Machine learning foundations I

Mitko Veta

Eindhoven University of Technology
Department of Biomedical Engineering

2019

Learning goals

At the end of this lecture you will:

- Have an understanding of the goal of machine learning (ML) models.
- Have a good understanding of basic mathematical concepts used in ML and be able to apply them in the design and implementation of ML methods.

Overview

Topics covered in this lecture:

1. Linear algebra

2. Gradient-based optimization

3. Two simple machine learning models
   Linear model
   Nearest-neighbours model

4. Probability theory

Linear algebra

Materials:

- Chapter I.2 from Goodfellow et al., *Deep Learning*
- Kolter et al., "Linear Algebra Review and Reference"

Scalars

- A scalar is a single number (integer, real, rational, ...).
- Denoted by italics a, n, x

Vectors

- A vector is a 1-D array of numbers (integer, real, rational, ...)

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

- Example notation for type and size

$$\boldsymbol{x} \in \mathbb{R}^n$$

Matrices

▶ A matrix is a 2-D array of numbers

$$\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}$$

▶ Example notation for type and shape
$\mathbf{A} \in \mathbb{R}^{m \times n}$

Tensors

- A tensor is an array of numbers that may have
  - a zero dimensions and be a scalar,
  - one dimension and be a vector,
  - two dimensions and be a matrix,
  - more dimensions …

Side note: One of the most popular frameworks for implementing deep machine learning models is called TensorFlow (https://www.tensorflow.org/).

Transpose matrix

$$(\boldsymbol{A}^T)_{i,j} = \boldsymbol{A}_{j,i}$$

$$\boldsymbol{A} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} \Rightarrow \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix}$$

The transpose matrix is a mirror image with regard to the main diagonal

$$(\boldsymbol{AB})^T = \boldsymbol{B}^T \boldsymbol{A}^T$$

- Identity matrix I_3

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- The identity matrices are neutral elements in matrix-matrix and matrix-vector multiplication, e.g.

$$\forall \boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{I}_n \boldsymbol{x} = \boldsymbol{x} \boldsymbol{I}_n = \boldsymbol{x}$$

Matrix (dot) product

$$C = AB$$

The matrices must be compatible: an $m \times n$ matrix is multiplied with an $n \times r$ matrix and as a result an $m \times r$ matrix is obtained

$$C_{i,j} = \sum_k A_{i,k} B_{k,j}$$

$$A \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} \times B \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{bmatrix} = C \begin{bmatrix} 4 \times 5 \end{bmatrix}$$

$$C_{2,5} = A_{2,1} B_{1,5} + A_{2,2} B_{2,5} + A_{2,3} B_{3,5} = 4 \cdot 5 + 5 \cdot 10 + 6 \cdot 15 = 160$$

Matrix (dot) product

- In general matrix multiplication is not commutative, i.e., most of the time $AB \neq BA$.
- Depending on the dimensions sometimes AB or BA are not possible.
- As a special case the matrix can be a (column or row) vector; an $m \times n$ matrix is multiplied with a $n \times 1$ vector to obtain a $m \times 1$ vector.

Systems of equations

- $Ax = b$ expands to

$$A_{1,:}x_1 = b_1 \tag{1}$$
$$A_{2,:}x_2 = b_2 \tag{2}$$
$$\ldots \tag{3}$$
$$A_{m,:}x_m = b_m \tag{4}$$

Solving systems of linear equations

- A linear system of equations can have
  - no solutions,
  - many solutions,
  - exactly one solution.
- Only one solution implies that multiplication by a matrix is an invertible operation.

Matrix inversion

▶ Matrix inverse is defined with

$$A^{-1}A = I_n$$

▶ A system of linear equations can be solved using inverse matrix

$$Ax = b$$

$$A^{-1}Ax = A^{-1}b$$

$$I_n x = A^{-1}b$$

$$x = A^{-1}b$$

▶ This is useful mostly for abstract analysis.

▶ From a numerical point of view there are much more efficient methods.

A matrix cannot be inverted if

- the number of rows and columns is not the same, or
- some rows and columns are "redundant" ("linearly dependent", "low rank").

