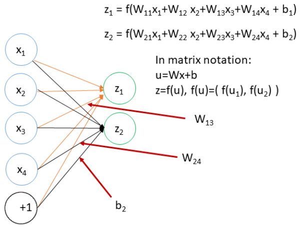
Chapter 2: Introduction to Machine Learning – ML Basics



$z_1 = f(W_{11}x_1+W_{12}x_2+W_{13}x_3+W_{14}x_4 + b_1)$
$z_2 = f(W_{21}x_1+W_{22}x_2+W_{23}x_3+W_{24}x_4 + b_2)$
In matrix notation:
$u=Wx+b$
$z=f(u)$, $f(u)=(f(u_1), f(u_2))$
x_1
x_2
x_3
x_4
+1
z_1
z_2
W_{13}
W_{24}
b_2

Christian Heumann

October 2021

Overview

- Linear Algebra
- Probabilities in NLP
- Train-test-Splits
- Loss functions
- Performance measures
- Useful functions

Scalars, vectors, matrices and tensors I

- A scalar is a single number, e.g. a real number $x \in \mathbb{R}$, or a natural number $n \in \mathbb{N}$
- A vector is an array of (real) numbers. We define vectors as column vectors. A vector $\boldsymbol{x}$ of dimension n, $\boldsymbol{x} \in \mathbb{R}^n$ is written as

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

  x_i returns the element of the vector at row i.

Scalars, vectors, matrices and tensors II

- A matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ is a two-dimensional array (of real numbers) of dimension $n \times p$. We use usually capital letters, i.e.

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{12} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}$$

  $\boldsymbol{X}_{i,j} = x_{ij}$ is the element in row i and column j. $\boldsymbol{X}_{:,j}$ returns the whole column j, $\boldsymbol{X}_{i,:}$ the i-th row of $\boldsymbol{X}$.

- A tensor is a general multi-dimensional array, e.g. a three-dimensional $n \times p \times m$ array $X \in \mathbb{R}^{n \times p \times m}$. The element in cell (i, j, k) is denoted by $x_{i,j,k}$.

Transpose

- The transpose of a matrix is denoted by ^{T} and exchanges rows and columns, e.g.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

  That means that $\boldsymbol{X}^T_{i,j} = \boldsymbol{X}_{j,i} = x_{ji}$.

Addition

- Two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can usually only be added if they have the same dimensions. The addition $\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B}$ is done element by element, i.e. $c_{ij} = a_{ij} + b_{ij}$. Example:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} = \begin{bmatrix} 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix}$$

- If b is a scalar, $\boldsymbol{C} = \boldsymbol{A} + b$ means that b is added to each element of $\boldsymbol{A}$, i.e. $\boldsymbol{C}_{i,j} = c_{ij} = a_{ij} + b$.

Multiplication I

- Two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can only be multiplied if the number of columns of $\boldsymbol{A}$ is equal to the number of rows of $\boldsymbol{B}$, i.e. if $\boldsymbol{A}$ is $n \times p$ and $\boldsymbol{B}$ is $p \times m$. The multiplication $\boldsymbol{C} = \boldsymbol{AB}$ is done as

$$\boldsymbol{C}_{i,j} = c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}$$

  i.e. row i of $\boldsymbol{A}$ is multiplied by column j of $\boldsymbol{B}$. $\boldsymbol{C}$ is of dimension $n \times m$. Example:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 \\ 15 & 16 & 17 & 18 \end{bmatrix} = \begin{bmatrix} 74 & 80 & 86 & 92 \\ 173 & 188 & 203 & 218 \end{bmatrix}$$

- The dot product of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of same length n is defined as the scalar value $\boldsymbol{x}^T \boldsymbol{y} = \sum_{i=1}^{n} x_i y_i$. Therefore $\boldsymbol{C}_{ij}$ is also the dot product of row i of $\boldsymbol{A}$ and column j of $\boldsymbol{B}$.

