MARC DE KAMPS

MACHINE LEARNING - UNIT 2 (v1.0)

UNIVERSITY OF LEEDS

Contents

List of Figures

List of Tables

1 The Central Limit Theorem and the Ubiquity of the Gaussian distribution

1.1 MLE of Gaussian Parameters

Assume that we have good reason to believe that a sample is generated from a Gaussian distribution with unknown parameters μ, σ. The likelihood for a single data point x_1 is given by:

$$p(x_1 \mid \mu, \sigma) = \mathcal{N}(x_1 \mid \mu, \sigma^2)$$

Again, if we assume that samples are generated *idd* (independently identically distributed), the likelihood for an entire dataset $\mathcal{D} = \{x_1, \cdots, x_N\}$ is:

$$P(\mathcal{D} \mid \mu, \sigma^2) = (\frac{1}{\sqrt{2\pi\sigma^2}})^N e^{-\frac{1}{2\sigma^2}(x_1-\mu)^2} \cdots e^{-\frac{1}{2\sigma^2}(x_N-\mu)^2}$$

This is a product of exponentials. Taking the logarithm reduces this to a relatively simple sum:

$$\ln(\mathcal{D} \mid \mu, \sigma^2) = -\frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i - \mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln 2\pi \tag{1.1}$$

Maximising Eq 1.1 with respect to μ gives:

$$\mu_{ML} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1.2}$$

Maximising with respect to σ gives

$$\sigma^2_{ML} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_{ML})^2 \tag{1.3}$$

Given a set of N data points, which are random variables, μ_{ML} and σ_{ML} themselves are random variables. One can show that:

$$\mathbb{E}[\mu_{ML}] = \mu \tag{1.4}$$
$$\mathbb{E}[\sigma^2_{ML}] = (\frac{N-1}{N})\sigma^2$$

1.1.1 *The Central Limit Theorem*

It is remarkable that many quantities in nature follow an (approximate) Gaussian distribution. Height distribution often are bell shaped and can be modeled accurately with Gaussian distribution. Scores from IQ tests often are modelled by Gaussian distribution and if the mean and variance are known - by fitting a Gaussian to a histogram of a large number of such scores - it is possible to determine whether this score is exceptionally high or low, and quantify that, e.g. by giving a percentage of the population that this score exceeds.

The ubiquity of the Gaussian distribution can at least be partly explained with the *Central Limit Theorem*. Let $S_N = \sum_{i=1}^{N} x_i$, be a sum of N *idd* variables of almost any distribution [1]. The Central Limit Theorem states that under mild conditions [2] as N increases, the distribution of S_N approaches [3]:

$$p(S_n = s) = \frac{1}{\sqrt{2\pi N\sigma^2}} e^{-\frac{(s-N\mu)^2}{2N\sigma^2}}, \tag{1.5}$$

where μ and σ^2 are the mean and variance of the distribution governing the random variables x_i. This implies that the quantity

$$Z_N \equiv \frac{S_N - N\mu}{\sigma\sqrt{N}} = \frac{\bar{X} - \mu}{\sigma/\sqrt{N}}$$

converges to a *standard normal* distribution, i.e. $\mathcal{N}(0,1)$.

Loosely speaking, if we sample sums of N random variables they follow a Gaussian distribution. The higher N, the more accurate the correspondence. Since averages are sums, they too follow a Gaussian.

In Activity *The Central Limit Theorem in Action* you will experiment with the application of this theorem to concrete datasets.

[1] The are a few exceptions, the Cauchy distribution, which has infinite variance is the best known

[2] C. Gardiner (2009). *Stochastic methods*, volume 4. Springer Berlin

[3] K. P. Murphy (2012). *Machine learning: a probabilistic perspective*. MIT press

1.1.2 *Z-score and Quantiles*

For large populations the mean and variance can often be estimated very accurately. A distribution of male heights in cm for example can be given as $\mathcal{N}(179, 10^2)$. If mean and variance are known we can estimate whether an individual is large compared to the rest of the population. The *cumulative distribution function* $F(x)$ can be given in terms of a probability density function by:

$$F(x) = \int_{-\infty}^{x} f(x)dx$$

The cumulative probability density (CPD) gives the probability $P(X \leq x)$. For the Gaussian, the cumulative probability is closely related to the error function, which is defined as:

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt, \tag{1.6}$$

which can easily be calculated in numpy. It is a straightforward exercise to relate this function to the CPD of the Gaussian distribution:

$$\Phi(x; \mu, \sigma) = \frac{1}{2} \left(1 + \text{erf} \left(\frac{x - \mu}{\sigma \sqrt{2}} \right) \right) \tag{1.7}$$

Without reference to μ and σ, the function Φ refers to the CDF of $\mathcal{N}(0, 1)$.

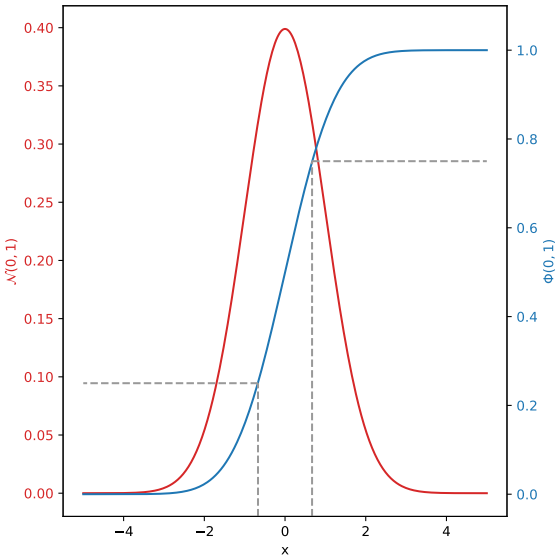

Figure 1.1: Standard Gaussian distribution (red). Its cumulative distribution function (blue). First and last quartile indicated by grey dashed lines.

For any CDF F that is monotonically increasing, there exists an inverse which is denoted by F^{-1}. $F^{-1}(\alpha)$ is called the α *quantile*. By definition, then, this is value x_α for which $P(X < x_\alpha) = \alpha$. The value of $F^{-1}(0.5)$ is called the *median* of the distribution. The values of $F^{-1}(0.25)$ and $F^{-1}(0.75)$ are called the lower and upper *quartiles*. They are shown in Fig 1.1.

The inverse CDF can be used to calculate *tail area probabilities*. Points to the left of $F^{-1}(\alpha/2)$ contain $\alpha/2$ of the probability mass. Points to the right of $F^{-1}(1 - \alpha/2)$ also contain $\alpha/2$ of the probability mass. The *central interval* $(F^{-1}(\alpha/2), F^{-1}(1 - \alpha/2)$ contains $1 - \alpha$ of the probability mass. For example, for $\alpha = 0.05$, the 95 % central interval is given by: $(\Phi^{-1}(0.025), \Phi^{-1}(0.975)) = (-1.96, 1.96)$.

When μ, σ are accurately known, the z-score for a data point x is given by $z = \frac{x-\mu}{\sigma}$. Using the z-score, a data point can be compared

to the $\mathcal{N}(0,1)$ interval. For example, if a data point falls outside the central interval as calculated above, it is atypically large or small. This is only possible when the parameters are accurately known rather than estimated. This is the case for very large populations, or comparing to ideal coins or dice and making predictions about the outcome using the central limit theorem.

As an example, consider the height of UK males, which is given as 174.5 cm for the 50-percentile, 186.0 cm for the 95-percentile and 164.1 cm for the 5-percentile [4] (cited from https://unece.org/fileadmin/DAM/trans/doc/2005/wp29grsp/HR-04-14e.pdf). Assuming that the height distribution is a Gaussian and that these numbers were measured on a large sample of the population we find that apparently $\mu = 174.5$ and since $\Phi^{-1}(0.95) = 1.65$, this gives a z-score which we can find σ by:

$$\sigma = \frac{x_{95} - \mu}{z_{95}} = \frac{186.9 - 174.5}{1.65} = 7.52\text{cm}$$

We now have estimates for μ and σ and should be able to make predictions about the 5-percentile, which we have not used in the calculation. $z_{05} = -1.65$ (as expected), giving $x_{05} = \mu + \sigma * z_{05} = 162.1$ cm which is in the right ball park, but shows a deviation from the reported value of 164.cm, suggesting heights are not distributed as a perfect Gaussian.

The z-score is useful when the the Gaussian parameters are known, or when a score can be compared to a hypothetically perfect distribution like the Bernoulli distribution of a perfect coin. The average of N observations will be Gaussian by the central limit theorem, so when we observe 100 coin tosses and calculate the z-score, we will be able to calculate the percentile function for this given outcome. We can then do a primitive form of hypothesis testing where the null hypothesis is that the coin is fair, and where we use an observation that has less than 5 % probability under the assumption that the coin is fair to reject this hypothesis.

When the parameters are not known, for example when the variance must be estimated from the data itself, the calculated $z-$score no longer follows a Gaussian distribution and other means like t-tests must be used. We will not discuss this further here. Standard statistic textbooks can be consulted, e.g. [5].

