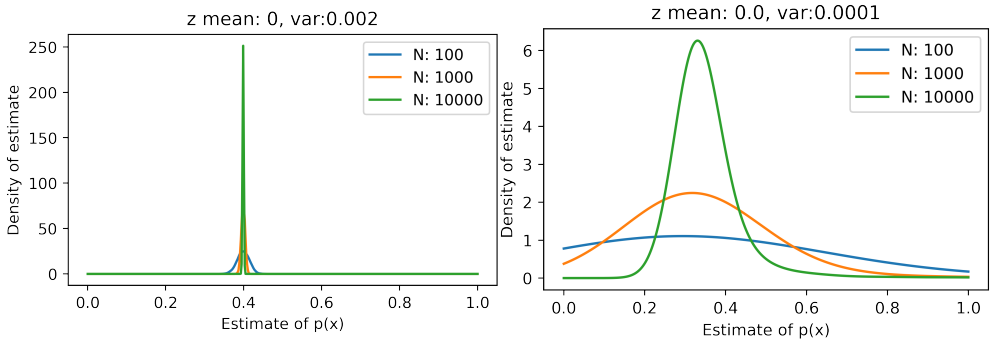
In this case, the analytic solution can be computed as $p(x) = \mathcal{N}(x; 0, 1 + 10^{-4})$. A Monte-Carlo estimate of this distribution however, works poorly. Using samples $z^{(i)}$, $i = 1, \dots, N$ drawn from $p(z)$, the sample estimate is

$$p(x) \approx \hat{Z} = \frac{1}{N} \sum_{i=1}^{N} p(x|z^{(i)})$$

The resulting distribution of values obtained from the estimator for different sample sizes is given in Figure 15. Even though this problem is a simple 1D problem, $N = 100$ samples are not enough to estimate the true value ($p(x = 0) \approx 0.39$) and the distribution often contains values around 0. Only with $N = 10000$ samples, the estimate becomes reasonable.

This example demonstrates a key issue of the naive Monte-Carlo estimator. Most samples drawn will lead to a value of $p(x|z) \approx 0$, unless the sample drawn has $x \approx z$, in which case the term $p(x|z)$ will become large (if $x = z$, we have $p(x|z) \approx 16$).

We will next introduce several algorithms that become gradually more complicated while building up on the previous algorithms. Afterwards, we will showcase an application to the task of model selection.



(a) $q(x|z) \approx p(x|z)$

(b) $q(x|z) \not\approx p(x|z)$

Figure 16: Visualisation of the distributions of the VAE estimation problem of estimating $p(x)$ for $x = 0$ with importance sampling using different choices of $q(z)$. In both cases $q(z)$ is normal with mean and parameters given in the title.

8.1 Estimators of Normalization Constants

Before we continue, let us rephrase the problem in more general terms. We will separate our distribution of interest into a known unnormalized probability and its normalization constant

$$p(x) = \frac{1}{Z}\tilde{p}(x) \tag{84}$$

Here, Z is the normalisation constant that normalizes $\int p(x)dx = 1$ and thus

$$Z = \int \tilde{p}(x)\ dx\ .$$

This normalization constant is often unknown, for example as a result of conditioning. We can use this to rephrase our VAE probability estimation problem as a problem of estimating normalization constants

Example: In our VAE example, we can write the conditional $p(z|x)$ as

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}\ .$$

Here, $p(x)$ is the unknown normalization constant of $p(z|x)$ and we can write the distribution as

$$p(z|x) = \frac{1}{Z}\tilde{p}(z|x),\ Z = p(x)\ .$$

We will now turn to the problem of estimating the normalization constant of a generic probability distribution for which we can only compute its unnormalized probability $\tilde{p}$.

Importance Sampling estimator. The easiest estimator can be obtained if we have access to another distribution $q(z)$ that approximates $p(x)$ and for which the normalization constant is known. In this case, we can define the importance-sampling estimator based on the identity

$$Z = \int_{\Omega_X} \tilde{p}(x)\ dx = \int_{\Omega_X} q(x)\frac{\tilde{p}(x)}{q(x)}\ dx\ . \tag{85}$$

Using a set of samples $x^{(1)}, \ldots, x^{(N)}$ from $q(x)$ we can estimate Z via the Monte-Carlo estimator

$$Z \approx \hat{Z} = \frac{1}{N}\sum_{i=1}^{N}\frac{\tilde{p}(x^{(i)})}{q(x^{(i)})}\ . \tag{86}$$