Multiplication II

- If b is a scalar, $\boldsymbol{C} = b\boldsymbol{A}$ means that each element of $\boldsymbol{A}$ is multiplied by b, i.e. $C_{i,j} = c_{ij} = ba_{ij}$.
- If two matrices have the same dimensions, the element by element multiplication is called Hadamard product

  $$\boldsymbol{C} = \boldsymbol{A} \odot \boldsymbol{B}$$

  i.e. $c_{ij} = a_{ij}b_{ij}$.
- The Kronecker product of $\boldsymbol{A}$ ($n \times p$) and $\boldsymbol{B}$ returns a matrix $\boldsymbol{C}$ where each (scalar) element of $\boldsymbol{A}$ is multiplied by the matrix $\boldsymbol{B}$, i.e. $c_{ij} = a_{ij}\boldsymbol{B}$ so that

  $$\boldsymbol{C} = \boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \dots & a_{1p}\boldsymbol{B} \\ \vdots & \vdots & & \vdots \\ a_{n1}\boldsymbol{B} & a_{n2}\boldsymbol{B} & \dots & a_{np}\boldsymbol{B} \end{bmatrix}$$

Calculation rules

- $\boldsymbol{C}(\boldsymbol{A}+\boldsymbol{B}) = \boldsymbol{C}\boldsymbol{A}+\boldsymbol{C}\boldsymbol{B}$
- $\boldsymbol{C}(\boldsymbol{B}\boldsymbol{A}) = (\boldsymbol{C}\boldsymbol{B})\boldsymbol{A}$
- $(\boldsymbol{A}\boldsymbol{B})^T = \boldsymbol{B}^T\boldsymbol{A}^T$
- For n vectors $\boldsymbol{x}, \boldsymbol{y}$: $\boldsymbol{x}^T\boldsymbol{y} = (\boldsymbol{x}^T\boldsymbol{y})^T = \boldsymbol{y}^T\boldsymbol{x}$
- Trace of a quadratic $n \times n$ matrix $\boldsymbol{X}$:

$$\mathrm{tr}\boldsymbol{X} = \sum_{i=1}^{n} x_{ii}$$

There are also useful rules, e.g. $\mathrm{tr}(\boldsymbol{A}+\boldsymbol{B}) = \mathrm{tr}\boldsymbol{A} + \mathrm{tr}\boldsymbol{B}$ or $\mathrm{tr}(\boldsymbol{A}\boldsymbol{B}) = \mathrm{tr}(\boldsymbol{B}\boldsymbol{A})$.

Special matrices

- Inverse of a quadratic $n \times n$ matrix $\boldsymbol{A}$: $\boldsymbol{A}^{-1}$, such that $\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}_n$, where $\boldsymbol{I}_n$ is the (quadratic) n-dimensional identity matrix with 1's on the diagonal and 0's elsewhere.
- Inverse exists if all rows (columns) are linearly independent.
- If $\boldsymbol{A}$ is symmetric and positive definite, the inverse exists, all eigenvalues are positive and real, and the following decomposition exists:

$$\boldsymbol{A} = \boldsymbol{\Gamma}\boldsymbol{\Lambda}\boldsymbol{\Gamma}^T$$

  where the columns of $\boldsymbol{\Gamma}$ are the orthogonal eigenvectors of $\boldsymbol{A}$ and $\boldsymbol{\Lambda}$ is a diagonal matrix with the eigenvalues on the diagonal. All matrices are real valued in that case.
  Further, $\boldsymbol{A}^{-1} = \boldsymbol{\Gamma}\boldsymbol{\Lambda}^{-1}\boldsymbol{\Gamma}^T$.