[4] D. of Trade and Industry (1998). *ADULTDATA - The handbook of adult anthropometric and strength measurements.*

[5] C. Gardiner (2009). *Stochastic methods*, volume 4. Springer Berlin

2 Multivariate Gaussian Distributions

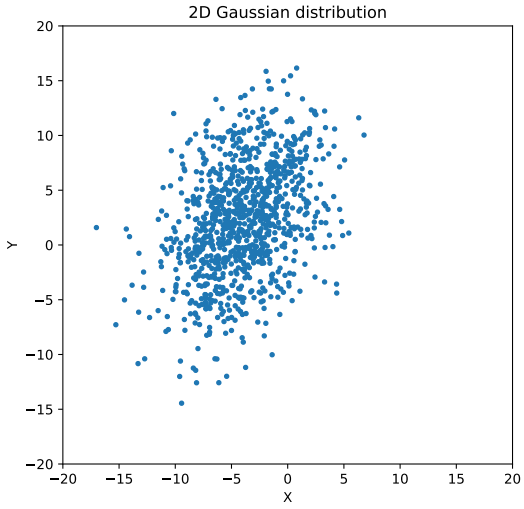

Figure 2.2: Sample of a two dimensional Gaussian distribution in two variable: X and Y. The variables are correlated and not centred at the origin.

2.1 Introduction

Multivariate means that it is a distribution in more than one variable. An example of samples drawn from a two-dimensional distribution are shown in Fig. 2.2. Before giving the formula for the multivariate Gaussian, a few remarks on notation.

A data point in multivariate distribution of dimension N is a tuple of N numbers, which is usually represented as a vector. In much of the machine learning literature, vectors are indicated by boldface lower case symbols. E.g. suppose we record IQ and height of individuals we have a data points where one dimension is used to record the height of the individual and a second one to record the IQ. We

denote the data point by:

$$\mathbf{x}_i = \begin{pmatrix} \text{iq}_i \\ \text{height}_i \end{pmatrix} \tag{2.1}$$

Some books use the vector notation $\vec{x}$, but the boldface notation is more prevalent.

Mathematical purists might object that these data points are tuples rather than vectors and they would have a point, but many of the operations in machine learning rely on so-called matrix-vector manipulations by numerical software and it is easier to stick to the conventions used in the machine learning literature.

The boldface notation indicates a column vector. Transposing it yields a row vector, so:

$$\mathbf{x}^T = (\text{iq}, \text{height}) \tag{2.2}$$

Matrix multiplication rules govern how expressions should be evaluated. If $\mathbf{v}$ and $\mathbf{w}$ are both N-dimensional vectors, then $\mathbf{v}^T\mathbf{w}$ represents the scalar product between these vectors, since it is the product of a row vector with a column vector, e.g.:

$$(v_1, v_2) \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = v_1 w_1 + v_2 w_2.$$

On the other hand, $\boldsymbol{v}\boldsymbol{w}^T$ represents a matrix whose components are $v^i w_j$ as can be seen from:

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \begin{pmatrix} w_1 & w_2 \end{pmatrix} = \begin{pmatrix} v_1 w_1 & v_1 w_2 \\ v_2 w_1 & v_2 w_2 \end{pmatrix}$$

Matrices are indicated by boldface upper case symbols. Where matrix and vector dimensions match matrix vector multiplications are implied by the order in which symbols occur. So, $\mathbf{M}\mathbf{v}$, represents a column vector:

$$\sum_{j=1}^{N} M_j^i v^j,$$

whereas $\mathbf{v}^T M \mathbf{w}$ represents a number:

$$\sum_{i,j} v_i M_j^i w^j$$

The boldface notation without component indices is usually more efficient and often unambiguous. Sometimes it is necessary to resort to components, in particular in proofs.

The multivariate Gaussian distribution is given by:

$$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{\mid \det(\boldsymbol{\Sigma}) \mid^{\frac{1}{2}}} \exp\left\{ -\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right\} \tag{2.3}$$

The functional dependence of the Gaussian on $\mathbf{x}$ is given by the *Mahalanobis distance*:

$$\Delta^2 = (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \tag{2.4}$$

Here, $\boldsymbol{\mu}$ is an M-dimensional vector, and $\boldsymbol{\Sigma}$ an $M \times M$ matrix. Without loss of generality, we can assume that $\boldsymbol{\Sigma}$ is a symmetric matrix.

Any matrix $\boldsymbol{M}$ can be written as the sum of a symmetric and an anti-symmetric matrix:

$$\boldsymbol{M}^i_j = \frac{1}{2}(\boldsymbol{M}^i_j + M^j_i) + \frac{1}{2}(\boldsymbol{M}^i_j - M^j_i)$$

You should verify that the anti-symmetric part does not contribution to the term under exponential, so we can always assume that $\boldsymbol{\Sigma}$ is symmetric.

2.1.1 *Spectral Decomposition of the Covariance Matrix*

The material in this section is relatively abstract. A Jupyter notebook titled: **Eigenvectors of the Covariance Matrix** *contains a worked example of the concepts discussed in this section.*

It is well known from linear algebra that a symmetric matrix can be diagonalised if a coordinate transformation is carried out from the original coordinate system wherein the data points are represented to an orthonormal basis comprised of the eignvectors of the matrix.

The eigenvector equation for the covariance matrix is:

$$\boldsymbol{\Sigma} \boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i,$$

where $i = 1, \cdots, D$.

In practice, in machine learning, we will find both eigenvectors and eigenvalues using numerical method. The notebook *Eigenvectors of the Covariance Matrix* gives examples and shows how the concepts discussed below apply to the Gaussian distribution. In general, any symmetric $D \times D$ matrix has D real eigenvalues. Moreover, the eigenvectors corresponding to these eigenvalues can be chosen to be orthonormal, i.e.

$$\boldsymbol{u}_i^T \boldsymbol{u}_j = \mathbb{1}, \tag{2.5}$$

where $\mathbb{1}$ is the D-dimensional identity matrix. The matrix $\boldsymbol{U}$ is a matrix where row i is given by $\boldsymbol{u}_i^T$

Now introduce:

$$\boldsymbol{y}_i = \boldsymbol{u}_i^T (\boldsymbol{x} - \boldsymbol{\mu})$$

The $\boldsymbol{y}_i$ are new coordinates which are translated and rotated with respect to the old ones. Forming the vector $\boldsymbol{y} = (\boldsymbol{y}_1, \cdots, \boldsymbol{y}_D$, one can write:

$$\boldsymbol{y} = \boldsymbol{U}(\boldsymbol{x} - \boldsymbol{\mu})$$

What this achieves is that, first the distribution is centred at the origin, and then a rotation is to the distribution so that the coordinate axes coincide with the eigenvectors (this is possible because the eigenvectors are orthonormal, so a rotation exists that aligns the eigenvectors with the coordinate axes; the rotation matrix is given by $\boldsymbol{U}$).

In the new coordinate system the distribution is the product of D independent Gaussian distributions:

$$p(\boldsymbol{y}) = \Pi_{j=1}^{D} \frac{1}{(2\pi\lambda_j)^{1/2}} \exp\left\{-\frac{y_j^2}{2\lambda_j}\right\} \tag{2.6}$$

If you are familiar with the process of diagonalising a matrix, this is what we have just done. We have diagonalised $\boldsymbol{\Sigma}$ to the diagonal matrix $\text{diag}(\lambda_1, \cdots, \lambda_D)$. For a diagonalised matrix the distribution factorises.

It is sometimes convenient to build a covariance matrix from a set of eigenvectors and eigenvalues. The spectral decomposition theorem states that covariance matrix can be expressed as an expansion in terms of its eigenvectors:

$$\boldsymbol{\Sigma} = \sum_{i=1}^{D} \lambda_i \mathbf{u}_i \mathbf{u}_i^{\mathrm{T}} \tag{2.7}$$

For the inverse:

$$\boldsymbol{\Sigma}^{-1} = \sum_{i=1}^{D} \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^{\mathrm{T}} \tag{2.8}$$

This has useful applications. In *Activity The Central Limit Theorem* you will learn how you can employ this equation to generate synthetic data sets.

2.1.2 *Maximum Likelihood Estimation*

We will derive the maximum likelihood estimators for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. It is more important that you are able to read the resulting formulae and can convert them into working code than that you can perform these derivations. We will not assess you on them. Having said that, if you want to be able to engage with the theoretical literature in the future you should be able to follow the derivations below and if not, you should acquire the matrix algebra underlying them. Appendix C of [1] provides a relatively comprehensive summary of the results. If you do not want to follow the details of the derivation, skip until Eqs. 2.21 and 2.26.

[1] C. M. Bishop (2006). *Pattern recognition and machine learning*. springer

Given a set of data points, that we suspect are Gaussian, how do we determine its parameters? Assume that we have a data set $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N)^T$. Note that this is a matrix with each individual

data point constituting one row of the matrix. For N data points, each of dimension D, the log likelihood function is given by:

$$\ln p(\boldsymbol{X} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{ND}{2}\ln(2\pi) - \frac{N}{2}\ln\det(\boldsymbol{\Sigma}) - \frac{1}{2}\sum_{n=1}^{N}(\boldsymbol{x}_n - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}) \tag{2.9}$$

In the following we will use the following results from matrix algebra: The *trace* of a matrix is the sum of its diagonal elements:

$$\mathrm{Tr}\boldsymbol{A} = \sum_{i=1}^{D} A_{ii} \tag{2.10}$$

For real symmetric matrices, remember that they can be diagonalised by means of a rotation:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T, \tag{2.11}$$

where $\Lambda = \mathrm{diag}(\lambda_1, \cdots, \lambda_D)$, the diagonal matrix with the eigenvalues of $\boldsymbol{A}$ at the diagonal entries.