Norms

- Norms are functions that measure how "large" a vector is.
- Similar to a distance between zero and the point represented by the vector
  - $f(\boldsymbol{x}) = 0 \Rightarrow \boldsymbol{x} = 0$
  - $f(\boldsymbol{x} + \boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y})$ (**the triangle inequality**)
  - $\forall \alpha \in \mathbb{R} : f(\alpha \boldsymbol{x}) = \alpha f(\boldsymbol{x})$

Norms

- L^p- norm

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

  - Most popular L^2-norm (for $p = 2$)
  - L_1-norm (for $p = 1$): $\|\boldsymbol{x}\|_1 = \sum_i |x_i|$
  - Max norm (for infinite p): $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$

Special vectors and matrices

- Unit vector $\|\boldsymbol{x}\|_n = 1$
- Symmetric matrix $\boldsymbol{A} = \boldsymbol{A}^T$
- Orthogonal matrix

$$\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I} = \boldsymbol{A}^T\boldsymbol{A}$$

- It follows that for orthogonal matrices $\boldsymbol{A}^T = \boldsymbol{A}^{-1}$

Eigendecomposition

- Eigenvector and eigenvalue

$$\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x}$$

- Eigendomdecomposition of a matrix

$$\boldsymbol{A} = \boldsymbol{V}\mathrm{diag}(\lambda)\boldsymbol{A}^{-1}$$

where $\mathrm{diag}(\lambda)$ is a diagonal matrix having the (scalar) eigenvalues λ as diagonal elements.
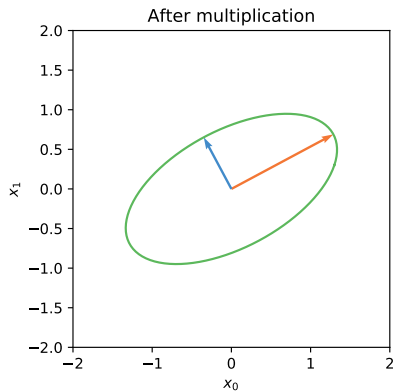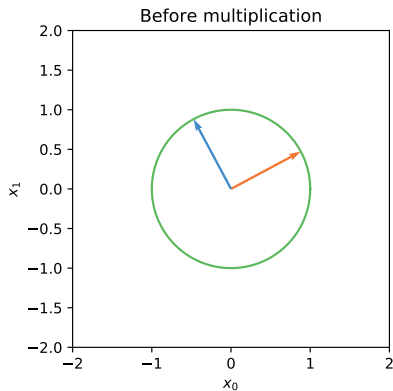
Eigendecomposition

- Every real symmetric matrix has a real orthogonal eigendecomposition

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

  where $\boldsymbol{Q}$ is an orthogonal matrix composed of eigenvectors of $\boldsymbol{A}$ and $\boldsymbol{\Lambda}$ is a diagonal matrix.

- The eigenvalue Λ_{ii} is associated with the eigenvector in column i of $\boldsymbol{Q}$, denoted as $\boldsymbol{Q}_{:,i}$.

- We can think of $\boldsymbol{A}$ as scaling space by factor λ_i in the direction of its corresponding eigenvector $v^{(i)}$ (represented by $\boldsymbol{Q}_{:,i}$).

Effect of eigenvalues

Eigendecomposition

- From the eigendecomposition we learn useful properties of the matrix.
- The eigendecomposition of a real symmetric matrix is used in optimization of quadratic expressions of the form $f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}$ under the constraint $\|\boldsymbol{x}\|_2 = 1$.
- For instance, if $\boldsymbol{x} = \boldsymbol{v}^{(i)}$, then $f(\boldsymbol{x}) = \lambda_i$, when $\boldsymbol{v}^{(i)}$ is an eigenvector of A and λ_i is its corresponding eigenvalue.
- The maximal (minimal) value of f within the constraint region is equal to the maximal (minimal) eigenvalue.

Singular value decomposition

- Similar to eigenvalue decomposition
- More general: matrix need not be square

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$$

- $\boldsymbol{U}$ and $\boldsymbol{V}$ are square matrices and are both orthogonal, $\boldsymbol{D}$ is diagonal.
- The diagonal elements of D are called **singular values** of matrix $\boldsymbol{A}$; the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ are **left-singular** and **right-singular vectors** of $\boldsymbol{A}$, respectively.

- Matrix inversion is not defined on matrices that are no square.
- The **Moore-Penrose pseudoinverse** is defined as

$$\boldsymbol{A}^+ = \lim_{\alpha \searrow 0} (\boldsymbol{A}^T \boldsymbol{A} + \alpha \boldsymbol{I})^{-1} \boldsymbol{A}^T$$

Moore-Penrose pseudoinverse

Now we can consider

$$x = A^+ y$$

- If the equation has
  - exactly one solution: this is the same as inverse,
  - no solution: gives the solution with the smallest error, $\|Ax - y\|_2$
  - many solutions: gives the solution with the smallest norm of x.

