Applied Machine Learning Lecture notes Part I

Timo Koch

February 2024

These lecture notes are being updated from time to time and sections may be added or deleted upon request. We try our best but cannot guarantee that all content is indeed correct. In case you find errors in the script, please report them at `timo.koch@nmbu.no` so we can fix them.

This work is licensed under a Creative Commons "Attribution-ShareAlike 4.0 International" license.

Preface

These lecture notes provide some additional theoretical insights and mathematical intuition into some basic machine learning techniques.

Notation

We use boldface lowercase symbols to denote vectors ($\mathbf{x}$) and boldface uppercase symbols for matrices ($\mathbf{X}$).

In a machine learning context, when denoting features as matrices and labels as vectors, $\mathbf{X}$ denotes the feature matrix, $\mathbf{x}^{(i)}$ denotes a row vector with entries of row i of $\mathbf{X}$ representing a single sample (and values for all features), $\mathbf{x}_j$ denotes a column vector with entries of column j of $\mathbf{X}$ representing a single feature (and values for all samples), and $x_j^{(i)}$ denotes the value of the j-th feature of the i-th sample. The column vector with labels/outcomes is $\mathbf{y}$ and the label/outcome for the i-th sample $y^{(i)}$. In the case of supervised learning, we denote the data (or training set) as $\mathbf{D} = (\mathbf{X}, \mathbf{y})$.

If distinguishing between features and samples is not of importance in a given context, we also denote the entry in the i-th row and the j-th column of $\mathbf{X}$ as X_{ij} and the j-th entry of the vector $\mathbf{x}$ as x_j.

The notation $\mathbf{x} \in \mathbb{R}^d$ is used to specify that vector $\mathbf{x}$ has dimension d (has d elements) and the elements are from the set of real numbers, $\mathbb{R}$.

1 Simple linear and logistic regression

1.1 Simple linear regression and least squares

Linear regression is one of the simplest supervised learning algorithms. We consider the model

$$f(x) = \theta_0 + \theta_1 x \tag{1}$$

with slope θ_1 and intercept θ_0 which represents a straight line.

Let us assume we have a single feature per sample. Given a list of data samples $x^{(i)}$ and corresponding outcomes $y^{(i)}$, we can write the error between model and data for a single sample i as

$$\varepsilon^{(i)} = y^{(i)} - (\theta_0 + \theta_1 x^{(i)}). \tag{2}$$

How about multiple features? Let us introduce a notational trick. If we always add a constant feature $x_0^{(i)} = 1$ as the zeroth element to all samples, we can also rewrite this as

$$\varepsilon^{(i)} = y^{(i)} - \mathbf{x}^{(i)}\boldsymbol{\theta}, \tag{3}$$

with the column vector $\boldsymbol{\theta} = \begin{bmatrix}\theta_0 \\ \theta_1\end{bmatrix}$ and the row vector $\mathbf{x}^{(i)} = [1, x_1]$, where

$$\mathbf{x}^{(i)}\boldsymbol{\theta} := \sum_{j=0}^{1} x_j^{(i)}\theta_j = \sum_{j=0}^{m} x_j^{(i)}\theta_j \tag{4}$$

is an *inner product* where is the last equality we generalized the notation to m features. This will slightly simplify the notation in the following because we do not need to carry around the offset θ_0 and what we derive will work for any number of features.

In the generalized form, we consider m features and $(m+1)$ parameters. In matrix form with m features (or if we also count the constant, $(m+1)$ features) and n samples, we can express the error for all samples as an error column vector

$$\boldsymbol{\varepsilon} = \mathbf{y} - \mathbf{X}\boldsymbol{\theta}, \tag{5}$$

where

$$\boldsymbol{\varepsilon} = \begin{bmatrix}\varepsilon^{(1)} \\ \varepsilon^{(2)} \\ \vdots \\ \varepsilon^{(n)}\end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix}y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)}\end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix}\mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \vdots \\ \mathbf{x}^{(n)}\end{bmatrix} = \begin{bmatrix}1 & x_1^{(1)} & \cdots & x_m^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_m^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_m^{(n)}\end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix}\theta_0 \\ \theta_1 \\ \vdots \\ \theta_m\end{bmatrix}.$$

Our goal is to find the $\boldsymbol{\theta}$ such that the error $\boldsymbol{\varepsilon}$ is minimized. Note that $\boldsymbol{\varepsilon}$ contains an error for each sample but for convenience, we would like to have a single number that measures the error. A typical choice for this measure is

$$L = \boldsymbol{\epsilon}^T\boldsymbol{\epsilon} = ||\boldsymbol{\epsilon}||_2^2 = \sum_{i=1}^{n} {\varepsilon^{(i)}}^2, \tag{6}$$

where L is called the *loss function* (or sometimes *cost function*) and $(\bullet)^T$ denotes the transpose (i.e. it here turns a row into a column vector or vice versa) and $||\mathbf{x}||_2 := \sqrt{x_1^2 + \cdots + x_n^2}$ denotes the 2-norm (or Euclidean norm or ℓ_2-norm) of a vector. This specific loss function is also known as *quadratic loss function* or *squared error loss function*.