It can be shown that this has as consequence that

$$\det(\boldsymbol{A}) = \Pi_{i=1}^{D}\lambda_i, \tag{2.12}$$

and

$$\mathrm{Tr}(\boldsymbol{A}) = \sum_{i=1}^{D}\lambda_i \tag{2.13}$$

Both these results depend on Eq. 2.11 so hold for real symmetric matrices, but not in general.

The following results from matrix algebra can be proven simply by writing them out in components:

$$\frac{\partial}{\partial \boldsymbol{a}}\boldsymbol{b}^T\boldsymbol{a} = \boldsymbol{b} \tag{2.14}$$

$$\frac{\partial}{\partial \boldsymbol{a}}(\boldsymbol{a}^T\boldsymbol{A}\boldsymbol{a}) = (\boldsymbol{A} + \boldsymbol{A}^T)\boldsymbol{a} \tag{2.15}$$

$$\frac{\partial}{\partial \boldsymbol{A}}\mathrm{Tr}(\boldsymbol{B}\boldsymbol{A}) = \boldsymbol{B}^T \tag{2.16}$$

$$\tag{2.17}$$

also easy to verify is:

$$\mathrm{Tr}(\boldsymbol{A}\boldsymbol{B}\boldsymbol{C}) = \mathrm{Tr}(\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}) = \mathrm{Tr}(\boldsymbol{B}\boldsymbol{C}\boldsymbol{A}) \tag{2.18}$$

With the last rule it is easy to justify the so-called *trace trick*:

$$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} = \mathrm{Tr}(\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}) = \mathrm{Tr}(\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{A}) = \mathrm{Tr}(\boldsymbol{A}\boldsymbol{x}\boldsymbol{x}^T) \tag{2.19}$$

If the first equality baffles you, remember that the first term is a row vector, left multiplying a matrix, left multiplying a column

vector. But a column row can also be interpreted as an $1 \times N$ matrix, and a column vector as an $N \times 1$ vector, so this a product of three matrices.

Based on the spectral decomposition Eq. 2.7, 2.8, and Eq. 2.12 as well as $\boldsymbol{u}_i^T \boldsymbol{u}_j = I$ one can show:

$$\frac{\partial}{\partial x} \ln \det(\boldsymbol{A}) = \mathrm{Tr}\left(\boldsymbol{A}^{-1} \frac{\partial \boldsymbol{A}}{\partial x}\right),$$

which specialises to:

$$\frac{\partial}{\partial \boldsymbol{A}} \ln \det(\boldsymbol{A}) = (\boldsymbol{A}^{-1})^T = \boldsymbol{A}^{-T}, \tag{2.20}$$

where the last equality introduces a convenient shorthand for the transpose of the inverse of a matrix.

To estimate μ_{ML} we set:

$$\frac{\partial}{\partial \boldsymbol{\mu}} \ln p(\boldsymbol{X} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \mid_{\mu=\mu_{ML}} = 0$$

Only terms that depend on $\boldsymbol{\mu}$ in Eq. 2.9 are relevant:

$$\frac{\partial}{\partial \boldsymbol{\mu}} \sum_{i=1}^{N} (\boldsymbol{x}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}) \mid_{\mu=\mu_{ML}} = 0$$

By virtue of identity 2.15 this is equivalent to:

$$-\sum_{i=1}^{N} (\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}^{-T})(\boldsymbol{x}_i - \boldsymbol{\mu}) \mid_{\mu=\mu_{ML}} = 0,$$

From this it follows that

$$-2\boldsymbol{\Sigma}^{-1} (\sum_{i=1}^{N} x_i - N\boldsymbol{\mu}) \mid_{\mu=\mu_{ML}} = 0$$

since $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^T$, and therefore:

$$\boldsymbol{\mu}_{ML} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i \tag{2.21}$$

The MLE of μ is the *empirical mean*.

To produce an MLE of the covariance matrix, it is convenient to introduce the *precision matrix*, which is the inverse of the covariance matrix:

$$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} \tag{2.22}$$

The only terms in the log likelihood that involve the covariance matrix are:

$$\mathcal{L} = -\frac{N}{2} \ln \det(\boldsymbol{\Sigma}) - \frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{x}_i - \mu)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu})$$

Expressed in terms of the precision matrix Λ this reads:

$$\mathcal{L}(\mathbf{\Lambda}) = \frac{N}{2}\ln\det(\mathbf{\Lambda}) - \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{x}_i - \mu)^T\mathbf{\Lambda}(\boldsymbol{x}_i - \boldsymbol{\mu}) \tag{2.23}$$

where we used $\det(\mathbf{\Lambda})\det(\mathbf{\Sigma}) = 1$, which follows from:

$$\det(\boldsymbol{AB}) = \det(\boldsymbol{A})\det(\boldsymbol{B})$$

Employing the trace trick on this expression gives:

$$\begin{aligned}\mathcal{L}(\mathbf{\Lambda}) &= \frac{N}{2}\ln\det(\mathbf{\Lambda}) - \frac{1}{2}\sum_{i=1}^{N}\mathrm{Tr}(\boldsymbol{x}_i - \boldsymbol{\mu})^T\mathbf{\Lambda}(\boldsymbol{x}_i - \boldsymbol{\mu}),\\ &= \frac{N}{2}\ln\det(\mathbf{\Lambda}) - \frac{1}{2}\mathrm{Tr}(\boldsymbol{S}_\mu\mathbf{\Lambda}),\end{aligned} \tag{2.24}$$

where the scatter matrix

$$\boldsymbol{S}_\mu \equiv \sum_{i=1}^{N}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T \tag{2.25}$$

The MLE for the precision matrix is very easy to derive in this form:

$$\frac{\partial\mathcal{L}(\mathbf{\Lambda})}{\partial\mathbf{\Lambda}}\,|_{\Lambda=\Lambda_{ML}} = \frac{N}{2}\mathbf{\Lambda}^{-T} - \frac{1}{2}\boldsymbol{S}_\mu^T = 0.$$

Using $\mathbf{\Lambda}^{-T} = \mathbf{\Lambda}^{-1} = \mathbf{\Sigma}$, we find:

$$\mathbf{\Sigma}_{ML} = \frac{1}{N}\boldsymbol{S}_\mu = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T \tag{2.26}$$

2.2 *Completing the Square - Marginal and Conditional Probabilities of the Gaussian*

In general, when given a high dimensional distribution it is difficult to marginalise with respect to given variables because this entails integration in high dimensional spaces unless one is fortunate enough to be able to integrate analytically. Normalising a distribution can be difficult for the same reason.

Gaussian distributions allow analytic integration and thereby marginalisation and normalisation. The technique occurs often enough in the machine learning literature to warrant an activity centred around this topic. A simple but important observation is that the log likelihood of a Gaussian is a quadratic form. Usually, it is sufficient to manipulate this quadratic form and restore the original distribution after the fact.

To see this, observe that:

$$-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T\mathbf{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}) = -\frac{1}{2}\boldsymbol{x}^T\mathbf{\Sigma}^{-1} + \boldsymbol{x}^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu} + \text{const} \tag{2.27}$$

When the log likelihood is brought into the form of the lefthand side, the full distribution is easily restored: it just requires exponentiating the quadratic form, and obtaining the overall normalisation factor which a function of the covariance matrix Σ only. When the log likelihood is in the form of the righthand side, it can be brought into the form of the lefthand side by completing the square.

In practice, this means that if the linear and quadratic terms of the log likelihood are known, the entire probability distribution can be reconstructed. Many mathematical operations involving Gaussians can be performed by manipulating quadratic forms. If this is not clear now, in Activity *Completing the Square: Manipulating the Gaussian log likelihood* , we will also show that if prior and likelihood are Gaussian, the posterior is Gaussian again, a fact that can be used immediately in Bayesian Linear Regression.

3 Bayesian Linear Regression

3.1 Introduction

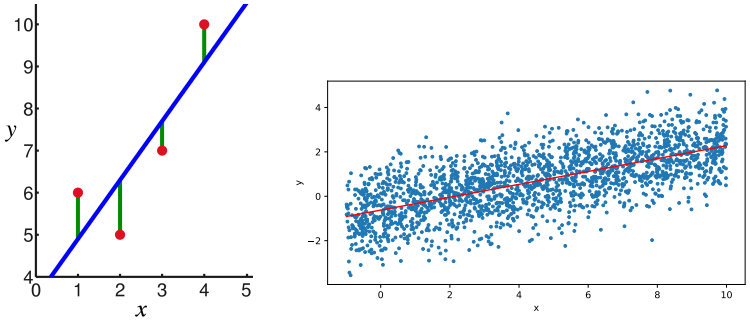

Figure 3.3: Left: observed points in red. A line fit to these points in blue. The residues in green (Wikimedia-ccby). Right: An example of linear regression.

The purpose of this section is to introduce the algebraic structure of linear regression, to contrast it in the next section with Bayesian linear regression. It is not necessary to be able to perform the derivations yourself, but you should be able to apply the resulting formulae. Examples of such applications are given in the notebook *Example 1.7: Linear Regression*.