Computing the pseudoinverse

- Efficient implementations are based on the formula allowed by the singular decomposition

$$A^+ = VD^+U^T$$

- U, D, V are from the singular value decomposition of A.
- The pseudoinverse D^+ of D is obtained by taking the reciprocal non-zero elements and after that taking the transpose of the resulting matrix.

Trace

- A **trace** of a matrix is defined as

$$Tr(\boldsymbol{A}) = \sum_i \boldsymbol{A}_{i,i}$$

- Expressions in terms of the trace operators allow to exploit many useful identities, e.g.

$$Tr(\boldsymbol{ABC}) = Tr(\boldsymbol{BCA}) = Tr(\boldsymbol{CAB})$$

Gradient-based optimization

Materials:

- Chapters I.4 and I.5 from Goodfellow et al., *Deep Learning*
- Kolter et al., "Linear Algebra Review and Reference"

Gradient

- Let $f : \mathbb{R}^{m \times n} \mapsto \mathbb{R}$ be a function that takes $m \times n$ matrix $\boldsymbol{A}$ as input and returns a real number (scalar).

- A **gradient** of f with respect to A is the matrix

$$\nabla_{\boldsymbol{A}} f(\boldsymbol{A}) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \frac{\partial f}{\partial A_{12}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \frac{\partial f}{\partial A_{21}} & \frac{\partial f}{\partial A_{22}} & \cdots & \frac{\partial f}{\partial A_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \frac{\partial f}{\partial A_{m2}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix}$$

- i.e. an $m \times n$ matrix with

$$(\nabla_{\boldsymbol{A}} f(\boldsymbol{A}))_{ij} = \frac{\partial f}{\partial A_{ij}}$$

- The size of the gradient of $\boldsymbol{A}$ is the same as the size of A.

Gradient

- In the special case when A is a vector we obtain the (possibly more familiar) gradient

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_m} \end{bmatrix}$$

- In general to define a gradient we require that the function returns a **real** value.

Jacobian

▶ The Jacobian J_f is a generalization of the gradient for vector valued functions.

▶ Let $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ be a function that takes n-dimensional vector x as input and returns a m-dimensional vector as an output.

▶ The Jacobian J_f is defined as

$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

▶ Note that for the special case of a scalar-valued function, the Jacobian is the transpose of the gradient.

Optimization

- Most machine learning methods involve some kind of optimization.
    - One exception is the k-Nearest neighbour classifier introduced later.
- Optimization means minimizing or maximizing some function $f(\boldsymbol{x})$, i.e. finding the values of $\boldsymbol{x}$ for which $f(\boldsymbol{x})$ has a minimum or a maximum.
- Notation: $\boldsymbol{x}^* = \operatorname{argmin} f(\boldsymbol{x})$

- The derivative tells us how to change x in order to make a small improvement of $f(x)$.
- Therefore, derivatives can be useful in optimization.

Two simple machine learning models

Materials:

- Chapter 2.3 from Friedman et al., *The Elements of Statistical Learning*

Some notations

- We denote an input variable with the symbol x (scalar) or $\boldsymbol{x}$ (vector).

- The i-th component of a vector input $\boldsymbol{x}$ is denoted as x_i.

- Quantitative (numerical) outputs are denoted with y.

- Qualitative outputs are denoted with g (from group) and take values from a set $\mathcal{G}$.

- Matrices are denoted with bold and uppercase letters $\boldsymbol{X}$ for instance, a set of N input p-vectors $\boldsymbol{x}_i$ $(1 \leq i \leq N)$ is "packed" in a $N \times p$ input matrix $\boldsymbol{X}$.

- Since by default vectors are assumed to be column vectors, the rows of $\boldsymbol{X}$ are the transposes $\boldsymbol{x}_i^T$.

The learning task

- Given a value of the input vector $\boldsymbol{x}$ make a good prediction of the output y, denoted as $\hat{y}$.
- Both y and $\hat{y}$ should take values from the same numerical set.
- Similarly, g and $\hat{g}$ should both take values from the same set $\mathcal{G}$.
- We suppose that we have available a set of measurements $(\boldsymbol{x}_i, y_i)$ or $(\boldsymbol{x}_i, g_i)$ $(1 \leq i \leq N)$ called **training data** (in matrix form: $(\boldsymbol{X}, \boldsymbol{y})$ and/or $(\boldsymbol{X}, \boldsymbol{g})$).
- Our task is to construct a prediction rule based on the training data.