Inserting the expression for the error into the loss function, we obtain

$$\begin{aligned} L(\boldsymbol{\theta}) &= (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) \\ &= \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\boldsymbol{\theta} - \boldsymbol{\theta}^T\mathbf{X}^T\mathbf{y} + \boldsymbol{\theta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\theta}, \end{aligned}$$

where we used the property of the transpose that $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$.

At the minimum of a differentiable function, the function's derivative is zero. (It turns out that L is a strictly convex function and strictly convex functions have at most one minimum.) Therefore, we are looking for $\boldsymbol{\theta}$ such that

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \mathbf{0}. \tag{7}$$

Inserting the loss function expression yields

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{\theta}} &= \frac{\partial \left(\mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\boldsymbol{\theta} - \boldsymbol{\theta}^T\mathbf{X}^T\mathbf{y} + \boldsymbol{\theta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\theta}\right)}{\partial \boldsymbol{\theta}} \\ &= -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} = \mathbf{0}. \end{aligned}$$

Rearranging, results in

$$\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} = \mathbf{X}^T\mathbf{y}, \tag{8}$$

which is called the *normal equation*. The normal equation can be solved for $\boldsymbol{\theta}$,

$$\boxed{\boldsymbol{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}} \tag{9}$$

(where $\mathbf{A}^{-1}$ denotes the inverse of a matrix $\mathbf{A}$,) if the matrix $\mathbf{X}^T\mathbf{X}$ is invertible. It is invertible if we have at least as many samples as features, and the samples are linearly independent ("no redundant data").

Definition 1 (Normal matrix)**.** *The matrix* $\mathbf{N} := \mathbf{X}^T\mathbf{X}$ *is known as the* normal matrix*. Note that*

$$N_{ij} = \mathbf{x}_i^T\mathbf{x}_j = \sum_{k=1}^{n} x_i^{(k)} x_j^{(k)}, \tag{10}$$

that is the entries N_{ij} *of* $\mathbf{N}$ *are given by the inner products of feature vectors (containing the value of one feature for all samples).*

To get more intuition for the matrix notation, we can look at the loss function in the expanded form

$$L = \sum_{i=1}^{n} {\varepsilon^{(i)}}^2 = \sum_{i=1}^{n} \left(y^{(i)} - \sum_{k=0}^{m} x_k^{(i)}\theta_k \right)^2, \tag{11}$$

(naming the index of the inner sum κ to avoid confusion in the following,) and only consider the derivative with respect to a single parameter θ_j

$$\begin{aligned} \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_j} &= -2\sum_{i=1}^{n} \left(x_j^{(i)} \left[y^{(i)} - \sum_{k=0}^{m} x_k^{(i)}\theta_k \right] \right) \\ &= -2\sum_{i=1}^{n} x_j^{(i)} y^{(i)} + 2\sum_{i=1}^{n} \left(x_j^{(i)} \left[\sum_{k=0}^{m} x_k^{(i)}\theta_k \right] \right) = 0. \end{aligned} \tag{12}$$

This shows us that the full derivative (with respect to each parameter θ_j) is a column vector (with $(m+1)$ columns) and each row has to individually be zero. Now note that

$$\sum_{i=1}^{n} x_j^{(i)} y^{(i)} = \mathbf{x}_j^T\mathbf{y} \tag{13}$$

is an inner product of feature (column) vectors $\mathbf{x}_j$ and labels $\mathbf{y}$. For the second term, we have

$$\begin{aligned} \sum_{i=1}^{n} \left(x_j^{(i)} \left[\sum_{k=0}^{m} x_k^{(i)}\theta_k \right] \right) &= \sum_{i=1}^{n}\sum_{k=0}^{m} x_j^{(i)} x_k^{(i)}\theta_k = \sum_{k=0}^{m}\sum_{i=1}^{n} x_j^{(i)} x_k^{(i)}\theta_k \\ &= \sum_{k=0}^{m} \left(\left[\sum_{i=1}^{n} x_j^{(i)} x_k^{(i)} \right] \theta_k \right) = \sum_{k=0}^{m} N_{jk}\theta_k = [\mathbf{N}\boldsymbol{\theta}]_j, \end{aligned} \tag{14}$$

where we used that we can interchange sums when the inner summation index is independent of the outer summation index. Hence, we can write

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \theta_j} = -2\mathbf{x}_j^T\mathbf{y} + 2[\mathbf{N}\boldsymbol{\theta}]_j = 0. \tag{15}$$

Finally, if we simply write all j equations under each other in a column vector, we get

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{N}\boldsymbol{\theta} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} = \mathbf{0}, \tag{16}$$

which is exactly what we derived using matrix notation.