The technique of linear regression is widely used in statistics and machine learning. You are expected to be familiar with it, because it was presented in the Data Science module. The essence of the method is shown in Fig. 3.3, where a linear functional relationship between two variables x and y is assumed, but where this relationship is imperfect, for example due to noise. It is clear that no linear relationship can fit the data perfectly, but on the other hand representation of the data by a linear function has a number of advantages. First, it requires much less information to store the parameters of the line than the entire data set. The linear function is called a *regression function* and the data is said to be regressed to a linear relationship. Second, the linear function may be used for *prediction*: when a new value of x is presented, the linear relation should present a reasonable prediction of y.

It is clear that some lines will be a good fit, and the line shown

in Fig. 3.3 is the best line in the following sense. It is clear that none of the data points are perfectly represented by the line. Consider a data point (x_i, y_i) and a model line $y = ax + b$ with a, b known parameters. For a given data point $r_i = y_i - (ax_i + b)$, the residue of point i represents a mismatch between the model prediction for value x_i, $ax_i + b$, and its actual value, y_i. Consider the sum of all residues squared: $R = \sum_i r_i^2$. For unknown a, b,R can be considered a function of a and b. OLS finds the values of a, b that minimise this function. Since R is essentially a quadratic function in a and b, we can be assured that a single global minimum exists. This should be expected, if we shuffle the line around in Fig. 3.3, it is clear that some sort of optimum line can be found.

Algebraically, conditions values for a and b can be found by minimising the function:

$$R = \sum_{i=1}^{N} (y_i - (ax_i + b))^2,$$

where N is the number of data points.

Differentiation with respect to a and b and setting the derivatives to o gives conditions for find the minimum:

$$\frac{\partial}{\partial a} R(a, b) = 0$$
$$\frac{\partial}{\partial b} R(a, b) = 0$$

This works out as a linear system of two equations with two unknowns:

$$\begin{aligned} \sum_i x_i y_i &= a \sum_i x_i^2 + b \sum_i x_i \\ \sum_i y_i &= a \sum_i x_i + bN \end{aligned} \tag{3.1}$$

In this particular case, formulae for a and b can easily be found by solving this system, which are the famous linear regression formulae.

A formal solution can be found by writing it in matrix-vector form:

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_i x_i^2 & \sum_i x_i \\ \sum_i x_i & N \end{pmatrix}^{-1} \begin{pmatrix} \sum_i x_i y_i \\ \sum_i y_i \end{pmatrix} \tag{3.2}$$

It is perfectly possible to include higher order terms, and for example fit a second degree polynomial to the data. A second degree polynomial has three parameters and would lead three equations with three unknowns that can solved directly. We will formulate this approach in way that allows easy generalisation to arbitrary degree.

Introduce the vector $\phi(x_i)$, defined as:

$$\phi(x_i) = \begin{pmatrix} 1 \\ x_i \\ x_i^2 \end{pmatrix} \tag{3.3}$$

and a vector $\boldsymbol{w}$:

$$\boldsymbol{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}, \tag{3.4}$$

so that:

$$\boldsymbol{w}^T \boldsymbol{\phi}(x) = w_0 + w_1 x + w_2 x^2$$

indeed represents a general second degree polynomial. This idea easily generates to a polynomial of arbitrary degree. If we are fitting an $M-1$ degree polynomial, we need M parameters that we organise in an M-dimensional vector $\boldsymbol{w}$.

The idea of minimising the residual squared sum is completely equivalent to that when regressing to a linear function:

$$R = \sum_{i=1}^{N} (y_i - \boldsymbol{w}^T \boldsymbol{\phi}(x_i))^2 \tag{3.5}$$

Again, we will find the value for $\boldsymbol{w}_{OLE}$ by setting:

$$\frac{\partial}{\partial \boldsymbol{w}} R \mid_{\boldsymbol{w}=\boldsymbol{w}_{OLE}} = 0$$

Calculating the gradient leads to:

$$\sum_{i=1}^{N} (t_i - \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i)) \boldsymbol{\phi}(\boldsymbol{x}_i)^T = 0,$$

or

$$\sum_{i=1}^{N} y_i \boldsymbol{\phi}^T(x_i) = \boldsymbol{w}^T (\sum_{i=1}^{N} \boldsymbol{\phi}(x_i) \boldsymbol{\phi}^T(x_i)) \tag{3.6}$$

For each data point we have a column vector $\boldsymbol{\phi}(x_n)$. The *design matrix* is an $N \times M$, matrix whose elements are given by $\Phi_{nj} = \phi_j(x_n)$. Here $M-1$ is the degree of the polynomial that we intend to fit to the data and N is the number of data points. For the case of fitting a second degree polynomial, $M = 3$.

$$\Phi = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_{M-1}(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_{M-1}(x_2) \\ \vdots & \vdots & & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_{M-1}(x_N) \end{pmatrix} \tag{3.7}$$

Using this matrix, we can write Eq. 3.6 as

$$\boldsymbol{\Phi}^T \boldsymbol{t} = \boldsymbol{\Phi}^T \boldsymbol{\Phi} \boldsymbol{w},$$

so that

$$w_{OLE} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\boldsymbol{t} \tag{3.8}$$

Some care must be used in applying these equations as they may be numerically unstable. This may happen if two basis functions evaluate to nearly the same vector. In that case an SVD decomposition must be used [1].

[1] C. M. Bishop (2006). *Pattern recognition and machine learning*. springer

Although Eq. 3.8 is a very compact notation, the result is not fundamentally different from the linear regression example, nor is the reasoning leading to it. It is instructive to formulate the case of a straight line, which is a polynomial of degree 1, so $M = 2$.

The column vector for data point x_i in this case is:

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

so the design matrix Φ is given by:

$$\Phi = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{pmatrix}$$

so that its transpose Φ^T is:

$$\Phi^T = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_N \end{pmatrix},$$

The product of the two is:

$$\Phi\Phi^T = \begin{pmatrix} N & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{pmatrix}, \tag{3.9}$$

Since $\boldsymbol{t}$ is:

$$\boldsymbol{t} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix},$$

the formula for finding the appropriate coefficients (weights) for our polynomial is:

$$\begin{pmatrix} w_0 \\ w_1 \end{pmatrix} = \begin{pmatrix} N & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{pmatrix}^{-1} \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{pmatrix} \tag{3.10}$$

Again, you will not be assessed on whether you can derive this result, but you are strongly encouraged to study notebook *Example 1.7: Linear Regression*, to see how the design matrix is built from the data and how Eq. 3.8 works out in practice.

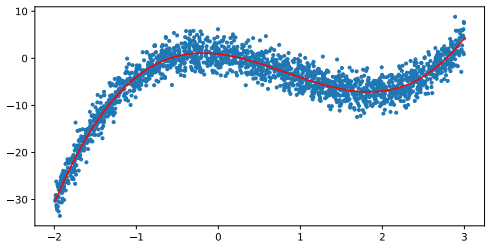

Figure 3.4: The result of linear regression on a cubic polynomial.

Figure 3.4 shows the fit of a cubic

It is important to realise that all examples of this section are linear regression, even if we fit a non linear function to the data. The distinction between linear and non linear regression is not whether or not to use of linear functions to model the data, but whether the fit function is a linear function its weights. In that case Eq. 3.8 applies. An example of non linear regression is *logistic regression*, to be discussed in Unit 2.

Also important is the realisation that polynomials are not the only function we can regress to. Any sufficiently rich set of *basis functions* can be used. In the case of fitting to a polynomial, the basis functions are the *monomials* $1, x, x^2, \cdots$. But Gaussians can also be used as basis functions. For example, if functions in a range $[a, b]$ need to be modeled, one can create a grid in μ, σ representing Gaussians of different means and variances. The notebook *Example 1.7: Linear Regression* gives a simple example of that.

So, the vector of functions $\boldsymbol{\phi}(x)$ can be chosen differently, but the process of constructing the design matrix and the regression procedure remain the same, given $\boldsymbol{\phi}(x)$, which are sometimes called *features*. For this reason linear regression is a powerful framework that is applicable to data with very non linear features.

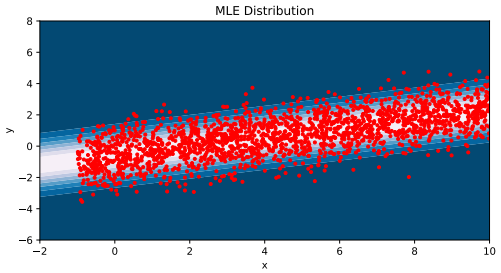

Figure 3.5: The MLE, a single point in (μ, σ) space, determines a Gaussian distribution centred around the line represented by the MLE.

3.2 *Maximum Likelihood and Least Squares*

In the ordinary least squares approach from the previous section no particular assumption was made about the origin of the residuals. Here we will see that it has a natural interpretation as the maximum likelihood estimation of a deterministic model with additive Gaussian noise.

Assume that we are observing data that has been generated by a process that can be described by

$$t = y(\boldsymbol{x}, \boldsymbol{w}) + \epsilon, \tag{3.11}$$

where ϵ is a Gaussian variable with zero mean and *precision* β (precision is the inverse of variance so $\beta = 1/\sigma$, its use is a matter of convenience, unburdening the notation of $^{-1}$s). In general, we will know the function y, but not the parameters $\boldsymbol{w}$, which we will have to infer from the observed data, just as in the case of linear regression.