When $q(x) = p(x)$, this estimator has variance zero as in that case for all x

$$\frac{\tilde{p}(x)}{q(x)} = \frac{Zp(x)}{p(x)} = Z$$

and thus, each sample drawn from $q(x)$ results in a perfect estimate of Z. However, this estimato performs poorly, if we have the wrong guess about $q(x)$. We visualize the differences as part of our ongoing VAE example:

Example: In the VAE example from before, we can use the proposal distribution $q(z|x)$ for an importance sampling estimate. Given N samples $z^{(i)}$ drawn according to $q(z|x)$, we obtain

$$p(x) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{p(x|z^{(i)})p(z^{(i)})}{q(z^{(i)}|x)}$$

When $q(z|x) = p(z|x)$ this leads to the best possible estimator as

$$\frac{p(x|z)p(z)}{q(z|x)} = \frac{p(x,z)}{p(z|x)} = \frac{p(z|x)p(x)}{p(z|x)} = p(x) \ .$$

However, if this does not hold, the estimate can be rather poor. We show two examples of this in Figure 16. In the first case the variance of $q(z|y)$ is a mere factor two larger than the variance of $p(z|x)$ and predicts the correct mean. In the second case, the variance is a factor 10 lower than the optimal estimate and the estimator performs worse than the naive Monte-Carlo estimator, Figure 15.

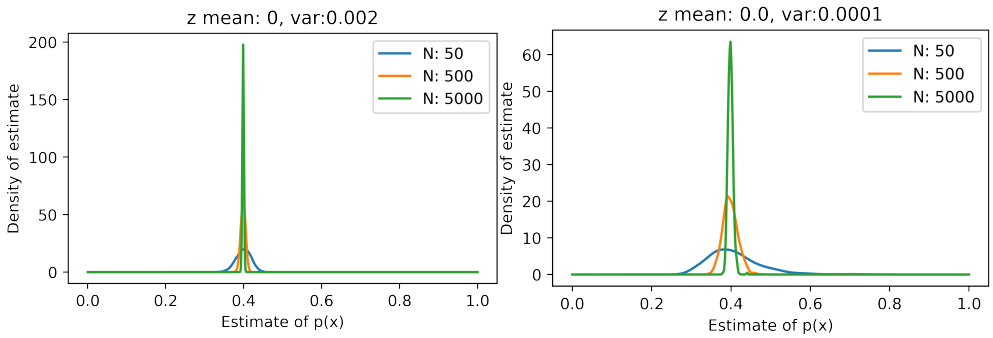
Indeed, the importance-sampling estimate can be so bad, that the variance of $\hat{Z}$ can become infinite. e show this in an example where we set $\tilde{p}(x) = p(x)$, that is, $Z = 1$. Then, the second moment of a sample is:

$$\int_{\Omega_X} q(x) \left(\frac{\tilde{p}(x)}{q(x)}\right)^2 dx = \int_{\Omega_X} q(x) \left(\frac{p(x)}{q(x)}\right)^2 dx = \int_{\Omega_X} \frac{p(x)^2}{q(x)} dz \ .$$

When we choose $p(x) = \mathcal{N}(x; 0, 1)$ and $q(x) = \mathcal{N}(x; 0, 1/10)$, the integral will diverge. However, even though the variance is infinite, in many cases the estimator will still *appear* to work fine. This is because the integral of the importance sampling estimate is dominated by low-probability events that result in large quotients $p(x)/q(x)$. A result of this is that even repeated estimates $\hat{Z}$ might a) show consistent results, while being completely wrong and b) show low-variance, even though it is infinite. An example for this can be seen when setting $p(x) = \mathcal{N}(x; 0, 1)$ and $q(x) = \mathcal{N}(x; 10, 0.1)$. In this case, the estimates will be close to zero (while $Z = 1$ is the correct result) and millions of samples are required to detect that there is a problem. It is important to note at this point that no amount of samples can fix this issue as the variance stays infinite, no matter how large the sample size is.