1.2 Probabilistic interpretation of linear regression

In the previous section, we chose to minimize the squared error loss function. Why did we choose the squared error? In this section, we look at a set of different starting assumptions from which the squared error loss function naturally arises. Again using the notational trick of adding a constant feature of 1 to all samples (see previous section), we choose a model

$$y^{(i)} = \mathbf{x}^{(i)}\boldsymbol{\theta} + \varepsilon^{(i)}, \tag{17}$$

where $\varepsilon^{(i)}$ is a noise term that models unconsidered effect or randomness in the model prediction (say the actual house price also depends on the mood of the seller and the weather on the sales day). We now assume

$$\varepsilon^{(i)} \sim \mathcal{N}(0, \sigma^2), \tag{18}$$

which means that the error follows a normal distribution (also called Gaussian distribution) with zero mean and variance σ^2. Hence, the probability density function $p(\varepsilon^{(i)}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{{\varepsilon^{(i)}}^2}{2\sigma^2}\right)$.

We will now assume that the $\varepsilon^{(i)}$ are i.i.d. (independent and identically distributed), meaning that the error for one sample is completely independent of the error of another sample. For a real system, this is an approximation because we cannot guarantee that there is no correlated effect that we are missing in the data. We can then state that

$$P(y^{(i)} \mid x^{(i)}; \boldsymbol{\theta}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\left(y^{(i)} - \mathbf{x}^{(i)}\boldsymbol{\theta}\right)^2}{2\sigma^2}\right), \tag{19}$$

that is the probability P of the outcome $y^{(i)}$ given the sample $x^{(i)}$ and parameterized by $\boldsymbol{\theta}$ ($\boldsymbol{\theta}$ is not a random variable, P is parameterized by given $\boldsymbol{\theta}$) is given by the expression on the right. In other words, the data is normally distributed with mean $\mathbf{x}^{(i)}\boldsymbol{\theta}$ and variance σ^2. With these assumptions, we define the likelihood function of the parameters $\boldsymbol{\theta}$ as

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) &= P(\mathbf{y} \mid \mathbf{X}; \boldsymbol{\theta}) \\
&= \prod_{i=1}^{n} P(y^{(i)} \mid x^{(i)}; \boldsymbol{\theta}), \qquad \text{(since } \varepsilon^{(i)} \text{ are i.i.d.)} \\
&= \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\left(y^{(i)} - \mathbf{x}^{(i)}\boldsymbol{\theta}\right)^2}{2\sigma^2}\right).
\end{aligned} \tag{20}$$

A common way in statistics to choose the best $\boldsymbol{\theta}$ is to use the $\boldsymbol{\theta}$ that maximizes the likelihood, which is called the *maximum likelihood estimator* (MLE). In other words, we choose the $\boldsymbol{\theta}$ to maximize the probability of the data ($P(\mathbf{y} \mid \mathbf{X}; \boldsymbol{\theta})$). In practice, it's often easier to work with the log-likelihood (the natural logarithm of the likelihood):

$$\begin{aligned}
\ell(\boldsymbol{\theta}) := \log \mathcal{L}(\boldsymbol{\theta}) &= \log \prod_{i=1}^{n} P(y^{(i)} \mid x^{(i)}; \boldsymbol{\theta}), \\
&= n \log \frac{1}{\sigma\sqrt{2\pi}} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y^{(i)} - \mathbf{x}^{(i)}\boldsymbol{\theta}\right)^2.
\end{aligned} \tag{21}$$

Since the natural logarithm is a strictly monotone function, the MLE $\boldsymbol{\theta}$ maximizes both $\ell(\boldsymbol{\theta})$ and $\mathcal{L}(\boldsymbol{\theta})$. Since the first term is a constant independent of $\boldsymbol{\theta}$, and maximizing a function is the same as minimizing the negative function, we can look for $\boldsymbol{\theta}$ that minimizes

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(y^{(i)} - \mathbf{x}^{(i)}\boldsymbol{\theta}\right)^2 = \sum_{i=1}^{n} {\varepsilon^{(i)}}^2, \tag{22}$$

where we left away the multiplier $(2\sigma^2)^{-1}$ which does not affect the minimizer but only scales the loss function.