Eq. 3.11 implies the existence of a probability distribution of target values, condition $\boldsymbol{x}$ and parameters $\boldsymbol{w}$ through the deterministic function y:

$$p(t \mid \boldsymbol{x}, \boldsymbol{w}, \beta) = \mathcal{N}(t \mid y(\boldsymbol{x}, \boldsymbol{w}), \beta^{-1}) \tag{3.12}$$

Consider the data set of inputs $\boldsymbol{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ with corresponding target values $t_1, \cdots t_N$, we group the targets values $\{t_i\}$ into a column vector $\boldsymbol{t}$. Assuming data points are independently drawn from distribution 3.12, it is possible to define a likelihood function with adjustable parameters $\boldsymbol{w}$ and β

$$p(\boldsymbol{t} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) = \Pi_{i=1}^{N} \mathcal{N}(t_i \mid \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i), \beta^{-1}), \tag{3.13}$$

where, as usual, $\mathcal{N}$ denotes the Gaussian distribution. The log likelihood then is:

$$\begin{aligned} \ln p(\boldsymbol{t} \mid \boldsymbol{w}, \beta) &= \sum_{i=1}^{N} \ln \mathcal{N}(t_i \mid \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i), \beta^{-1}) \\ &= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(\boldsymbol{w}) \end{aligned} \tag{3.14}$$

Here, the sum-of-squares error function is given by:

$$E_D(\boldsymbol{w}) = \frac{1}{2} \sum_{i=1}^{N} (t_i - \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i))^2 \tag{3.15}$$

But this is the same form as the sum of squared residues Eq. 3.5! So, the estimate for $\boldsymbol{w}$ that minimises the log likelihood and thereby maximises the likelihood, $\boldsymbol{w}_{MLE}$ is the same that minimises the sum of residues squared, and we find:

$$\boldsymbol{w}_{MLE} = (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{t} \tag{3.16}$$

It is important to note that no particular noise model underlies the OLE estimates, whereas the interpretation of $\boldsymbol{w}_{MLE}$ as maximum likelihood estimate explicitly determines on the Gaussian assumption made in Eq. 3.12. However, when the Gaussian model is appropriate it allows a more extensive analysis for example Bayesian linear regression.

3.3 *Bayesian Linear Regression*

Bayesian linear regression is a vast subject, and here we only give a bare bones introduction. The key idea is the same as in our example of the false coin: we assume that our parameters are uncertain and model this with a probability distribution. You are forced to assume a prior distribution, but as data becomes available, you use it to infer a posterior distribution *given* the data. This distribution will become more and more focused as more data becomes available. In the limit of infinite data, the distribution would become a sharp narrow peak that centres a value. In this limit the maximum likelihood would also converge on this value and both approaches would give the same result: a specific numerical prediction that contains no uncertainty. When not much data is available, the posterior distribution will be wider, representing the uncertainty in our fit parameters. MLE on the other hand will produce point estimate for them. Only with cross validation do we get a feeling for the uncertainty in this estimate.

The Bayesian linear regression case is an important illustration of the fact that: *Bayesian methods are less prone to overfitting than MLE estimates*. The concept of regularisation emerges as a natural consequence of the Bayesian approach, and Bayesian linear regression serves as a good demonstration.

As in the earlier example of the false coin where we had to assume a prior distribution for the parameter μ, we define a prior distribution for the weights $\boldsymbol{w}$:

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{S}_0) \tag{3.17}$$

The likelihood function is given by:

$$\mathcal{L} = \Pi_{i=1}^{N} \mathcal{N}(t_i \mid \boldsymbol{w}^T \boldsymbol{\phi}(x_i), \beta^{-1}), \tag{3.18}$$

which is the same we used in the case of the MLE.

Throughout this example, we will assume that β, the precision, is known. If it is not known, the approach can be extended to infer it as well, although there will be some technical difficulties that we do not want to address here. Here we will use the data to infer the set of weights $\boldsymbol{w}$ that will describe the data best.

In this particular case Bayes' rule can be written as:

$$p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{t}, \alpha, \beta) \sim p(\boldsymbol{t} \mid \boldsymbol{X}\boldsymbol{w}\beta)p(\boldsymbol{w} \mid \alpha) \tag{3.19}$$

It is a relatively straightforward exercise, which you will do yourself in a worksheet, to show that for a Gaussian prior the posterior will also be a Gaussian. In this exercise you will find the mean and covariance of the posterior distribution in terms of the data and the mean and covariance of the prior distribution.

Here, we briefly state the formulae for Bayesian Linear regression. Again we want to describe our data with a deterministic polynomial, whose coefficients we want to determine from the data. In line with the Bayesian approach, we define a prior probability distribution over our weights. Based on the assumption of Gaussian distributed noise 3.13, we assume a Gaussian prior:

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w} \mid 0, \alpha^{-1}\mathbb{1}),$$

that is: we centre our weights on zero and allow a single parameter to set the variance in each dimension. In the absence of prior knowledge about the data, this is a reasonable choice.

The posterior distribution is Gaussian again:

$$p(\boldsymbol{w} \mid \mathcal{D}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{m}_N, \boldsymbol{S}_N)$$

Here $\mathcal{D}$ is shorthand for the dataset $\boldsymbol{X}, \boldsymbol{t}$, where

$$\boldsymbol{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$$

and

$$\boldsymbol{t} = \{\boldsymbol{t}_1, \cdots, \boldsymbol{t}_N\},$$

where $\boldsymbol{t}_i$ is the observed value associated with $\boldsymbol{x}_i$, the value that our fit should reproduce as closely as possible.

We see that the posterior distribution is again a Gaussian, with a mean $\boldsymbol{m}_N, \boldsymbol{S}_N$ given by:

$$\boldsymbol{m}_N = \boldsymbol{S}_N(\boldsymbol{S}_0^{-1}\boldsymbol{m}_0 + \beta\boldsymbol{\Phi}^T\boldsymbol{t}) \tag{3.20}$$

$$\boldsymbol{S}_N = \boldsymbol{S}_0^{-1} + \beta\boldsymbol{\Phi}^T\boldsymbol{\Phi} \tag{3.21}$$

Here Φ is again the design matrix. $\boldsymbol{t}$ are the target values of the data set. α is a parameter which represents our prior. This value reflects the *subjective belief* about the weights prior to the data, so it must be chosen by us. If we chose it large, then this reflects a belief that the weights will be small. If we are not certain about this, we should pick a smaller value resulting in a broader prior.

We assume that β is known. If it is not, the likelihood function and prior become more complex than Gaussians, something we will for

now ignore, although estimating β from the data is certainly possible. We happen to know that $\beta = 1$, and will use that value here.

There are at least two possible ways of using these formulae.

1. *Batch learning*. Here we consider $\boldsymbol{m}_0, \boldsymbol{S}_0$ a prior, and the matrix Φ is constructed using the entire data set. A conventional choice is then:

$$p(\boldsymbol{w} \mid \alpha) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{0}, \alpha^{-1}\boldsymbol{I})$$

The formulae simplify somewhat in this case:

$$\boldsymbol{m}_N = \beta \boldsymbol{S}_N \boldsymbol{\Phi}^T \boldsymbol{t} \tag{3.22}$$

$$\boldsymbol{S}_N = \alpha \boldsymbol{I} + \beta \boldsymbol{\Phi}^T \boldsymbol{\Phi} \tag{3.23}$$

2. *Sequential learning*. We may again start with the following prior:

$$p(\boldsymbol{w} \mid \alpha) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{0}, \alpha^{-1}\boldsymbol{I}),$$

i.e. $\boldsymbol{m}_0 = \boldsymbol{0}$ and $\boldsymbol{S}_0 = \alpha^{-1}\boldsymbol{I}$.

and construct Φ from data that has just been acquired (which may be a single data point). We then calculate the N-th step according to:

$$\boldsymbol{m}_N = \boldsymbol{S}_N(\boldsymbol{S}_{N-1}^{-1}\boldsymbol{m}_{N-1} + \beta \boldsymbol{\Phi}^T \boldsymbol{t}) \tag{3.24}$$

$$\boldsymbol{S}_N = \boldsymbol{S}_{N-1}^{-1} + \beta \boldsymbol{\Phi}^T \boldsymbol{\Phi} \tag{3.25}$$

3.3.1 *Interpretation of Bayesian linear regression*

The MLE estimate itself is a point value. In weight space it corresponds to a single point. In data space, this is a distribution, since we model the data with a predictive distribution:

$$\mathcal{N}(t \mid \boldsymbol{w}^T \boldsymbol{\phi}(x), \beta^{-1}),$$

where we have assumed that the precision of the noise, $\beta = 1.0$. For each value of x, the MLE determines a distribution that peaks around the value $\boldsymbol{w}^T \boldsymbol{\phi}(x)$.

Let us recapitulate the MLE estimate for a linear relationship.

The values above give 'the best fitting line', both according to the criterion of a minimal sum of squared residues and the MLE of the likelihood. The predictive distribution is given by

$$\mathcal{N}(t \mid \ w_0 + w_1 x, 1.0),$$

We visualize this distribution as a heat plot, together with the original data.