The learning task

Example:

- **Variable values**: Let g (and therefore also $\hat{g}$) be two valued (categorical), e.g. $\mathcal{G} = \{\text{BLUE}, \text{ORANGE}\}$.

- **Encoding of gs with ys**: Then each class can be encoded binary, i.e., with $y \in \{0, 1\}$, e.g., BLUE and ORANGE, would correspond to 0 and 1, respectively.

- **Predicted output values**: $\hat{y}$ ranges over the interval $[-\infty, +\infty]$ (of which $\{0, 1\}$ is a subset).

- **Prediction rule**: $\hat{g}$ is assigned a (class label) BLUE if $\hat{y} < 0.5$ and ORANGE, otherwise.

- Linear model fit
  - strong assumptions about the structure of the decision boundary
- k-nearest neighbours
  - weak assumptions about the structure of the decision boundary

Linear model fit by least squares

- Despite relative simplicity one of the most important statistical tools
- Input vector $\mathbf{x}^T = (x_1, x_2, \ldots, x_p)$
- Output y predicted using the model

$$\hat{y} = \hat{w}_0 + \sum_{j=1}^{p} x_j \hat{w}_j$$

- $\hat{w}_i$ ($0 \leq i \leq p$) are the parameters of the linear model
- In vector form

$$\hat{y} = \hat{\mathbf{w}}^T \mathbf{x} = \mathbf{x}^T \hat{\mathbf{w}}$$

using the fact that the scalar (inner) product of two vectors is a commutative operation.

- We assume that w_0 is in $\mathbf{w}$ and 1 is included in $\mathbf{x}$.
- $\hat{y}$ is a scalar, but in general can be a k-vector $\hat{\mathbf{y}}$, in which case $\mathbf{w}$ becomes a $p \times k$ matrix of coefficients.

Linear model fit by least squares

Some hyper(space) terminology:

- Points $x, \hat{y}$ form a **hyperplane** in the $(p+1)$-dimensional input-output hyperspace.

- If x is extended with constant 1 then the hyperplane includes the origin and it forms a **subspace**.

- If 1 is not included then the hyperplane is an **affine** set and it cuts the y-axis at the point $(\mathbf{0}, \hat{w}_0)$, where the vector $\mathbf{0}$ has all x_i coordinates equal to 0.

- Reminder: from now on we assume that 1 is included in x and $\hat{w}_0$ in $\hat{\boldsymbol{w}}/$

- The function $f(x) = \boldsymbol{w}^T x$ defined on the p-dimensional (input) space is a **linear** function (we omit the hats over the ws since now we consider them as free variables).

- The gradient $\nabla f(x)$ is a vector pointing along the direction of maximal change.

Linear model fit by least squares

- There are many ways to fit a linear model to a training dataset.
- **Least squares** method
    - We need to find coefficients $\hat{w}_i$ which minimize the error estimated with the **residual sum of squares**

$$\text{RSS}(\boldsymbol{w}) = \sum_{i=1}^{N}(y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2$$

  assuming N input-output pairs.
- $\text{RSS}(\boldsymbol{w})$ is a quadratic function.
- A minimum always exists though not necessarily a unique one.

Linear model fit by least squares

- We look for the solution $\hat{\boldsymbol{w}}$ using the matrix notation:
- $\boldsymbol{y} = [y_1, y_2, \ldots, y_N]^T$ is the vector formed from the N output vectors and $\boldsymbol{X}$ is an $N \times p$ matrix

$$\text{RSS}(\boldsymbol{w}) = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

- To find the minimum we differentiate with respect to $\boldsymbol{w}$ which gives

$$(-\boldsymbol{X})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) + (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T (-\boldsymbol{X})$$

using the rule $(\boldsymbol{A}\boldsymbol{B})^T = \boldsymbol{B}^T \boldsymbol{A}^T$ this is equivalent to

$$-2\boldsymbol{X}^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

Linear model fit by least squares

- To find the minimum our derivative must be $\mathbf{0}$, hence:

$$\mathbf{X}^T(\mathbf{y} - \mathbf{Xw}) = \mathbf{0}$$

$$\mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{Xw} = \mathbf{0}$$

$$\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{Xw}$$

- If $\mathbf{X}^T\mathbf{X}$ is non-singular there exists a unique solution given by

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Question: Why not simply $\mathbf{y} - \mathbf{Xw} = \mathbf{0} \rightarrow \mathbf{y} = \mathbf{Xw} \rightarrow \hat{\mathbf{w}} = \mathbf{X}^{-1}\mathbf{y}$?