This means that the maximum likelihood estimator (given that the errors are Gaussian and i.i.d.) *minimizes* the sum of squared errors. This is exactly the loss function chosen in the previous section. Setting the derivative to zero gives the condition

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \theta_j} = -2 \sum_{i=1}^{n} \left(x_j^{(i)} \left[y^{(i)} - \mathbf{x}^{(i)}\boldsymbol{\theta}\right]\right) = 0, \quad \Rightarrow -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} = \mathbf{0}, \tag{23}$$

which yields the same explicit solution as in the previous section, that is, the maximum likelihood estimator (given that the errors are Gaussian and i.i.d.) is given by

$$\boxed{\boldsymbol{\theta}_{\text{MLE}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}} \tag{24}$$

We get one additional result from the probabilistic perspective. We can also ask, what is the value of σ that maximizes the likelihood? In other words, what is the MLE of the variance? Let's compute

$$\frac{\partial \ell(\boldsymbol{\theta}, \sigma)}{\partial \sigma} = -\frac{n}{\sigma} + \frac{1}{\sigma^3} \sum_{i=1}^{n} \left(y^{(i)} - \mathbf{x}^{(i)^T} \boldsymbol{\theta} \right)^2 = 0, \tag{25}$$

which can be solved for σ resulting in

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - \mathbf{x}^{(i)^T} \boldsymbol{\theta} \right)^2 = \frac{1}{n} \sum_{i=1}^{n} \varepsilon^{(i)^2}. \tag{26}$$

Hence, the MLE of the variance is the mean squared error,

$$\boxed{\sigma^2_{\text{MLE}} = \frac{1}{n} \boldsymbol{\varepsilon}^T \boldsymbol{\varepsilon} = \frac{1}{n} (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})} \tag{27}$$

1.3 Robustness and regularization

A problem with the least squares solution (or the MLE) $\boldsymbol{\theta}$ in Eq. (9) is that to compute it, we need to invert the normal matrix $\mathbf{X}^T\mathbf{X}$. The normal matrix may be ill-conditioned or singular. If we have many more features and parameters than samples ($m > n$), the equation system given by the normal equation is *underdetermined*, and the inverse of $\mathbf{X}^T\mathbf{X}$ does not exist, $\mathbf{X}^T\mathbf{X}$ is singular. In that case, no unique solution exists. One solution would be to collect more samples. However, this can be quite expensive or completely infeasible. A trick to improve the robustness of the estimator in this situation is to add a regularization term and solve

$$\boxed{\boldsymbol{\theta}_{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \delta^2\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}} \tag{28}$$

with a scalar regularization parameter δ and where $\mathbf{I}$ denotes the identity matrix, a diagonal matrix where all diagonal elements are 1.

The modified $\boldsymbol{\theta}_{\text{ridge}}$ is the minimizer of the following regularized quadratic loss function

$$L_{\text{ridge}}(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \delta^2 \boldsymbol{\theta}^T\boldsymbol{\theta} = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \delta^2 ||\boldsymbol{\theta}||_2^2. \tag{29}$$

Hence, to minimize L_{ridge} we need to minimize the squared error but at the same time minimize the magnitude of the parameters $\boldsymbol{\theta}$. This is also called ridge regression or ℓ_2-penalized least squares (because we penalize large 2-norms of the parameter vector).

Remark 1. *The regularized matrix $(\mathbf{X}^T\mathbf{X} + \delta^2\mathbf{I})$ with $\delta \neq 0$ is always invertible (non-singular). To check this, we can check that the matrix is positive definite (since all positive definite matrices are invertible). A matrix $\mathbf{A}$ is positive definite if $\mathbf{v}^T\mathbf{A}\mathbf{v} > 0$ for all non-zero vectors $\mathbf{v}$. Indeed, for any non-zero δ, we have*

$$\mathbf{z}^T \left(\mathbf{X}^T\mathbf{X} + \delta^2\mathbf{I} \right) \mathbf{z} = \mathbf{z}^T\mathbf{X}^T\mathbf{X}\mathbf{z} + \delta^2\mathbf{z}^T\mathbf{I}\mathbf{z} = (\mathbf{X}\mathbf{z})^T(\mathbf{X}\mathbf{z}) + \delta^2\mathbf{z}^T\mathbf{z} = \|\mathbf{X}\mathbf{z}\|_2^2 + \delta^2\|\mathbf{z}\|_2^2 > 0.$$