Bridge Sampling estimator. These results are not very encouraging. While we have seen that for a good choice of q, importance sampling works perfectly, for a bad choice, it can fail undetectably. this is a problem in practice, because often we do not have a good choice for q, or in the case of the VAE, we do not know how well $q(z|x)$ approximates $p(z|x)$ for an unseen sample. We will try to fix this problem in two ways, and we first start by introducing a new estimator. In general importance sampling, we have for arbitrary $f : \Omega_X \to \mathbb{R}$:

$$\begin{aligned} \int p(x)f(x)dx &= \int_{\Omega_X} q(x)\frac{p(x)}{q(x)}f(x)\,dx \\ &= \frac{1}{Z}\int_{\Omega_X} q(x)\frac{\tilde{p}(x)}{q(x)}f(x)\,dx \end{aligned}$$



(a) $q(x|z) \approx p(x|z)$ (b) $q(x|z) \not\approx p(x|z)$

Figure 17: Visualisation of the distributions of the VAE estimation problem of estimating $p(x)$ for $x = 0$ with importance sampling using different choices of $q(z)$. In both cases $q(z)$ is normal with mean and parameters given in the title.

We can now isolate Z and assuming $f(x) > 0$ to prevent division by zero, we obtain:

$$Z = \frac{\int_{\Omega_X} q(x) \frac{\tilde{p}(x)}{q(x)} f(x)\, dx}{\int_{\Omega_X} p(x) f(x)\, dx} \quad . \tag{87}$$

If we pick $f(x) = 1$, we arrive at our importance sampling identity (85). However we can now investigate other $f(x) > 0$ and once we have that, we can find Monte-Carlo estimators for both quotients. In general, this requires us to sample from both distributions, $p(x)$ and $q(x)$: sample N samples $x^{(1)}, \ldots, x^{(N)}$ from $q(x)$ and $x'^{(1)}, \ldots x'^{(N)}$ from $p(x)$ and then we have the estimate:

$$Z = \frac{\frac{1}{N} \sum_{i=1}^{N} \frac{\tilde{p}(x^{(i)})}{q(x^{(i)})} f(x^{(i)})}{\frac{1}{N} \sum_{j=1}^{N} f(x'^{(j)})} \quad .$$

While this estimator is not unbiased, it will asymptotically converge to the true value, provided that both quotients have finite variance. We can now try to pick $f(x)$ such, that the variance of both quotients stays finite. An approach is to pick

$$f(x) = \sqrt{\frac{q(x)}{\tilde{p}(x)}}$$

Which leads to the Monte-Carlo estimator

$$\hat{Z} = \frac{\frac{1}{N} \sum_{i=1}^{N} \sqrt{\frac{\tilde{p}(x^{(i)})}{q(x^{(i)})}}}{\frac{1}{N} \sum_{j=1}^{N} \sqrt{\frac{q(x'^{(j)})}{\tilde{p}(x'^{(j)})}}} \quad . \tag{88}$$

Proof of finite variance: To proof that both estimates individually have finite variance, we just compute the second moment of nominator and denominator of (88) and note that the second moment is never smaller than the variance. In the following, we will use the identity $\tilde{p}(x) = Zp(x)$ which allows us to rewrite f as

$$f(x) = \sqrt{\frac{q(x)}{Zp(x)}} \quad .$$

For the nominator, we have the second moment of a sample:

$$\int_{\Omega_X} q(x) \left(\frac{\tilde{p}(x)}{q(x)} f(x) \right)^2 dx = \int_{\Omega_X} q(x) \left(\frac{Zp(x)}{q(x)} \sqrt{\frac{q(x)}{Zp(x)}} \right)^2 dx$$
$$= \int_{\Omega_X} q(x) \frac{Zp(x)}{q(x)} dx = Z$$

Similarly, for the denominator we can derive:

$$\int_{\Omega_X} q(x) f(x')^2 = \frac{1}{Z}$$

While both second moments can become large when Z or $1/Z$ is large, as $Z > 0$, they stay finite, and thus the variances will also stay finite.

This estimator is called bridge-sampling, as $p(x)f(x) \propto \sqrt{p(x)q(x)}$, which can be understood as the geometric average of p and q, bridging the gap between both distributions. We can now use bridge-sampling to improve our estimate in the VAE example:

Example: We apply the bridge sampling estimator to the VAE to estimate $p(x)$. Inserting everything into the definition, we arrive at:

$$p(x) \approx \frac{\sum_{i=1}^{N} \sqrt{\frac{p(x|z^{(i)})p(z^{(i)})}{q(z^{(i)}|x)}}}{\sum_{j=1}^{N} \sqrt{\frac{q(z'^{(j)}|x)}{p(x|z'^{(j)})p(z'^{(j)})}}} \tag{89}$$

Here, $z^{(i)}$ are sampled from $q(z)$ and $z'^{(j)}$ are sampled from $p(z|x)$. the results are given in Figure 17. We can see that the estimator now even performs well for the case where $q(z|x)$ is only a poor approximation of the true estimate. However, we had to pay the price that we now need to obtain samples from the true $p(z|x)$. Thus, in a real VAE example we would need to sample, for example using HMC - however, if we are confident that $q(z|x)$ is not too far off the interesting region, we can save time by proposing a starting value, sampled from $q(z|x)$.