Norms

- L_p norm of an n vector $\boldsymbol{x}$:

$$||\boldsymbol{x}||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$$

  Special cases: L_1 norm, L_2 norm (Euclidean norm)
- Max norm: $||\boldsymbol{x}||_\infty = \max_i |x_i|$
- Frobenius norm of a $n \times p$ matrix:

$$||\boldsymbol{X}||_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{p} x_{ij}^2}$$

- Applications: L_1 distance, L_2 distance of two vectors, Frobenius distance of two matrices
- $\boldsymbol{x}^T \boldsymbol{x} = (||\boldsymbol{x}||_2)^2 = ||\boldsymbol{x}||_2^2$
- $\boldsymbol{x}^T \boldsymbol{y} = ||\boldsymbol{x}||_2 ||\boldsymbol{y}||_2 \cos(\phi)$, where ϕ is the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$.

Derivations I

Formulas for derivations of expressions wrt a vector (or a matrix) are useful tools for optimizations. The gradient notation $\nabla_{\boldsymbol{w}}$ is used for the derivative of a scalar wrt a vector $\boldsymbol{w}$.

- Derivative of a scalar product two vectors wrt one vector (we always assume that the dimensions are compatible):

$$\frac{\partial}{\partial \boldsymbol{w}} \boldsymbol{w}^T \boldsymbol{x} = \nabla_{\boldsymbol{w}} \boldsymbol{w}^T \boldsymbol{x} = \boldsymbol{x}$$

This can be shown as follows:

$$\frac{\partial}{\partial \boldsymbol{w}} \boldsymbol{w}^T \boldsymbol{x} = \begin{bmatrix} \frac{\partial}{\partial w_1}(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n) \\ \frac{\partial}{\partial w_2}(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n) \\ \vdots \\ \frac{\partial}{\partial w_n}(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n) \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \boldsymbol{x}$$

Derivations II

- Let $\boldsymbol{A}$ symmetric. Then

$$\nabla_{\boldsymbol{w}} \boldsymbol{w}^T \boldsymbol{A} \boldsymbol{w} + \boldsymbol{b}^T \boldsymbol{w} = 2\boldsymbol{A}\boldsymbol{w} + \boldsymbol{b}$$

This is later applied in the linear (regression) model.

Derivations III

With the previous rule, it suffices to show that $\nabla_{\boldsymbol{w}} \boldsymbol{w}^T \boldsymbol{A} \boldsymbol{w} = 2\boldsymbol{A}\boldsymbol{w}$.

$$
\begin{aligned}
\frac{\partial}{\partial \boldsymbol{w}} \boldsymbol{w}^T \boldsymbol{A} \boldsymbol{w} &= \begin{bmatrix} \frac{\partial}{\partial w_1} (\sum_{i=1} w_i^2 a_{ii} + 2 \sum_{i<j} w_i w_j a_{ij}) \\ \frac{\partial}{\partial w_2} (\sum_{i=1} w_i^2 a_{ii} + 2 \sum_{i<j} w_i w_j a_{ij}) \\ \vdots \\ \frac{\partial}{\partial w_n} (\sum_{i=1} w_i^2 a_{ii} + 2 \sum_{i<j} w_i w_j a_{ij}) \end{bmatrix} \\
&= \begin{bmatrix} 2w_1 a_{11} + 2 \sum_{j \neq 1} w_j a_{1j} \\ 2w_2 a_{22} + 2 \sum_{j \neq 2} w_j a_{2j} \\ \vdots \\ 2w_n a_{nn} + 2 \sum_{j \neq n} w_j a_{nj} \end{bmatrix} \\
&= \begin{bmatrix} 2 \sum_{j=1}^{n} w_j a_{1j} \\ 2 \sum_{j=1}^{n} w_j a_{2j} \\ \vdots \\ 2 \sum_{j=1}^{n} w_j a_{nj} \end{bmatrix} = 2\boldsymbol{A}\boldsymbol{w}
\end{aligned}
$$

Derivations IV

- Further example: Let $f : \mathbb{R} \to \mathbb{R}$ a scalar (non-linear) function.