Moreover, it can be useful in practice to use a simplified form of the matrix inversion lemma to reformulate the solution. It holds the following identity (push-through identity)

$$(\mathbf{I} + \mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T = \mathbf{X}^T(\mathbf{I} + \mathbf{X}\mathbf{X}^T)^{-1}. \tag{30}$$

(This can be obtained from $\mathbf{X}^T(\mathbf{I} + \mathbf{X}\mathbf{X}^T) = \mathbf{X}^T + \mathbf{X}^T\mathbf{X}\mathbf{X}^T = (\mathbf{I} + \mathbf{X}^T\mathbf{X})\mathbf{X}^T$ and multiplying with the inverses of the bracket terms from both sides.) Applied to our case we have

$$\boxed{\boldsymbol{\theta}_{\text{ridge}} = \mathbf{X}^T(\delta^2\mathbf{I} + \mathbf{X}\mathbf{X}^T)^{-1}\mathbf{y}} \tag{31}$$

Now if $\mathbf{X}$ is flat and $\mathbf{X}^T$ tall (i.e. we have many more features than samples), computing $(\delta^2\mathbf{I} + \mathbf{X}\mathbf{X}^T)^{-1}$ instead of $(\mathbf{X}^T\mathbf{X} + \delta^2\mathbf{I})^{-1}$ means inverting a much smaller matrix. We will also see that this reformulation is beneficial when generalizing ridge regression to nonlinear function relationships via the kernel trick.

2 Logistic regression

Logistic regression is a classification model and one of the most widely used machine learning models in industry. There is a connection between linear regression and logistic regression although the former is a regression algorithm (fitting continuous data) and the latter a classification algorithm (fitting discrete binary data). (Both models are also part of a larger class of models from statistics called generalized linear models (GLM) [5] not discussed further here.) The logistic regression algorithm introduced here is a binary classification model and the labels $y^{(i)} \in \{0, 1\}$, that is they are either 0 or 1.

Let us define $p := P(y^{(i)} = 1 \mid \mathbf{x}^{(i)})$, that is the probability of the class label being 1 (or an event being true), given the sample features $\mathbf{x}^{(i)}$. The probability of the opposite event, $y^{(i)} = 0$, is then given by $P(y^{(i)} = 0 \mid \mathbf{x}^{(i)}) = 1 - p$. Using this, we can define the *odds*: $\frac{p}{p-1}$, meaning the odds that something is true versus it not being true. It turns out that the log-odds (the natural logarithm of the odds) is a very useful function that we will call the logit function defined as

$$\text{logit}(p) := \log \frac{p}{1-p}, \quad \text{logit} : (0, 1) \to \mathbb{R}. \tag{32}$$

It is a function that takes values in the range $(0, 1)$ and maps them to the entire range of the real numbers $\mathbb{R}$. Hence, we found a way via probabilities to turn discrete events into real numbers.

Logistic regression assumes the model

$$\text{logit}(p) = \mathbf{x}^{(i)}\boldsymbol{\theta} = b + \mathbf{x}^{(i)}\mathbf{w}, \tag{33}$$

which is a linear model for the log-odds parameterized by $\boldsymbol{\theta}$.

Remark 2. *We introduced here another common notation for the parameters. For m features, we have*

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_m \end{bmatrix} := \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix}, \quad \textit{that is} \quad \mathbf{w} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m \end{bmatrix}, \quad b = \theta_0. \tag{34}$$

We usually call $\mathbf{w}$ *the (vector of) weights and* b *the (scalar) bias. We will later see that we can also interpret* $\mathbf{w}$ *and* b *as the normal vector and the offset defining a hyperplane that is the decision boundary.*

Instead of the log-odds directly, we are actually interested in p. Solving Eq. (33) for p gives

$$p = \sigma(z) = \frac{1}{1 + \exp(-z)}, \quad z := b + \mathbf{x}^{(i)}\mathbf{w} = \mathbf{x}^{(i)}\boldsymbol{\theta}. \tag{35}$$

Definition 2 (Logisitic funcion)**.** *The function*

$$\sigma : \mathbb{R} \to (0, 1), \quad \sigma(z) = \frac{1}{1 + \exp(-z)},$$

is called the sigmoid function *or the* logisitic function *and maps from the real numbers onto the open interval* $(0, 1)$ *with* $\sigma(0) = 0.5$.

Remark 3. *We imagine the following procedure. Using linear regression, we get a predictive model for the log-odds. Next, using* σ *we convert to a probability. Finally, using a simple thresholding, we can recover a class label prediction as*

$$\hat{y}(z) = \begin{cases} 1 & \textit{if } \sigma(z) \leq 0.5 \\ 0 & \textit{otherwise} \end{cases} = \begin{cases} 1 & \textit{if } z \leq 0 \\ 0 & \textit{otherwise.} \end{cases} \tag{36}$$

This sounds like a good plan.