Sometimes, we are in a situation, where we do not even know the normalization constant of $q(x)$ but only know the normalized probability $\tilde{q}(x) = Z_q q(x)$. In this case, we can still use bridge sampling or importance sampling, but the resulting quantity will be the fraction of the normalization constants

$$\frac{Z_p}{Z_q} = \frac{\int_{\Omega_X} q(x) \sqrt{\frac{\tilde{p}(x)}{\tilde{q}(x)}} dx}{\int_{\Omega_X} p(x) \sqrt{\frac{\tilde{q}(x)}{\tilde{p}(x)}} dx} \tag{90}$$

We will see in the following how this entity can be used.

Tempering As the difference between p and q grows large, even bridge-sampling will eventually fail. This is because when the two distributions are far apart, both will predominantly create samples in regions where the other distribution has probability close to zero, and thus the samples are not containing information about the shape of the other distribution anymore.

A solution is tempering, which considers a large sequence of distributions $p_0, p_1, \ldots, p_T$, where $p_0 = q$ and $p_T = p$ and

$$p(x) = \frac{1}{Z_p} \tilde{p}(x), \; q(x) = \frac{1}{Z_q} \tilde{q}(x) \; .$$

All intermediate distributions in the sequence are interpolations between p and q. This is done by adding a new parameter, called the inverse temperature β_t, $0 = \beta_0 < \beta_1 < \cdots < \beta_T = 1$. This

parameter performs a geometric interpolation and the resulting distribution reads

$$p_t(x) = \frac{1}{Z_t}\tilde{p}(x)^{\beta_t}\tilde{q}(x)^{1-\beta_t} = \frac{1}{Z_t}\tilde{p}_t(x) \ .$$

Ideally, we know one of $Z_0 = Z_p$ and $Z_T = Z_q$, but their interpolated values Z_t are most often unknown. Using this set of distributions allows us to divide the problem of estimating Z_p/Z_q into more manageable chunks:

$$\begin{aligned}\frac{Z_p}{Z_q} &= \frac{Z_T}{Z_0} \\ &= \frac{Z_T Z_1}{Z_1 Z_0} = \frac{Z_T}{Z_1}\frac{Z_1}{Z_0} \\ &= \frac{Z_T Z_2}{Z_2 Z_1}\frac{Z_1}{Z_0} = \frac{Z_T}{Z_2}\frac{Z_2}{Z_1}\frac{Z_1}{Z_0} \\ &= \cdots = \prod_{t=1}^{T}\underbrace{\frac{Z_t}{Z_{t-1}}}_{C_t}\end{aligned}$$

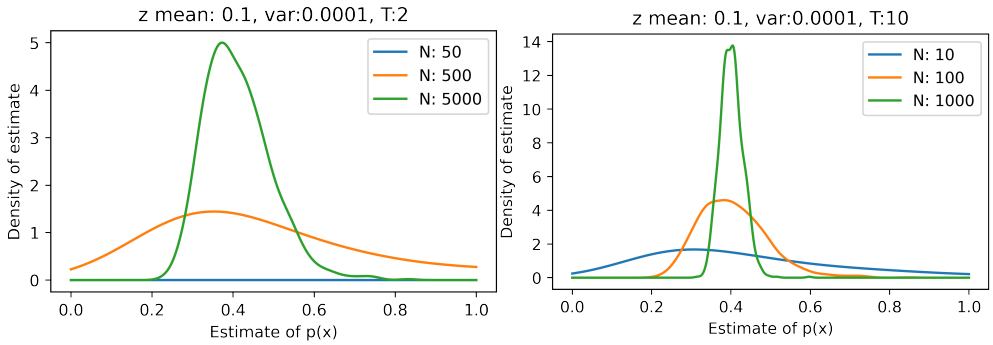
This suggests that we can solve our task of estimating Z_p/Z_q by estimating $C_t = Z_t/Z_{t-1}$ between each pair of distributions and then compute their products. Each factor C_t in turn can be estimated using bridge sampling as we did before and the estimation task is easier, because the interpolated distributions should be closer to each other. We can use entity (90) where we set $p = p_t$, $q = p_{t-1}$. We then obtain a Monte-Carlo estimate based on samples $x^{(i)}$ from p_{t-1} and $x'^{(j)}$ from p_t and arrive at:

$$\hat{C}_t = \frac{\frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{\tilde{p}_t(x^{(i)}))}{\tilde{p}_{t-1}(x^{(i)})}}}{\frac{1}{N}\sum_{j=1}^{N}\sqrt{\frac{\tilde{p}_{t-1}(x'^{(j)})}{\tilde{p}_t(x'^{(j)})}}} \ . \tag{91}$$