- For each input $\boldsymbol{x}_i$ there corresponds the fitted output

$$\hat{y}_i = \hat{y}_i(\boldsymbol{x}_i) = \hat{\boldsymbol{w}}^T \boldsymbol{x}_i$$

  .

- This is called "making a prediction" for $\boldsymbol{x}_i$.
- The entire fitted surface (hyperplane) is fully characterized by the parameter vector $\hat{\boldsymbol{w}}$.
- After fitting the model, we can "discard" the training dataset.

Example: Linear model fit by least squares

- Scatter plot of training data on a pair of inputs x_1 and x_2
- Output class variable g has two values BLUE and ORANGE.
- Linear regression model fitted with the response variable y coded as 0 for BLUE and 1 for ORANGE.
- Fitted values $\hat{y}$ converted to a fitted class variable $\hat{g}$ as

$$\hat{g} = \begin{cases} \text{BLUE} & \text{if } \hat{y} \leq 0.5 \\ \text{ORANGE} & \text{if } \hat{y} > 0.5 \end{cases}$$

Example: Linear model fit by least squares
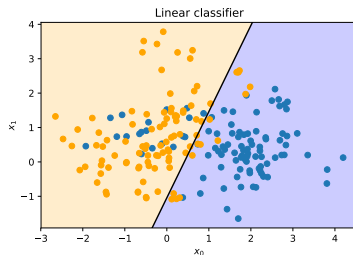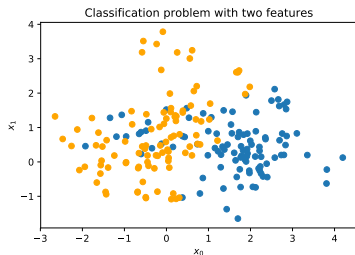
Example: Linear model fit by least squares

- Two classes separated in the plane ($\mathbb{R}^2$) by the decision boundary $\{x : w^T x = 0.5\}$
- $\{x : w^T x < 0.5\}$ set of BLUE points
- $\{x : w^T x \geq 0.5\}$ set of ORANGE points

Example: Linear model fit by least squares

- Wrong classifications on both sides of the boundary
- Are the errors caused by the model or are they unavoidable?
- Two possible scenarios
  - **Scenario 1**: data generated from bivariate Gaussian distribution
  - **Scenario 2**: data generated from 10 Gaussian distributions; the means of these distributions are also distributed as Gaussian
- In Scenario 1 the linear boundary is the best we can do since the overlap is inevitable.
- In Scenario 2 the linear boundary is unlikely to be optimal (in fact the boundary is non-linear and disjoint).

Nearest-neighbours model

- In nearest-neignbour methods $\hat{y}(\boldsymbol{x})$ is determined based on the inputs (points) in the training set $\mathcal{T}$ which are "closest" to the input $\boldsymbol{x}$.

- k-nearest neighbour fit is defined as

$$\hat{y}(\boldsymbol{x}) = \frac{1}{k} \sum_{\boldsymbol{x}_i \in N_k(\boldsymbol{x})} y_i$$

where $N_k(\boldsymbol{x})$ is the neighbourhood of $\boldsymbol{x}$ consisting of the k "closest" points to $\boldsymbol{x}$.

- "Closeness" requires a definition of **metrics**.

- For the moment we assume Euclidian distance (each $\boldsymbol{x}$ is a point in the hyperspace).