$$\boldsymbol{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \quad \boldsymbol{W} = \begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

and $\boldsymbol{z} = f(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$, i.e. f is applied elementwise:

$$\begin{aligned} u_1 &= W_{11}x_1 + W_{12}x_2 + W_{13}x_3 + W_{14}x_4 + b_1 = \boldsymbol{W}_{1,:}\boldsymbol{x} + b_1 \\ u_2 &= W_{21}x_1 + W_{22}x_2 + W_{23}x_3 + W_{24}x_4 + b_2 = \boldsymbol{W}_{2,:}\boldsymbol{x} + b_2 \\ z_1 &= f(u_1) = f(W_{11}x_1 + W_{12}x_2 + W_{13}x_3 + W_{14}x_4 + b_1) = f(\boldsymbol{W}_{1,:}\boldsymbol{x} + b_1) \\ z_2 &= f(u_2) = f(W_{21}x_1 + W_{22}x_2 + W_{23}x_3 + W_{24}x_4 + b_2) = f(\boldsymbol{W}_{2,:}\boldsymbol{x} + b_2) \end{aligned}$$

Derivations V

Then, using the chain rule (note, that z_1 only depends on u_1, z_2 only on u_2):

$$\begin{aligned}\frac{\partial z_i}{\partial W_{ij}} &= \frac{\partial z_i}{\partial u_i}\frac{\partial u_i}{\partial W_{ij}} \\ &= \frac{\partial f(u_i)}{\partial u_i}x_j = f'(u_i)x_j\end{aligned}$$

Further,

$$\begin{aligned}\frac{\partial z_i}{\partial b_i} &= \frac{\partial z_i}{\partial u_i}\frac{\partial u_i}{\partial b_i} \\ &= \frac{\partial f(u_i)}{\partial u_i} = f'(u_i)\end{aligned}$$

Derivations VI

Assume, that we have an additional (non-linear) function g:

$$\begin{aligned} t &= h_1 z_1 + h_2 z_2 \\ v &= g(t) \end{aligned}$$

then $(j = 1, 2, 3, 4)$

$$\begin{aligned} \frac{\partial v}{\partial x_j} &= \frac{\partial v}{\partial z_1}\frac{\partial z_1}{\partial x_j} + \frac{\partial v}{\partial z_2}\frac{\partial z_2}{\partial x_j} \\ &= \frac{\partial v}{\partial z_1}\frac{\partial z_1}{\partial u_1}\frac{\partial u_1}{\partial x_j} + \frac{\partial v}{\partial z_2}\frac{\partial z_2}{\partial u_2}\frac{\partial u_2}{\partial x_j} \\ &= [g'(t)h_1]f'(u_1)W_{1j} + [g'(t)h_2]f'(u_2)W_{2j} \end{aligned}$$

Probabilities in NLP

- In NLP, probabilities are put on the words of a corpus or sequences of words (e.g. a sentence)
- These distributions are discrete because each word or each sequence of words is treated as a possible outcome
- Language models are based on probabilities and maximization of the log-likelihood (or minimization of the negative log-likelihood), which is also based on (log-)probabilities.

Discrete random variables

- In NLP, the words may be treated as discrete random variables W.
- The number of words $|V|$ in a corpus is finite, but may be huge (in the millions)
- A realization of W is a certain word w, e.g. *house*
- The words in a sentence occur not independently from each other, i.e. the words are correlated. Therefore, we have to consider sequences of words as multivariate random variables $(W_1, W_2, \ldots, W_n)$.
- A realization is a certain sequence, e.g. $(W_1 = \mathrm{The}, W_2 = \mathrm{dog}, W_3 = \mathrm{is}, W_4 = \mathrm{barking})$.

Probability distributions in NLP

- Joint distribution of a sequence of words $w_1, w_2, \dots, w_n$, n may be any number:

$$P(W_1 = w_1, W_2 = w_2, \dots, W_n = w_n) = P(w_1, w_2, \dots, w_n)$$

  The right hand side of the equation is a short form of the left hand side. Example: $P(\mathrm{The}, \mathrm{dog}, \mathrm{is}, \mathrm{barking})$.
- Special choices of n:
  - $n = 1$: Unigrams
  - $n = 2$: Bigrams

  Generally: n-grams
- Conditional distributions, e.g. the probability of the next word given a sequence of previous words:

$$P(w_{i+1} | w_i, w_{i-1}, \dots, w_1)$$

  The sequence of previous words $(w_i, w_{i-1}, \dots, w_1)$ is also called context of w_{i+1}.