The algorithm for computing the estimate is:

1. Pick $q \approx p$ such, that $Z_q = 1$
2. Pick $0 = \beta_0 < \beta_1 < \cdots < \beta_T = 1$
3. Sample N samples $x_t^{(i)}$, $i = 1, \ldots N$ from p_t, $t = 0, \ldots, T$
4. Use (91) to obtain $\hat{C}_t$
5. Compute $Z_p = Z_p/Z_q \approx \prod_{t=1}^{T}\hat{C}_t$

Example : We now apply tempered bridge sampling to the VAE problem. We again choose a harder version of $q(z|x) = \mathcal{N}(0.1, 10^{-4})$. We investigate two cases. When $T = 2$, tempered bridge sampling is identical to bridge sampling. With $T = 10$, we pick β_t evenly spaced between 0 and 1. for better comparability, we choose N such, that $N \cdot T$ is one of 100,1000 or 10000 to have results comparable to previous runs.
The results are given in Figure 18. For $T = 2$ we see a case where bridge sampling seems to break down as the estimate with a small number of samples does not locate the true value at all. However, as the number of samples improves, the estimate also becomes better. For $T = 10$ we can see that all results are better than for $T = 2$. this shows that it is better to have more intermediate distributions.



(a) $T = 2$ (b) $T = 10$

Figure 18: Visualisation of the distribution of $\hat{C}$ of the VAE estimation problem of estimating $p(x)$ for $x = 0$ with bridge sampling using a difficult choice of $q(z)$ (normal with mean and variance given in the title.). The two cases differ in the number of distributions T used (2 or 10). The total number of samples used for the estimation is $T \cdot N$.

Annealed Importance Sampling Bridge sampling has very nice properties, but it requires us to run for each distribution a Markov-Chain that has a warm-up period long enough so that its samples approximate the distribution well. As we only have limited computation time to estimate all C_t, there is a trade-off between the size of T and the number of samples per chain. Moreover, these parameters require tuning. We have to investigate the estimate and find values that work, as too little intermediate temperatures will cause the method to break down. We will conclude now with a final method, called Annealed Importance Sampling (AIS), that does not require this trade-off and does not require warm-up for Markov-Chains. However, this comes at a price of in general poorer estimates.

The approach relies on properties of Markov Transition kernels. A Markov transition kernel $\mathbb{T}$ with stable distribution p is a conditional density for which holds that

$$p(x) = \int_{\Omega_X} p(x')\mathbb{T}(x \mid x')\, dx' \quad .$$

This property is the basis for Markov-Chain Monte Carlo, as repeated application of $\mathbb{T}$ will over time cover the full distribution. Here, we are interested in Markov transition kernels that fulfill a stronger property, called detailed balance, for which holds

$$p(x)\mathbb{T}(x' \mid x) = p(x')\mathbb{T}(x \mid x') \quad .$$

This property is fulfilled by all standard Markov-Kernels, for example Hamiltonian Monte-Carlo. We can show the first property easily using this, as

$$p(x) = \int_{\Omega_X} p(x)\mathbb{T}(x' \mid x)\, dx' = \int_{\Omega_X} p(x')\mathbb{T}(x \mid x')\, dx' \quad .$$

The first step holds because the Markov transition kernel is a normalized density and the second step holds due to application of detailed balance. Later on, we will use detailed balance in the following way using the unnormalized probability distribution $\tilde{p}(x)$

$$\frac{\mathbb{T}(x \mid x')}{\mathbb{T}(x' \mid x)} = \frac{p(x)}{p(x')} = \frac{\tilde{p}(x)}{\tilde{p}(x')} \quad . \tag{92}$$

With this necessary background in place, we will now introduce a set of tempered distributions, governed by inverse temperatures $0 = \beta_0 < \beta_1 < \ldots, \beta_T = 1$. We now require that we have a transition kernel for each temperature, $\mathbb{T}_t$, $t = 0, 1, \ldots, T$, where $\mathbb{T}_t$ fulfills the detailed balance condition for p_t. With this, we now sample a set of points the following way:

1. Sample x_0 from p_0

2. Sample x_t from $\mathbb{T}_t(x_t \mid x_{t-1})$ for $t = 1, \ldots, T$

The set of samples then follows the pdf

$$q_{\text{AIS}}(x_0, x_1, \ldots, x_T) = p_0(x_0) \prod_{t=1}^{T} \mathbb{T}_t(x_t \mid x_{t-1}) \ .$$

Intuitively, we take a sample x_{t-1} from the distribution with inverse temperature β_{t-1}, then pretend it is a sample of the distribution with inverse temperature β_t and use the Markov transition kernel of *that* distribution to generate a new sample x_t.