We first introduce a simple algebraic trick. We note that using our parameterized model, $P(y^{(i)} = 1|\mathbf{x}^{(i)}; \boldsymbol{\theta}) = \sigma(z)$ and $P(y^{(i)} = 0 \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}) = 1 - \sigma(z)$. We can rewrite this as a single statement

$$P(y^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}) = \sigma(z)^{y^{(i)}}(1 - \sigma(z))^{(1-y^{(i)})}, \tag{37}$$

recalling that $y^{(i)} \in \{0, 1\}$.

To learn the weights for logistic regression, we will use maximum likelihood estimation like in Section 1.2. Let's formulate the likelihood function for $\boldsymbol{\theta}$,

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= P(\mathbf{y} \mid \mathbf{X}; \boldsymbol{\theta}) \\ &= \prod_{i=1}^{n} P(y^{(i)} \mid x^{(i)}; \boldsymbol{\theta}), \qquad \text{(samples are i.i.d.)} \\ &= \prod_{i=1}^{n} \left[\sigma(z)^{y^{(i)}} (1 - \sigma(z))^{(1-y^{(i)})}\right]. \end{aligned} \tag{38}$$

(In comparison with Eq. (20), we note that the Gaussian probability density function has been replaced with the probability mass function of the Bernoulli distribution which models the probability of events with binary outcome.)

Again, we want to maximize the likelihood. To make the computation simpler and more robust (we can avoid underflow error for small probabilities) we use the log-likelihood

$$\begin{aligned}\ell(\boldsymbol{\theta}) &= \log \mathcal{L}(\boldsymbol{\theta}) = \log P(\mathbf{y} \mid \mathbf{X}; \boldsymbol{\theta}) \\ &= \log \prod_{i=1}^{n} \left[\sigma(z)^{y^{(i)}} (1 - \sigma(z))^{(1-y^{(i)})}\right] \\ &= \sum_{i=1}^{n} \left[y^{(i)} \log(\sigma(z)) + (1 - y^{(i)}) \log(1 - \sigma(z))\right]. \end{aligned} \tag{39}$$

Recall that $z = z(\boldsymbol{\theta}) = \mathbf{x}^{(i)}\boldsymbol{\theta}$. Maximizing the log-likelihood is the same as minimizing the negative log-likelihood, $L(\boldsymbol{\theta}) = -\ell(\boldsymbol{\theta})$, and the minimizer of the negative log-likelihood is the maximizer of the likelihood. Unlike linear regression, there is no closed-form solution for the maximum likelihood estimator. We therefore use gradient descent to find the minimizer of the negative log-likelihood by the iterative procedure

$$\boldsymbol{\theta}^{n+1} = \boldsymbol{\theta}^{n} - \eta \frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \boldsymbol{\theta}^{n} + \eta \frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}, \tag{40}$$

with learning rate η. To compute the derivative, note that

$$\frac{\partial \sigma(z)}{\partial z} = \frac{\partial}{\partial z}\left(\frac{1}{1 + e^{-z}}\right) = \frac{e^{-z}}{(1 + e^{-z})^2} = \sigma(z)\frac{e^{-z}}{1 + e^{-z}} = \sigma(z)\frac{1 + e^{-z} - 1}{1 + e^{-z}} = \sigma(z)(1 - \sigma(z)). \tag{41}$$

Moreover, using the chain rule

$$\frac{\partial(\log \sigma(z))}{\partial \boldsymbol{\theta}} = \frac{\partial \log \sigma}{\partial \sigma}\frac{\partial \sigma}{\partial z}\frac{\partial z}{\partial \boldsymbol{\theta}} = \frac{1}{\sigma}\sigma(1 - \sigma)\mathbf{x}^{(i)} = (1 - \sigma(z))\mathbf{x}^{(i)}, \tag{42}$$

and

$$\frac{\partial(\log(1 - \sigma(z)))}{\partial \boldsymbol{\theta}} = \frac{\partial \log(1 - \sigma)}{\partial(1 - \sigma)}\frac{\partial(1 - \sigma)}{\partial z}\frac{\partial z}{\partial \boldsymbol{\theta}} = -\frac{1}{1 - \sigma}\sigma(1 - \sigma)\mathbf{x}^{(i)} = -\sigma(z))\mathbf{x}^{(i)}. \tag{43}$$