Factorization of the joint distribution

- Sometimes called chain rule for conditional probabilities
- The joint distribution can be written as a product of conditional distributions

$$\begin{aligned}
&P(w_1, w_2, \ldots, w_n) \\
&\quad = P(w_1)P(w_2|w_1)P(w_3|w_2, w_1)\cdots P(w_n|w_{n-1}, \ldots, w_1) \\
&\quad = P(w_1)\prod_{i=2}^{n} P(w_i|w_{i-1}, \ldots, w_1)
\end{aligned}$$

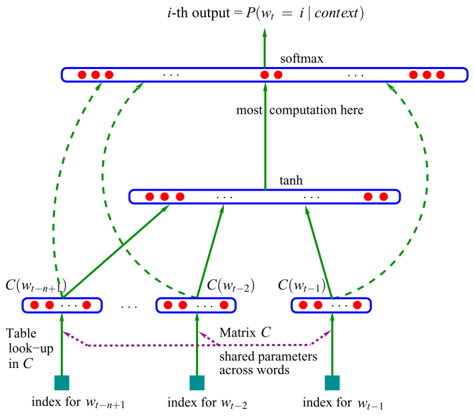
Markov assumptions

- The context of a word cannot be arbitrary long
- A fixed (or maximum) size for the context is usually assumed
- Under the Markov assumptions, the joint distribution can be factorized by conditional probabilities with a fixed context size.
- Example: Assume that $P(w_i|w_{i-1},\ldots,w_1) = P(w_i|w_{i-1})$ (i.e., we only consider bigrams). Then

$$P(w_1, w_2, \ldots, w_n) = P(w_1)\prod_{i=2}^{n} P(w_i|w_{i-1})$$

Bengio's et al. neural network language model (NNLM)

- In 2003, Bengio et al. introduced the NNLM which uses the introduced factorization with the Markov property



More on random numbers and probabilities

- Discrete random variables also occur in other contexts than NLP. Examples:
  - Binary random variables (Bernoulli experiments) are a model for experiments with two possible outcomes (0 or 1), e.g. coin tossing.
  - Poisson random numbers are often for counts, e.g. the number of car accidents during a certain interval (month, year) in a city.
- Random numbers can also be continuous, e.g. Gaussian random numbers, i.e., X is Gaussian, if $X \sim N(\mu, \sigma^2))$ where μ and σ^2 are two parameters which have usually to be estimated from the data. The distribution of random numbers is often given in terms of a *probability density* $f(x)$ which is positive and integrates to 1. For a Gaussian, the density is

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\} \ .$$

Expectation and variance

- If g is a function of a random variable X with density function $p(x)$, the expectation or expected value of g is the mean value or average over all possible values of X:

$$E(g(X)) = \int g(x)p(x)dx \ .$$

  If f is disrete, the integral is replaced by a sum.

- The variance of g is then

$$Var[g(X)] = E\{[g(X) - E(g(X))]^2\} = \int [g(x) - E(g(X))]^2 p(x)dx \ .$$

Theorem of Bayes

- The theorem of Bayes can be stated for probabilities and densities
  - For probabilities, it solves the following problem: if it is known that an event B has occured (e.g. that the thrown number on top of a dice is odd) than the probability of another event A (e.g. a 3 has been thrown) can be calculated. This is usually denoted as

    $$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

    E.g.: $P(\text{odd number} = \frac{1}{2})$ and

    $$P(3|\text{odd number}) = \frac{\text{odd number and 3}}{P(\text{odd number})} = \frac{\frac{1}{6}}{\frac{1}{2}} = \frac{1}{3}$$

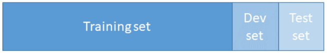
  - For densities: let $p(x, y)$ the joint density of two random variables, $p(x)$ the marginal and $p(y|x)$ the conditional density. Then

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

    This holds also, if $\boldsymbol{x}$ is an n-dimensional random vector .