We now define a second distribution that follows the reverse process:

$$p_{\text{AIS}}(x_0, x_1, \ldots, x_T) = p_T(x_T) \prod_{t=1}^{T} \mathbb{T}_t(x_{t-1} \mid x_t) \ .$$

In this path, we now begin with a sample x_t, and then use the transition kernel $\mathbb{T}_t$ to generate the sample x_{t-1}. We can use these two distributions to build an importance sampling estimator of the normalization constant, via:

$$\frac{Z_T}{Z_0} = \int q_{\text{AIS}}(x_0, x_1, \ldots, x_T) \frac{\tilde{p}_T(x_T) \prod_{t=1}^{T} \mathbb{T}_t(x_{t-1} \mid x_t)}{\tilde{p}_0(x_0) \prod_{t=1}^{T} \mathbb{T}_t(x_t \mid x_{t-1})} \, dx_0, \ldots, dx_T$$

The importance weights can be simplified using the detailed balance property (92) and index manipulations

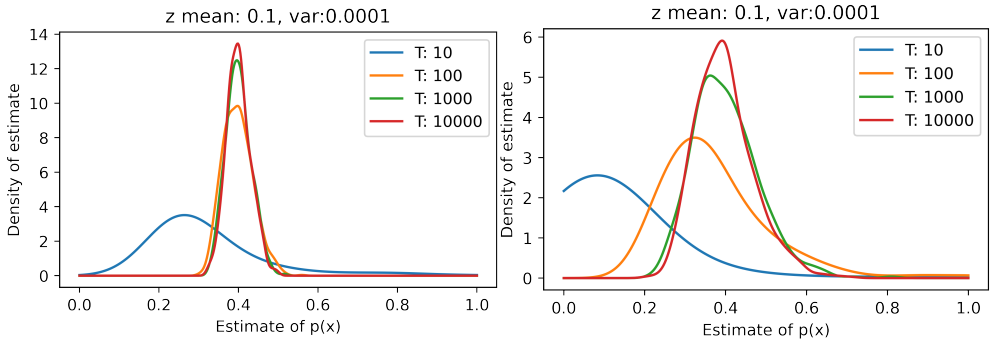
$$\begin{aligned}
&\frac{\tilde{p}_T(x_T) \prod_{t=1}^{T} \mathbb{T}_t(x_{t-1} \mid x_t)}{\tilde{p}_0(x_0) \prod_{t=1}^{T} \mathbb{T}_t(x_t \mid x_{t-1})} \\
=&\frac{\tilde{p}_T(x_T)}{\tilde{p}_0(x_0)} \prod_{t=1}^{T} \frac{\mathbb{T}_t(x_{t-1} \mid x_t)}{\mathbb{T}_t(x_t \mid x_{t-1})} \\
=&\frac{\tilde{p}_T(x_T)}{\tilde{p}_0(x_0)} \prod_{t=1}^{T} \frac{\tilde{p}_t(x_{t-1})}{\tilde{p}_t(x_t)} \\
=&\prod_{t=1}^{T} \frac{\tilde{p}_t(x_{t-1})}{\tilde{p}_{t-1}(x_{t-1})} = \prod_{t=0}^{T-1} \frac{\tilde{p}_{t+1}(x_t)}{\tilde{p}_t(x_t)} \ .
\end{aligned}$$

This final expression does not require us to compute the transition probabilities of the Markov transition kernel anymore, as we can just compute the fraction of unnormalized probabilities of the generated samples.