3.3.2 *The Posterior Distribution*

The posterior distribution is not a single point like the MLE, but a distribution. We will show that the peak of this distribution usually is close to the MLE estimate. Some degree of numerical discrepancy is expected because the Bayesian maximum depends on our choice of prior. But the posterior distribution is not a single point in weight space: it is a distribution in weight space. As such it is a distribution of distributions. The differences between the MLE are more pronounced when not much data is available. We will first demonstrate the posterior obtained from a relatively small number of data points. As Fig. 3.6 shows, after 10 points, parameter w_0 has considerable uncertainty, whereas parameter w_1 has been much more constrained.

In order to visualise the influence of uncertainty in the posterior distribution, we can sample from it. The $\boldsymbol{w}$ values thus sampled each represent a linear relationship. Remember that each $\boldsymbol{w}$ represents an entire probability distribution. By plotting the linear relationships we represent each distribution by its peak value. We see that the gradients of the lines are better constrained than the intercepts, something that is also clear from the posterior distribution itself.

We have an example here of *model uncertainty*, an uncertainty in the model parameters. This comes on top of the noise that is intrinsic in the data, which is modelled by the covariance matrix, and which in the case of the MLE shown above was visible as a zone around the mean.

You are strongly encouraged to experiment with the notebook *Bayesian Linear Regression*.

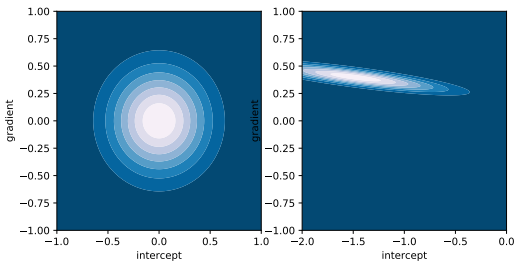

Figure 3.6: Prior and posterior weight distribution after applying Bayesian regression on ten points.

3.4 *Pros and cons of Bayesian Regression*

3.4.1 *Predictive Distribution and Maximum a Posteriori*

Using the MLE of the weights is simple. Once the optimal weights have been obtained, the are inserted in the polynomial. Using the

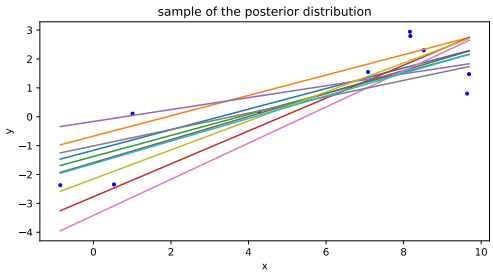

Figure 3.7: This plot shows 10 linear relationships that have been sampled from the posterior distribution, which was obtained by regressing on 10 data points.

regression results for prediction is as simple as inserting a new x value in the polynomial and obtain the corresponding y value, which is the prediction for this x.

To properly use the posterior distribution of weights, one would have to calculate the weighted expectation with respect to the posterior distribution values over all weights:

$$t = \int p(w \mid \boldsymbol{X}, \boldsymbol{t}) \boldsymbol{w}^T \boldsymbol{\phi}(x) dw$$

A simple estimate would probably be to generate a number of samples of the posterior distribution as we've done above, where we sampled 10 linear relationships. We can than average the predictions made by each of these lines as an estimated of the *predictive distribution*.

In many cases this is huge overkill. If the idea is to use regression to get an approximate prediction, it is not necessary to use the posterior distribution. Sometimes a good compromise can be to use its maximum. In this particular case this would be given by the value of $\boldsymbol{m}_N$. This is called the *maximum a posteriori* or *MAP*. Often a reasonable compromise, using the MAP is not entirely without risks, see for example section 5.2.1.2 of Murphy (2012).

In other cases, where little data is available and the penalty on a wrong decision based on MLE is severe it is better to accept the extra cost of the predictive distribution. In Unit 4 we will consider the case of determining whether a given point is an outlier. Such a decision may have financial or even legal consequences and here the importance of a principled estimate may outweigh the cost associated with evaluating the predictive distribution.

3.4.2 *Regularisation*

Bayesian models are far less prone to overfitting than MLE estimates, when applied consistently. The discussion for why this is the case becomes rather technical, but when two models explain the data adequately, the Bayesian approach favours the model with a smaller

number of parameters. This important aspect of Bayesian models is discussed in Section 3.4 and 3.5 of Bishop, but requires that you have digested most of the material earlier in Chapter 3 which goes into greater detail than we can do here.

We have seen that the Bayesian approach gives a posterior distribution of weights:

$$p(\boldsymbol{w} \mid \boldsymbol{t}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{m}_N, \boldsymbol{S}_N),$$

where you have derived the formulae for $\boldsymbol{m}_N, \boldsymbol{S}_N$. You should be able to verify that:

$$\ln p(\boldsymbol{w} \mid \boldsymbol{t}) = -\frac{\beta}{2} \sum_{i=1}^{N} \left\{ t_i - \boldsymbol{w}^T \boldsymbol{\phi}(x_i) \right\}^2 - \frac{\alpha}{2} \boldsymbol{w}^T \boldsymbol{w} + \text{const}$$

Maximising the likelihood is equivalent to maximising the log likelihood, which in turn is equivalent to minimising the quantity:

$$L = \sum_{i=1}^{N} \left\{ t_i - \boldsymbol{w}^T \boldsymbol{\phi}(x_i) \right\}^2 + \frac{\alpha}{2\beta} \boldsymbol{w}^T \boldsymbol{w}$$

The first term is a sum of squares, which also emerges in least squares. The second term effectively is a penalty term for large weights. This is called a *regularisation* term, as it puts constraints on the magnitude of the weights. A perfect fit which reduces the first term can still be spoilt if the weights to achieve it attain large values and in the earlier examples we saw this exactly what happens when we overfit a simple dataset.

L is sometimes called a *loss function* (for training purposes - not to be confused with a prediction loss, see the discussion in Section 1.5.5. of Bishop (2006)). Minimising it can be seen as an optimisation problem. In this unit we have achieved minimisation by analytic means, but in the following units we will often see loss functions that we have to minimise using numerical methods.

Modern neural network frameworks often allow the definition of models and an independent specification of loss functions. The *mean squared error* is a choice that is often made:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - t_i)^2,$$

and given the discussion so far it is not hard to see why it is an obvious choice, although it is by no means the only and often not the best one. Statisticians use a slightly different definition that incorporates the number of parameters.

Neural network frameworks also often the possibility for different forms of regularisation. The research into Bayesian neural networks is in its infancy (see [2] for a recent review), but regularisation offers some protection to overfitting.

[2] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun (2020). Hands-on bayesian neural networks–a tutorial for deep learning users. *arXiv preprint arXiv:2007.06823*

4 Information Theory and Probability

4.1 Introduction

In machine learning it is important to establish whether or not events are distributed as expected. If not, we cannot make accurate predictions about new events. We may have to adapt the distributions that we are comparing our events against, often by tweaking parameters, a process we call *learning*. The requires that we develop a quantitative measure to what extent past events conform to a given distribution. An interesting approach to this problem comes from information theory.

The key ideas are simple. Imagine that we have events that we measure at some remote station, which may fall in four categories: 'a', 'b', 'c' and 'd'. To transmit the occurances of each event we may use four bits: we can transmit the combinations '00', '01', '10', '11' and use a code book linking these messages to the four categories. Assuming that all occurrences are equally likely, this is the best we can do. If there were not four but sixteen categories, we would need 4 bits: 'oooo', etc. Binary words can use 2^N combinations using N bits. If all combinations are equally likely, it therefore makes sense to measure the information in terms of the number of bits required to transmit an event x:

$$h(x) = -^2 \log p(x).$$

Here $p(x)$ is the probability of the event which is the inverse of the number of possibilities, and therefore a minus sign is necessary to get a positive number of bits. When the probability for different categories is not uniform, it makes sense to adapt the coding scheme. In the example of 8 events $\{a, b, c, d, e, f, g, h\}$, with probabilities $(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64})$, it make sense to use a more efficient code [1] (as presented in [2]). If we were to ignore the difference in probabilities, we would have to assign 3 bits to each category to transmit an event. But if we use the following code strings 0, 10, 110, 1110, 111100, 111101, 111110, 111111, then on average we will transmit less bits.

This difference can be quantified using *entropy*: given a probability

[1] T. M. Cover (1999). *Elements of information theory*. John Wiley & Sons

[2] C. M. Bishop (2006). *Pattern recognition and machine learning*. springer

distribution $p(x)$ it is given by:

$$\mathbb{H}[x] = -\sum p(x)^2 \log p(x) \tag{4.1}$$

Note that this can be interpreted as the information averaged over all probabilities in the two examples we have given here. If we assign uniform probability to each of the 8 outcomes, we find:

$$\mathbb{H}[x] = -8 \times \frac{1}{8}^2 \log \frac{1}{8} = 3,$$

i.e. 3 bits.

If we calculate the average coding length under the probabilities listed above, using the coding scheme given above we find that it is:

$$\frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{16} \times 4 + 4 \times \frac{1}{64} \times 6 = 2,$$

so 2 bits. If we work out $\mathbb{H}[x]$ for that probability distribution, we also find 2 *by the same calculation*. This suggests an interpretation of $\mathbb{H}$ as average code length. The unequal distribution of probabilities allows a slightly more efficient transmission than the three bits per event that we found for equal probabilities. This idea is reminiscent of Morse code, which uses a short set of dashes and dots for letters that occur often (the 'e' is a '.'). You may wonder about why the particular code was chosen. This code allows the concatenation of events in longer strings, but allows an unambiguous resolution into individual event strings. You should check this.