Train-/Dev-/Test-Splits I

- To train a machine learning model, several strategies are possible. One strategy is to split the whole available training data into two or three parts. If the algorithm requires hyperparameter tuning, a split in three parts is necessary:

  The *training set* is used to generate the ML model (given that certain hyperparameters are fixed at some values), the *dev*elopment *set* is used to compare different hyperparameter values and to select (in best case) the optimal ones or at least those who give the best evaluation on the dev set. The test set is finally used to measure the performance of the ML model with the optimal hyperparameters.
- Another strategy is to use cross validation.

Loss function I

- A loss function $\mathcal{L}$ is a function into the positive real numbers (including 0).
- Let y the actual value and $\hat{y}$ a prediction then the loss function is given by $\mathcal{L} = \mathcal{L}(y, \hat{y})$.
- Loss functions can be discrete or continuous
- For optimization purposes, continuous loss functions are preferable
- Supervised learning tasks try to minimize the *expected* loss $EL = E_{(\boldsymbol{x},y)}(\mathcal{L}(Y, \hat{Y}))$. Usually, the prediction $\hat{Y}$ is estimated as a prediction function $f(\boldsymbol{x})$, depending on the features $\boldsymbol{x}$,

$$\hat{Y} = f(\boldsymbol{x}) \; .$$

The expected loss is then $EL = E_{(\boldsymbol{x},y)}(\mathcal{L}(Y, f(\boldsymbol{x}))$.

Loss function II

- If one chooses a quadratic loss function $\mathcal{L}(Y, f(\boldsymbol{X})) = [Y - f(\boldsymbol{X})]^2$, the expected loss (or prediction error) is

$$E(\mathcal{L}(Y, f(\boldsymbol{X})) = \int [y - f(\boldsymbol{x})]^2 P(dy, d\boldsymbol{x}) \; .$$

  Factorizing $P(Y, \boldsymbol{X}) = P(Y|\boldsymbol{X})P(\boldsymbol{X})$, this can be written as

$$E(\mathcal{L}(Y, f(\boldsymbol{X})) = E_{\boldsymbol{X}} E_{Y|\boldsymbol{X}}([Y - f(\boldsymbol{X})]^2 | \boldsymbol{X})$$

  A pointwise minimization leads to

$$f(\boldsymbol{x}) = \operatorname{argmin}_c E_{Y|\boldsymbol{X}}([Y - c]^2 | \boldsymbol{X} = \boldsymbol{x})$$

  This is called the regression function (conditional expectation) $f(\boldsymbol{x}) = E(Y|\boldsymbol{X} = \boldsymbol{x})$.

Loss function III

- Discrete case: let C the observed class and $\hat{C}(\boldsymbol{X})$ the function which assigns each $\boldsymbol{X}$ a certain class C from $\mathcal{C}$. Let the classes denoted as $1, 2, \ldots, K$. A possible loss function is the 0-1-loss. A wrong assignment is punished by 1:

$$\mathcal{L}(C, C(\boldsymbol{X})) = \begin{cases} 0 & \text{if } C = \hat{C}(\boldsymbol{X}) \\ 1 & \text{if } C \neq \hat{C}(\boldsymbol{X}) \end{cases}$$

It holds $EL = E_{\boldsymbol{X}}[\sum_{k=1}^{K} \mathcal{L}(k, \hat{C}(\boldsymbol{X}))P(k|\boldsymbol{X})]$. A pointwise minimization leads to

$$\hat{C}(\boldsymbol{x}) = \text{argmin}_{c\in\mathcal{C}} \sum_{k=1}^{K} \mathcal{L}(k, c)P(k|\boldsymbol{X} = \boldsymbol{x}) = \text{argmin}_{c\in\mathcal{C}}(1-P(c|\boldsymbol{X} = \boldsymbol{x}))$$

With the 0-1-loss one gets

$$\hat{C}(\boldsymbol{x}) = c_0, \quad \text{if } P(c_0|\boldsymbol{X} = \boldsymbol{x}) = \max_{c\in\mathcal{C}} P(c|\boldsymbol{X} = \boldsymbol{x}) \quad .$$

Loss function IV

This is the so-called Bayes-classifier which assigns the class which has the highest conditional probability.