The final AIS procedure then performs the following steps:

1. Sample $x_0^{(i)}$ from p_0, $i = 1, \ldots, N$
2. Sample $x_{t+1}^{(i)}$ from $\mathbb{T}_{t+1}(x_{t+1}^{(i)} \mid x_t^{(i)})$ for $t = 0, \ldots, T-1$, $i = 1, \ldots, N$
3. Compute
$$\frac{Z_T}{Z_0} \approx \hat{C} = \frac{1}{N} \sum_{i=1}^{N} \prod_{t=0}^{T-1} \frac{\tilde{p}_{t+1}(x_t^{(i)})}{\tilde{p}_t(x_t^{(i)})} \ .$$

Why should this be a better estimator than naive importance sampling? Remember that the cause for large variance in naive importance sampling was when $p(x)/q(x)$ was large and the estimate dominated by rare events. In contrast the AIS estimator uses a product of factors $\frac{\tilde{p}_{t+1}(x_t^{(i)})}{\tilde{p}_t(x_t^{(i)})}$. When the number of intermediate distributions T becomes large, these individual factors each become closer to one. At the same time, each x_t has been sampled using the Markov transition kernel. Thus, if T is large enough, the sample from the previous distribution is also a likely sample from the next distribution and the sampling step allows to track the small differences between the distributions,



(a) Optimal Markov Kernel

(b) Arbitrary Markov Kernel

Figure 19: Visualisation of the distribution of $\hat{C}$ of the VAE estimation problem of estimating $p(x)$ for $x = 0$ with AIS using a difficult choice of $q(z)$ (same as Figure 18). The two case differ in the type of transition kernel used. The total number of samples used for the estimation is $T \cdot N = 10000$.

which makes it unlikely to end up in locations where the sample has low probability under one of the two distributions. Theoretically one can show that the variance of the AIS estimator shrinks with both, T and N. Indeed, if T is large enough, $N = 1$ is sufficient. However, the downside of AIS is that it requires that T is sufficiently large, otherwise it suffers from the same problems as importance sampling: the variance can become infinite.

Example : finally, we apply AIS to the VAE problem. We use the same distribution of $q(z|x) = \mathcal{N}(0.1, 10^{-4})$ as in tempered bridge sampling. This time we are only interested in the case of $N \cdot T = 10000$ total samples and see how varying T improves the estimate. Moroever, we now have to deal with the complexity of a Markov Transition kernel. We therefore look at two cases: one case, where we choose $T_t(x'|x) = p_t(x')$, which means that our kernel is able to generate perfectly uncorrelated samples. This is the optimal case. We compare this to a different case, where we choose a poorer transition kernel.
The results are given in Figure 19. For the optimal kernel, we see that the result with a small number of tempering steps $T = 10$ is still poor with a strong skew towards lower values. As T increases, the estimate improves. Interestingly, between $T = 1000$ and $T = 10000$ there is no notable difference, even though the $T = 10000$ only uses a single AIS sample to obtain the estimate. However, all these results are worse than tempered bridge sampling with $T = 10$ sampling steps.
For the non-optimal Markov transition kernel, the results are in general worse. The skew is stronger for small T and there are still visible differences between $T = 1000$ and $T = 10000$ and the final result is still worse than the optimal kernel at $T = 10000$. The results can be understood as now with the non-optimal kernel, the Markov Chain with small T has difficulties following the changes in the true distribution.

8.2 Application: Empirical Bayes

We will now use our new tool to perform prior-selection in a Bayesian modeling approach. This approach combines Bayesian modeling with frequentist model-selection, which gives it the name Empirical Bayes. This technique allows us to select the prior in our modeling based on the performance on unknown data.

That means, we are given a set of probabilistic models

$$p_i(\mathcal{D}, \theta) = \mathcal{L}(\mathcal{D}|\theta)p_i(\theta).$$

Here, $p_i(\theta)$ is a prior on some model parameters and $\mathcal{L}(\mathcal{D}|\theta) = \prod_{j=1}^{\ell} p(y_j|x_j, \theta)$ is the data log-

likelihood. a measure of model performance is the evidence

$$p_i(\mathcal{D}) = \int \mathcal{L}(\mathcal{D}|\theta)p_i(\theta)\, d\theta$$

In a pure Bayesian context, the notion of several priors does not make sense, as the prior is supposed to include all our knowledge before seeing any data. However, in many applications, there is also uncertainty about our prior knowledge: we might not know certain parameters, or whether assumptions that were used to derive our knowledge actually hold in practice. In this case, it makes sense to question our prior.