Hence, the derivative of the loss function is given by

$$\begin{aligned}\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= -\sum_{i=1}^{n} \left[y^{(i)}(1 - \sigma(z))\mathbf{x}^{(i)} - (1 - y^{(i)})\sigma(z)\mathbf{x}^{(i)}\right] \\ &= -\sum_{i=1}^{n} \left[(y^{(i)} - \sigma(z))\mathbf{x}^{(i)}\right] = \mathbf{0}, \end{aligned} \tag{44}$$

or component-wise,

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \theta_j} = -\sum_{i=1}^{n} \left(x_j^{(i)} \left[(y^{(i)} - \sigma(\mathbf{x}^{(i)}\boldsymbol{\theta}))\right]\right) = 0. \tag{45}$$

Interestingly, this is the same loss function derivative as for linear regression (and Adaline), Eq. (23), if we were to choose $\sigma(z) = z = \mathbf{x}^{(i)}\boldsymbol{\theta}$, that is the identity function instead of the sigmoid function. (It turns out that for all GLMs, the loss function looks the same apart from the choice of activation function σ.) However here, since σ is the nonlinear sigmoid function for logistic regression, we cannot find a closed-form solution and need an iterative method like gradient descent. That means, given a learning rate η, we iterate

$$\boldsymbol{\theta}^{n+1} = \boldsymbol{\theta} - \eta \frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}. \tag{46}$$

2.1 Newton's method

An iterative method with much faster convergence for strictly convex optimization problems is Newton's method. Newton's method is an algorithm to find the root of a function $\mathbf{f}(\boldsymbol{\theta})$, that is find $\boldsymbol{\theta}$ such that $\mathbf{f}(\boldsymbol{\theta}) = \mathbf{0}$.

We start with the simple scalar case of a single parameter θ. Newton's method follows the following procedure. At the current guess (for the first guess we initialize θ with something that we believe may be close to the root), θ^n, evaluate the derivative of the function, $\frac{\partial f(\theta)}{\partial \theta^n}$. Draw a tangent line with the slope $\frac{\partial f(\theta)}{\partial \theta^n}$ and find the root (zero point) of this line. This is θ^{n+1}. The slope of a line is given by the ratio of height ($f(\theta)$) and corresponding width ($\Delta\theta$),

$$\frac{\partial f(\theta)}{\partial \theta}(\theta^n) := f'(\theta^n) = \frac{f(\theta)}{\theta^{n+1} - \theta^n} := \frac{f(\theta)}{\Delta\theta}. \tag{47}$$

Solving for θ^{n+1}, we obtain the iteration rule

$$\theta^{n+1} = \theta^n - \frac{f(\theta^n)}{f'(\theta^n)}. \tag{48}$$

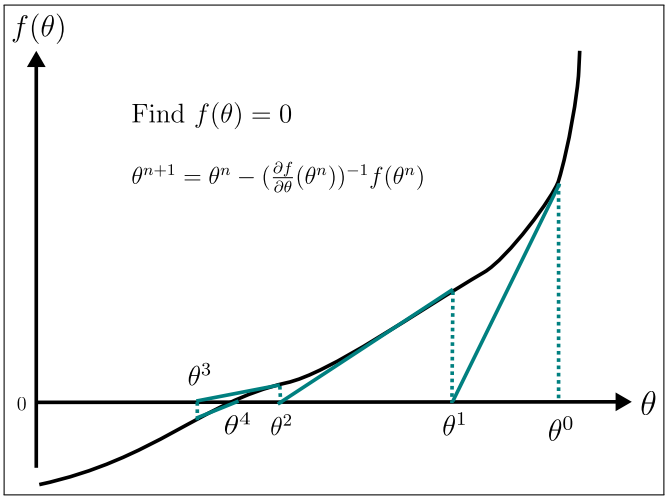

Figure 1: Newton's method for function of one parameter

In the vector-valued case $\mathbf{F}(\boldsymbol{\theta})$ ($\mathbf{F}$ outputs a vector), the function derivative is a matrix and called the Jacobian

$$\mathbf{J}_F := \frac{\partial \mathbf{F}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}. \tag{49}$$

Newton's method can then be written in this way (where we now use the matrix inverse)

$$\boldsymbol{\theta}^{n+1} = \boldsymbol{\theta}^n - \mathbf{J}(\boldsymbol{\theta}^n)^{-1}\mathbf{F}(\boldsymbol{\theta}^n). \tag{50}$$

In the case of logistic regression (or linear regression), the function we want to find the root of is $\mathbf{F}(\boldsymbol{\theta}) = \frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbf{0}$. If we derive this function again, to obtain the Jacobian matrix $\mathbf{J}_F$ of $\mathbf{F}(\boldsymbol{\theta})$, we derive by the parameters. Hence, the Jacobian of $\mathbf{F}(\boldsymbol{\theta})$ is the Hessian (matrix containing all second derivatives) of the loss function $L(\boldsymbol{\theta})$.