- Note that the expected loss is estimated from test data using the test samples (empirical loss). Therefore the distributions are not necessary to do the calculation of the empirical loss. We simply calculate the average over all validation or test samples (which is a consistent estimate for the expected loss):

$$\widehat{\mathcal{L}}(Y, f(\boldsymbol{X})) = \frac{1}{m'} \sum_{j=1}^{m'} \mathcal{L}(Y_j, \hat{f}(\boldsymbol{x}_j))$$

  where $\hat{Y}_j = \hat{f}(\boldsymbol{x}_j)$ is the prediction of the jth sample and m' is the number of samples.

- The negative log-likelihood is often used as loss function (we will see an example later).

Performance measures (supervised learning) I

- While the loss functions are used for optimization purposes, performance measures are usually used to evaluate a machine learning model.
- Many supervised tasks are classification tasks, often with two classes (binary task), e.g. sentiment analysis (positive or negative sentiment of a text), but in principle a finite number of classes is possible. In this case, many measures have been proposed which are based on the *confusion matrix*. Example: consider a classifier for images showing either a donkey or a horse:

	observed class	
predicted class	donkey	horse
donkey	4 (TP)	1 (FP)
horse	2 (FN)	3 (TN)

The data behind this table can be summarized as $y = (1,1,1,1,1,1,0,0,0,0)$, where 1 is used for representing a donkey (a "positive" case) and 0 is used for a horse (a "negative"

Performance measures (supervised learning) II

case). Then we call the prediction of a donkey as donkey as *true positive (TP)*, the prediction of a horse as a horse as *true negative (TN)*, the prediction of a donkey as a horse as *false negative (FN)* and the prediction of a horse as a donkey as *false positive (FP)*. All performance measures build on these four numbers:

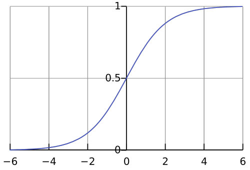
$$
\begin{aligned}
Precision &= \frac{TP}{TP + FP}\\
Recall &= \frac{TP}{TP + FN}\\
F1 &= 2\frac{Precision \cdot Recall}{Precision + Recall}\\
MC &= \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(FP + TN)(FN + TN)}}
\end{aligned}
$$

MC: Matthews correlation coefficient

Performance measures (supervised learning) III

- In multiclass problems, a generalization can be obtained by two approaches. One approach considers all correct predictions versus all wrong predictions (e.g. Micro F1 score). The second focuses on one class and pools all other classes together. Then, binary measures can be calculated for each class separately and then summarized into one measure (e.g. Macro F1 score).

Sigmoid function



- The sigmoid function (also used in logistic regression) is defined as

$$\sigma(z) = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}}$$

Note that $0 \leq \sigma(z) \leq 1$ and $1 - \sigma(z) = \frac{1}{1+e^z}$.

- The derivative is

$$\begin{aligned} \frac{d}{dz}\sigma(z) &= \frac{(1+e^z)e^z - e^z e^z}{(1+e^z)^2} = \frac{e^z}{(1+e^z)^2} \\ &= \frac{e^z}{1+e^z}\frac{1}{1+e^z} \\ &= \sigma(z)(1-\sigma(z)) \end{aligned}$$

Softmax function

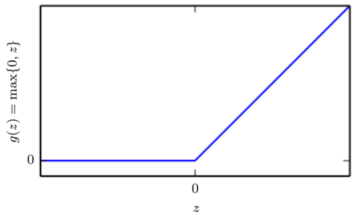
- Let $\boldsymbol{z} \in \mathbb{R}^K$ a real vector $\boldsymbol{z} = (z_1, \ldots, z_K)$. Then the value of the softmax function is also a vector of dimension K with elements

$$\sigma(\boldsymbol{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \ , j = 1, \ldots, K$$