The use of $^2\log$, logarithms with base 2 is relatively uncommon, and the natural logarithm is often preferred. If we write:

$$\mathbb{H}[x] = -\sum p(x) \ln p(x),$$

this amount to a change in units as changing the base of a logarithm multiplies the previous outcome by a constant factor. When we apply the natural logarithm, we no longer measure entropy in bits, but in nats.

Further credence to the interpretation of entropy as a measure of the average content carried by an event can be lent by the following observations:

- If only a single outcome has probably one, and all other potential possible outcomes have probability 0 then $\mathbb{H} = 0$. This requires an interpretation of $0 \ln 0$ as 0, which is justified since $\lim_{\epsilon \to 0} \epsilon \log \epsilon =$.
- The entropy for a given set of outcomes is maximal if all outcomes are equiprobable.

The analogy between entropy and the amount of information that can be transmitted holds only strictly for probability distributions which can be expressed as powers of $\frac{1}{2}$. For arbitrary distributions the entropy is a lower bound of the the information that can be transmitted using the best possible code. This is the essence of the *noiseless coding theorem* which is due to Shannon [3]. We will nonetheless stick to the code book metaphor as it provides good intuition for some of the measures that we will introduce below.

[3] C. M. Bishop (2006). *Pattern recognition and machine learning*. springer

Note that it is possible to express the amount of information associated with a certain event in terms of the probability of its outcome, again this $p_i \ln p_i$, where p_i is the probability of outcome i. This can be interpreted as the amount of nats required to store or transmit the event if an optimal code were to be used. The central role played by the logarithm in information theory is explained by the property of logarithms in any base:

$$\log(ab) = \log(a) + \log(b)$$

It is the only function with this property. It states that when possibilities multiply, information content adds. This should be clear from the examples above and you should now have an intuition for why logarithms play a central role in quantifying amounts of information.

The notion of entropy can be extended to continuous distributions but a subtlety occurs. Assume that the interval on which a continuous distribution is defined is split into a finite number of bins, each of width Δ. Assuming $p(x)$ is continuous, there exists an x_i such that

$$\int_{i\Delta}^{(i+1)\Delta} p(x)dx = p(x_i)dx$$

So, for a given discretisation the entropy can be written as:

$$\mathbb{H}_\Delta = -\sum_i p(x_i)\Delta \ln(p(x_i)\Delta) = \sum_i \Delta \ln p(x_i) - \ln \Delta$$

We omit the second term and then consider the limit $\Delta \to 0$. The first term on the righthand side then converges to

$$\lim_{\Delta \to 0} \sum_i p(x_i)\Delta \ln p(x_i) = -\int p(x) \ln p(x) dx$$

The integral is called *differential entropy*. It differs from the entropy by a term $\ln \Delta$ which diverges when Δ converges to 0. This reflects the fact that in a continuous distribution an infinite number of bits is required to register an event with total precision. In practice we are typically limited by machine precision and $\ln \Delta$ would be finite, but since it would be a constant contribution to every entropy formulation we are not particularly interested and simply omit in from our considerations.

For a multivariate density $p(\boldsymbol{x})$ the differential entropy is given by:

$$\mathbb{H}[\boldsymbol{x}] = \int p(\boldsymbol{x}) \ln p(\boldsymbol{x}) d\boldsymbol{x}$$

The following relationship sometimes appears in the literature, so we give it for completeness: Given a joint distribution $p(\boldsymbol{x}, \boldsymbol{y})$, assume that pairs of $\boldsymbol{x}, \boldsymbol{y}$ are drawn. Assume that the values of $\boldsymbol{x}$ is already known. The additional information needed to specify the corresponding value $\boldsymbol{y}$ is then $-\ln p(\boldsymbol{y} \mid \boldsymbol{x})$. The average information needed to specify $\boldsymbol{y}$ can be written as:

$$\mathbb{H}[\boldsymbol{y} \mid \boldsymbol{x}] = -\int \int p(\boldsymbol{y}, \boldsymbol{x}) \ln p(\boldsymbol{y} \mid \boldsymbol{x}) d\boldsymbol{y} d\boldsymbol{x}$$

using the product rule, this reads as:

$$\mathbb{H}[\boldsymbol{x}, \boldsymbol{y}] = \mathbb{H}[\boldsymbol{y} \mid \boldsymbol{x}] + \mathbb{H}[\boldsymbol{x}]$$

Here $\mathbb{H}[\boldsymbol{x}, \boldsymbol{y}]$ is the differential entropy of $p(\boldsymbol{x}, \boldsymbol{y})$ and $\mathbb{H}[\boldsymbol{x}]$ is the entropy of the marginal distribution $\mathbb{H}[\boldsymbol{x}]$. So the information required to specify $\boldsymbol{y}$ is the information required to specify $\boldsymbol{x}$ together with the extra information to specify $\boldsymbol{y}$ given $\boldsymbol{x}$.

4.2 *Relative Entropy*

Above, we introduced the notion of a code where the number of bits used to represent an event is coded is inversely proportional to the logarithm of the probability of the outcome of that event. That presupposes that we actually know this probability distribution. What if we have designed a code based on a probability distribution $p(x)$, when the actual distribution is a different one $q(x)$. We would be able to tell, because as we would collect events and store them on disk, we would need more information to record them than we would have expected.

Assume we want to transmit events based on a probability distribution $p(x)$. If we were to use a code book based on probability distribution $q(x)$, we will find that in general we will need to store a larger amount of information than if we had used a code book based on the true distribution $p(x)$. The amount of extra information is:

$$\begin{aligned} \mathrm{KL}(p \,||\, q) &= -\int p(x) \ln q(x) df - (-\int p(x) \ln p(x) dx) \\ &= -\int p(x) \ln \{\frac{q(x)}{p(x)} dx \end{aligned} \tag{4.2}$$

This is the *Kullback-Leibler* or *KL-divergence*, sometimes also called *relative entropy*. We will show that

$$\mathrm{KL}(p \,||\, q) \geq 0$$

and that only when $p(x) = q(x)$ for all x , $\mathrm{KL}(p \mid\mid q) = 0$. This underpins the statements made above that when using the 'wrong' code book, messages will be longer than they need to be.

All this implies that the KL-divergence can be used as a measure for how far two distributions diverge. It is not called a metric because it is asymmetric in its arguments, i.e. in general

$$\mathrm{KL}(p \mid\mid q) \neq \mathrm{KL}(q \mid\mid p).$$

As we will see, it derives its usefulness in part from this property.

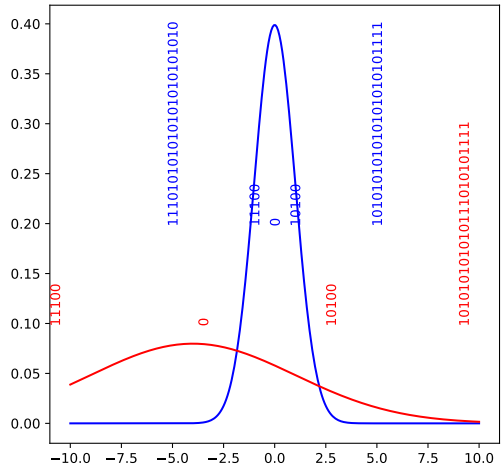

Figure 4.8: Events can be coded by a code book tailored to the red distribution or the blue distribution. When events are distributed according to the red distribution, they are more efficiently coded by the 'red' code book. Most events of the red distribution would fall well outside regular events according to the blue distribution, so using the 'blue' code book would lead to a huge increase in message size. The opposite is also true: events generated by the blue distribution are more efficiently code by the 'blue' code book, but events generated by the blue distribution could have plausibly come from the red distribution. This would lead to a longer message size if the 'red' code book were used although not nearly as much as in the opposite case. The KL-divergence is asymmetric in its arguments.

Consider Fig. 4.8. There is a blue and a red distribution. If events are distributed according to the blue distribution, their values are likely to be close to the peak of the blue distribution. A 'blue' code book would assign shorter bit sequences to those events, and longer ones to events well away from the peak because they are rare. Similarly for the red distribution. If we were to use the 'red' code book for events distributed according to the blue distribution, we would incur a penalty in terms of message length, because these events are somewhat away from the red centre and therefore would be longer sequences. Using the 'blue' code book for red events would be horrendous: most events of the red distribution cannot be plausibly generated by the red blue distribution. Only when the right code book is used for the right distribution of events is the message length minimal.

A real example in statistics would be to assume a Gaussian for events that are distributed according to a student distribution. The latter would be likely to generate outliers in a relatively small sample

that could not have been plausibly generated by a Gaussian. It is this property that makes the KL-divergence an efficient measure for divergence between two probability distributions.

Now suppose we sample data from an unknown distribution $p(x)$, which we may want to represent by a known distribution $q(|\ \theta)$, where θ are adjustable parameters. It would be nice to estimate the KL-divergence but the true distribution $p(x)$ is not available. If we have a finite set of data x_i, $i = 1 \cdots M$ drawn from distribution $p(x)$ we can approximate the KL-divergence by

$$\mathrm{KL}(p \,||\, q) \approx \sum_{i=1}^{N} \{-\ln q(x_n \mid \theta) + \ln p(x_n)\}\,.$$

The second term is independent of θ so minimising the KL-divergence with respect to θ is equivalent to maximising the likelihood function. In Unit 2, we will see how loss functions can be motivated based on the KL-divergence because of this.