Therefore, in our case Newton's method becomes

$$\boldsymbol{\theta}^{n+1} = \boldsymbol{\theta}^n - \mathbf{J}(\boldsymbol{\theta}^n)^{-1}\mathbf{F}(\boldsymbol{\theta}^n) = \boldsymbol{\theta}^n - \mathbf{H}(\boldsymbol{\theta}^n)^{-1}\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^n) = \boldsymbol{\theta}^n - \left(\frac{\partial^2 L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}\partial \boldsymbol{\theta}}\right)^{-1}\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^n) \tag{51}$$

where $\mathbf{H}(\boldsymbol{\theta})$ is called the Hessian matrix of the loss function and its entries are given by

$$H_{ij} = \frac{\partial^2 L(\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j}, \tag{52}$$

the derivative $\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ and the Hessian are evaluated at $\boldsymbol{\theta}_n$.

Remark 4. *In the vicinity of the solution (root), Newton's method converges quadratically. That means the number of significant digits doubles in each iteration. This is why Newton method's can be so fast. For non-convex problems, Newton's method may diverge or find at most one root in case there are many. For strictly convex problems (like logistic regression), convergence is guaranteed and usually fast.*

Remark 5. *The Hessian might be very expensive to compute. For so-called quasi-Newton methods, the Hessian is not explicitly formed but only approximated. This improves performance per iteration at the cost of a higher number of iterations. A popular quasi-Newton method for machine learning is the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS)[1, 2, 3, 6] or the limited-memory variant L-BFGS[4] for a large number of variables.*

For linear problems, Newton's method converges in a single step. Remember linear regression with least squares? The derivative of the objective function was given by the linear function

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\boldsymbol{\theta}, \tag{53}$$

and the Hessian is therefore the following constant

$$\mathbf{H}(\boldsymbol{\theta}) = \frac{\partial}{\partial \boldsymbol{\theta}}\left(\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right) = \frac{\partial^2 L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}\partial \boldsymbol{\theta}} = 2\mathbf{X}^T\mathbf{X}. \tag{54}$$

Using the iteration rule of Newton's method yields

$$\begin{aligned}\boldsymbol{\theta}^1 &= \boldsymbol{\theta}^0 - (2\mathbf{X}^T\mathbf{X})^{-1}(-2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\boldsymbol{\theta}^{\mathbf{0}}) \\ &= \boldsymbol{\theta}^0 - (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} - \boldsymbol{\theta}^0 \\ &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}.\end{aligned}$$

This shows that we converge in one step (to the solution derived above for the least squares problem) regardless of the initial guess $\boldsymbol{\theta}^0$.

2.2 One-versus-Rest: classification with more than two classes

Some algorithms can differentiate more than two classes. However, every binary classification model can be turned into a multi-class classification model using a technique called *One-versus-Rest* (OvR).

Let κ denote the number of classes in the set of classes κ. We train κ binary classifiers where for classifier $k \in K$ the labels are 1 for the κ-th class and 0 for all other classes. For the prediction of a new sample, we predict with all κ classifiers. We then take the result of the classifier with the highest confidence (highest probability of the predicted class label). For this to work well, we typically want the classifier to be able to predict probabilities (and not just discrete class labels) to reduce the chance of ambiguities in the final decision.

2.3 Start simple

We learned about the most common regression and classification models, linear regression, and logistic regression. When starting to analyze new data, it is often best to start with one of these simple models and evaluate performance before moving on to more complex techniques! But so far, we are only able to model linear relationships in the data. That seems like a big restriction. We will learn next how to model nonlinear relationships using some machine learning techniques originally derived to express linear relationships (like ridge regression or logistic regression).

References

[1] C. G. Broyden. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970. ISSN 1464-3634. doi:10.1093/imamat/6.1.76.

[2] R. Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322, 1970. ISSN 1460-2067. doi:10.1093/comjnl/13.3.317.

[3] Donald Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109):23–26, 1970. ISSN 1088-6842. doi:10.1090/s0025-5718-1970-0258249-6.

[4] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1–3):503–528, 1989. ISSN 1436-4646. doi:10.1007/bf01589116.

[5] P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Springer US, 1989. ISBN 9781489932426. doi:10.1007/978-1-4899-3242-6.

[6] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24(111):647–656, 1970. ISSN 1088-6842. doi:10.1090/s0025-5718-1970-0274029-x.

This work is licensed under a Creative Commons "Attribution-ShareAlike 4.0 International" license.