- It holds $0 \leq \sigma(\boldsymbol{z})_j \leq 1$ and $\sum_{k=1}^{K} \sigma(\boldsymbol{z})_j = 1$.
- This normalization property makes it useful for modeling discrete probability distributions
- It is also used in the multinomial logistic regression model.
- The derivative is

$$\frac{\partial}{\partial z_k} \sigma(\boldsymbol{z})_j \quad = \quad \text{exercise!}$$

Rectified linear unit (ReLu)



- The ReLu is defined as

$$g(z) = \max(0, z) = z^+$$

- Piecewise linear function
- Derivative is either 0 ($z < 0$) or 1 ($z > 0$) or undefined ($z = 0$).
- Many variants exist

Addon: multiclass performance measures I

- If the target variable has $k > 2$ classes, then the confusion matrix looks as follows:

		observed (true) class			
		class 1	class 2	$\dots$	class k
	class 1	TP_1			
	class 2		TP_2		
predicted	$\dots$			$\ddots$	
	class k				TP_k

- The accuracy is then

$$\mathrm{Acc} = \frac{1}{m} \sum_{j=1}^{k} TP_j \ ,$$

where m is the number of samples. The accuracy gives an equal weight to each observation. In case of an imbalanced class distribution, classifiers which predict the majority class well will

Addon: multiclass performance measures II

achieve high accuracy but low accuracy for classes with few observations.

- F_1^{macro}-score ($0 \leq F_1^{\mathrm{macro}} \leq 1$):

$$F_1^{\mathrm{macro}} = \frac{1}{m} \sum_{j=1}^{k} F_1^{\mathrm{class_j}} = \frac{1}{m} \sum_{j=1}^{k} 2 \frac{precision_j \cdot recall_j}{precision_j + recall_j}$$

where $precision_j = \frac{TP_j}{TP_j + FP_j}$ and $recall_j = \frac{TP_j}{TP_j + FN_j}$. This version gives each class the same weight. Therefore it is suited for data sets with imbalanced class distributions.

Addon: multiclass performance measures III

- F_1^{micro}-score $(0 \leq F_1^{\text{micro}} \leq 1)$:

$$F_1^{\text{micro}} = 2 \cdot \frac{\textit{precision}_{\text{micro}} \cdot \textit{recall}_{\text{micro}}}{\textit{precision}_{\text{micro}} + \textit{recall}_{\text{micro}}}$$

where

$$\textit{precision}_{\text{micro}} = \frac{\sum_{j=1}^{k} TP_j}{\sum_{j=1}^{k} TP_j + FP_j}$$

and

$$\textit{recall}_{\text{micro}} = \frac{\sum_{j=1}^{k} TP_j}{\sum_{j=1}^{k} TP_j + FN_j} \, .$$

This version gives equal weight to each observation. Therefore its not suited for data sets with an imbalanced class distributions.

Addon: multiclass performance measures IV

- Generalization of Matthews correlation coefficient. Let $\boldsymbol{C}$ the confusion matrix:

$$MCC_{\mathrm{multiclass}} = \frac{m\,\mathrm{tr}\boldsymbol{C} - \sum_{k,l}\boldsymbol{C}_k\boldsymbol{C}_l}{\sqrt{m^2\sum_{k,l}\boldsymbol{C}_k(\boldsymbol{C}^T)_l}\sqrt{m^2\sum_{k,l}(\boldsymbol{C}^T)_k\boldsymbol{C}_l}}$$

  where $\boldsymbol{C}_k$ is the kth row, $\boldsymbol{C}_l$ the lth column, and $\boldsymbol{C}^T$ the transpose of $\boldsymbol{C}$.

Performance measure for multi-label problems: subset accuracy, ...