Assuming we have two models that differ only in their priors $p_1(\theta)$ and $p_2(\theta)$, we are interested in computing whether

$$\frac{p_2(\mathcal{D})}{p_1(\mathcal{D})} > 1$$

In this case, the we can argue that prior $p_2(\theta)$ models the data better than $p_1(\theta)$. We now turn this problem into a problem of estimating normalisation constants. We have

$$p_i(\theta|\mathcal{D}) = \frac{\mathcal{L}(\mathcal{D}|\theta)p_i(\theta)}{p_i(\mathcal{D})}$$

and thus the problem can be phrased as finding the quotient of normalisation constants of $p_1(\theta|\mathcal{D})$ and $p_2(\theta|\mathcal{D})$. We can turn this into a bridge-sampling estimation problem by arguing that p_1 takes the role of the proposal distribution q to estimate the normalization constant of $p = p_2$.

As a start we can use bridge sampling (90) with only the two distributions. Interestingly, the quotient of the unnormalized distribution simplifies:

$$\frac{p_1(\mathcal{D},\theta)}{p_2(\mathcal{D},\theta)} = \frac{\mathcal{L}(\mathcal{D}|\theta)p_1(\theta)}{\mathcal{L}(\mathcal{D}|\theta)p_1(\theta)} = \frac{p_1(\theta)}{p_2(\theta)}$$

And thus, the bridge sampling estimator reads

$$\frac{p_2(\mathcal{D})}{p_1(\mathcal{D})} \approx \frac{\frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{p_1(\theta^{(i)})}{p_2(\theta^{(i)})}}}{\frac{1}{N}\sum_{j=1}^{N}\sqrt{\frac{p_2(\theta'^{(j)})}{p_1(\theta'^{(j)})}}} \ ,$$

where $\theta^{(i)} \sim p_2(\theta|\mathcal{D})$ and $\theta'^{(j)} \sim p_1(\theta|\mathcal{D})$.

While this estimator is direct, it only works if we can assume that both posterior distributions are close to each other, in which case the choice of prior hardly matters with a moderatly sized dataset. We could improve on this estimator by using tempered bridge sampling, or AIS. We can also split the estimation tasks so that we estimate both evidences $p_i(\mathcal{D})$ independently. We will follow this path and use AIS for $p_i(\mathcal{D})$.

In absence of a good initial sampling distribution q, we interpolate between the prior and posterior:

$$\tilde{p}_t(\theta) = \tilde{p}_i(\theta|\mathcal{D})^{\beta_t}p_i(\theta)^{1-\beta_t} = \mathcal{L}(\mathcal{D}|\theta)^{\beta_t}p_i(\theta)$$

The AIS weights then simplify:

$$\prod_{t=0}^{T-1}\frac{\tilde{p}_{t+1}(x_t)}{\tilde{p}_t(x_t)} = \prod_{t=0}^{T-1}\frac{\mathcal{L}(\mathcal{D}|\theta_t)^{\beta_{t+1}}p_i(\theta_t)}{\mathcal{L}(\mathcal{D}|\theta_t)^{\beta_t}p_i(\theta_t)} = \prod_{t=0}^{T-1}\mathcal{L}(\mathcal{D}|\theta_t)^{\beta_{t+1}-\beta_t}$$

We can now pick a HMC kernel for each temperature and use AIS to obtain the estimate

$$p_i(\mathcal{D}) \approx \hat{C} = \frac{1}{N}\sum_{i=1}^{N}\prod_{t=0}^{T-1}\mathcal{L}(\mathcal{D}|\theta_t^{(i)})^{\beta_{t+1}-\beta_t} \ .$$

How can we improve on the AIS estimator, or in general assure that our model evaluation works and is efficient as possible? The first we can do, and the aspect that has the most impact, is a better choice of initial/proposal distribution. We can not expect to obtain cheap and fast

estimates when we begin with a poor guess of the distribution. An approach is to use variational approximations of the distributions e.g., using SVI in pyro). Even if this is available, the estimate might take too long. In our earlier VAE example, to estimate $p(\mathcal{D})$ we would need to estimate $p(x^{(i)})$ for each point in the test set independently. AIS can help here a little: if we are fairly sure that a moderate T leads to a decent estimate (e.g., by investigating a few samples), we can set $N = 1$ and use the sample average over the datapoints. This way, we can use that errors will also cancel out as we underestimate $p(x^{(i)})$ for some points while overestimating it for others.

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[2] Akim Kotelnikov et al. "Tabddpm: Modelling tabular data with diffusion models". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 17564–17579.

[3] Alexander Quinn Nichol and Prafulla Dhariwal. "Improved denoising diffusion probabilistic models". In: *International conference on machine learning*. PMLR. 2021, pp. 8162–8171.

[4] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.

[5] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising diffusion implicit models". In: *arXiv preprint arXiv:2010.02502* (2020).