4.3 *Jensen's Inequality*

The use of the KL-divergence as a measure that quantifies the inequality of two probability functions as a positive number rests on a simple mathematical theorem called *Jensen's inequality*. You will not be assessed on deriving it, but it occurs in many places in the machine learning literature. Alternatives exist to using the KL-divergence, e.g. *free energy*, but this too relies on Jensen's inequality. It is worth knowing about and being able to look up its proof when you encounter it in the literature.

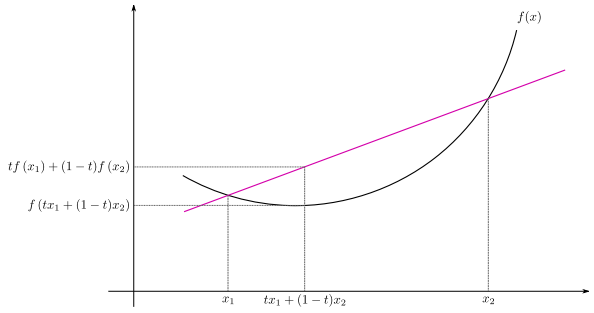

Figure 4.9: Jensen's inequality demonstrated. Source: Wikimedia.

A convex function is a function whose second derivative is positive everywhere. Examples are $f(x) = x^2$ and $g(x) = x \ln x$. An example is shown in Fig. 4.10. Consider a point of the purple line between x_1 and x_2. It is intuitively obvious that the entire purple line lies above $f(x)$ when $x_1 < x < x_2$. More precisely,

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

with strict inequality for $x_1 < t < x_2$ is $f(x)$ is strictly convex in that interval.

This is Jensen's inequality.

Using induction, this inequality can be proven for higher dimensions as well [4] (exercise 1.38) and then reads:

$$f(\sum_{i=1}^{M} \lambda_i x_i) \leq \sum_{i=1}^{M} \lambda_i f(x_i), \tag{4.3}$$

where $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$. By taking the values x_i as a discrete set, we can interpret the λ_i as a probability distribution over the discrete variables $\{x_i\}$. In that case Jensen's inequality can be written as:

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

There is a nice graphical interpretation of this inequality that can be found on Wikipedia: Consider Fig. 4.10. An arbitrary probability

[4] C. M. Bishop (2006). *Pattern recognition and machine learning*. springer

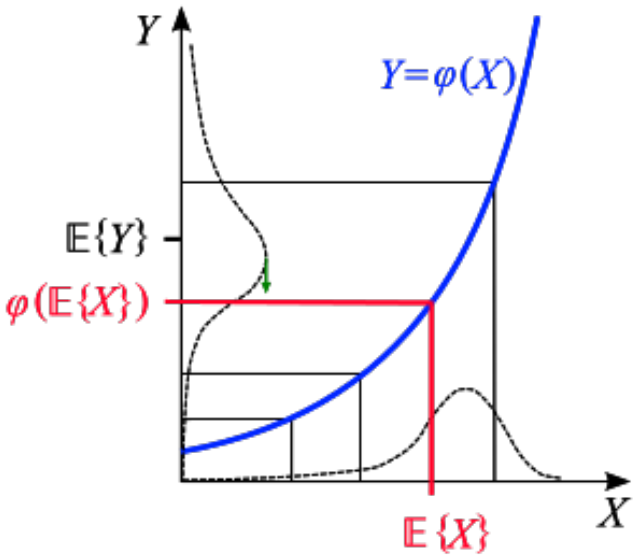

Figure 4.10: A graphical explanation for Jensen's inequality. Source: Wikipedia.

distribution can be found in the bottom of the plot with the density displayed as a function of X. The expectation value $\mathbb{E}[X]$ is shown in red and is slightly to the left of the bulge, due to the heavy tail extending to the left. A convex function $\phi(x)$ is the applied to all x values. The variables y are shown on the vertical axis. An interval in x, will be transformed into an interval in y and this interval in y must contain the same fraction of events as the interval in x. This leads to a transformation of the probability density over X into one over Y. It is clearly visible that the distribution is more stretched towards the higher the values in y. So you can look at the transformed value

of the expectation value, $\phi(\mathbb{E}[X])$ and compare it to the expectation value $\mathbb{E}[y]$, the expectation value of the y-distribution. The latter is always higher because the convex function $\phi(x)$ stretches larger values of x more than smaller values. This is the geometrical intuition underlying Jensen's inequality.

A final point that is both trivial and subtle. We can rewrite Jensen's inequality because the actual values of the x_i are immaterial. Suppose there is a function $y = g(x)$. Jensen's inequality only cares about the convexity of the function $f(x)$, not the value of the x_i. So written in terms of y_i, it is still true that:

$$f(\sum_{i=1}^{M} \lambda_i y_i) \leq \sum_{i=1}^{M} \lambda_i f(y_i),$$

In other words, Jensen's inequality can also be written as:

$$f(\sum_{i=1}^{M} \lambda_i g(x_i)) \leq \sum_{i=1}^{M} \lambda_i f(g(x_i)).$$

For continuous variables, this becomes:

$$f(\int \boldsymbol{x} p(g(\boldsymbol{x}))dx) \leq \int f(\boldsymbol{x}) p(g(\boldsymbol{x}))d\boldsymbol{x} \tag{4.4}$$

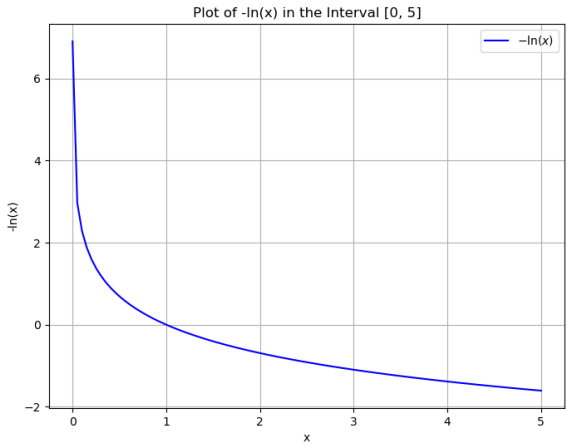

Figure 4.11: $-\ln x$ is a convex function.

Now we can apply this inequality to the KL-divergence. So far, we have discussed Jensen's inequality in terms of an abstract convex function. One such function is:

$$f(x) = -\ln x \tag{4.5}$$

Remember that this function plays a central role in this story due to its appearance in the entropy definitions. For completeness, we show

a plot of $f(x) = -\ln x$ in Fig 4.11. We can now use the definitions:

$$f(x) = -\ln x$$

and

$$g(x) = \frac{q(x)}{p(x)}$$

.

$$\mathrm{KL}(p \,||\, q) = -\int p(\boldsymbol{x}) \ln \left\{ \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})} \right\} d\boldsymbol{x} \geq -\ln \int q(\boldsymbol{x}) d\boldsymbol{x} \tag{4.6}$$

Here we used that $\int q(\boldsymbol{x}) d\boldsymbol{x} = 1$, as $q(\boldsymbol{x})$ is a probability distribution and the fact that $-\ln x$ is a strictly convex function, so that equality will only hold when $p(\boldsymbol{x}) = q(\boldsymbol{x})$.

This proof is important as it confirms what we so far based on intuition: using the 'wrong' probability distribution to define a code book *always* leads to longer messages. Moreover, the properties of the KL-divergence are not really dependent on our intuition any more. Although it is useful to think of entropy and KL-divergence in terms of information theory, Eq. 4.6 shows that the properties of the KL-divergence are based on Jensen's inequality and as such can be rigorously defined, independent of a coding length metaphor.

5 *Some Cautions*

The material presented here is intended to provide the theoretical framework for the later material. It is not suitable for the analysis of real world data without further reading. We have not discussed the need for more robust inference when data contains *outliers*. At the very least, you should consult section 7.4 of [1]. In Unit 5 you may find useful techniques to model outliers as a mixture of different processes.

From this unit you may have received the impression that Bayesian analysis is the norm. It is not. Its use, in particular with complex models such neural networks is in its infancy. The main problem is that a neat interpretation of the posterior in terms of the parameters of a distribution is possible in only the simplest of cases. Even in logistic regression (Unit 2), only a numerical estimate of the posterior is possible and this is a huge complication in its own right. Unit 4 and 5 deal with methods to alleviate this problem.

[1] K. P. Murphy (2012). *Machine learning: a probabilistic perspective*. MIT press

Bibliography

C. M. Bishop (2006). *Pattern recognition and machine learning*. springer.

T. M. Cover (1999). *Elements of information theory*. John Wiley & Sons.

C. Gardiner (2009). *Stochastic methods*, volume 4. Springer Berlin.

L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun (2020). Hands-on bayesian neural networks–a tutorial for deep learning users. *arXiv preprint arXiv:2007.06823*.

K. P. Murphy (2012). *Machine learning: a probabilistic perspective*. MIT press.

D. of Trade and Industry (1998). *ADULTDATA - The handbook of adult anthropometric and strength measurements.*