- An average of the classes of the k closest points

- We use the same training data as in the linear model example.
- New borderline between the classes generated with 15-nearest-neighbour model
- Since ORANGE is encoded as 1 $\hat{y}$ is the proportion of ORANGE points in the 15-neighbourhood
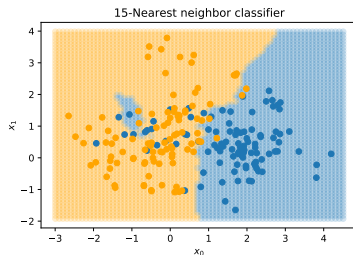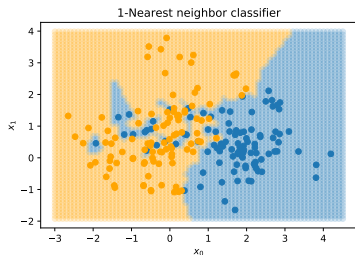- Class ORANGE assigned to $\boldsymbol{x}$ if $\hat{y}(\boldsymbol{x}) > 0.5$ (majority is ORANGE)

15-Nearest neighbour classifier
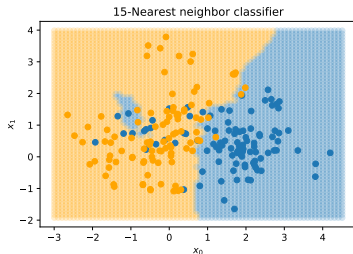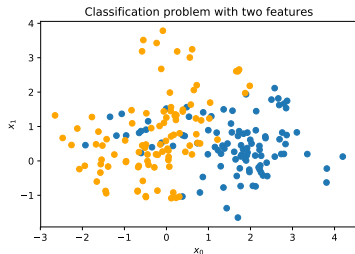
Linear classifier vs. 15-Nearest neighbour
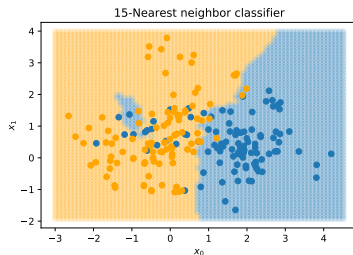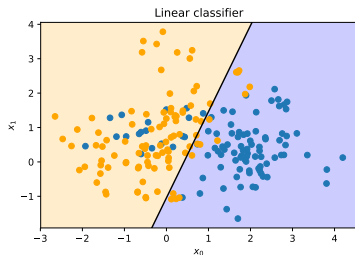
1-Nearest neighbour vs. 15-Nearest neighbour

Comparison of the techniques

- 15-NN seems to work better than the linear classifier since fewer points are missclassified.
- On the other hand, **none** of the points in the 1-NN case was misclassified!?
- Actually with the 1-NN method the error on **training data** is always 0.
- An independent test set needed to obtain a better comparison of the methods.

Comparison of techniques

- At first sight it looks like k-NN has only one parameter, k versus p parameters (number of weights w_i) of the linear model.
- The **effective** number of parameters of k-NN is N/k which is in general bigger than p (N is the size of the training set).
- For instance, assume non-overlapping neighbourhoods
  - There will be N/k neighbourhoods.
  - To each neighbourhood there correspond one parameter (the mean of the elements of the neighbourhood).

Probability theory

Materials:

- Chapter I.3 from Goodfellow et al., *Deep Learning*

Probability theory

- Probability theory is a mathematical framework for dealing with uncertainty, i.e., modeling and analyzing uncertain events and statements
- In AI probability theory is used in two major ways:
  - To design AI systems, i.e., derive models and expressions and the corresponding algorithms.
  - To analyze the behaviour of the AI systems.

Probability theory

- A **random variable** is a variable that can take values randomly.
- We will denote random variables with plain (ordinary text) typeface and their values with standard math typeface for example, if the random variable is denoted as x its values can be x_1 and x_2.
- A vector-valued random variable is denoted with bold typeface, e.g. $\mathbf{x}$.
- On its own a random variable just denotes the set of its possible values; to get its full meaning in needs to be coupled with a distribution.

- There are two types of random variables: **discrete** and **continuous**.
- Consequently there are two ways to describe probability distributions: **probability mass functions** and **probability density functions**.

Probability mass function

▶ The domain of a probability mass function P is the set of all possible states of the random variable x.

▶ $\forall x \in x : 0 \le P((x)) \le 1$
  ▶ An impossible event has probability 0 and no state can be less probable than that.
  ▶ An event that is guaranteed to happen has probability 1 and no state can have a greater chance of occurring.

▶ $\sum_{x \in x} P(x) = 1$
  ▶ We say that x is **normalized**.

▶ Example: Uniform distribution: $P(x = x_i) = \frac{1}{k}$.

Probability density function

- The domain of the probability density function p must be the set of all possible states of x.
- $\forall x \in x : p(x) \geq 0$.
-
$$\int p(x)dx = 1$$

- Example: uniform distribution $u(x; a, b) = \frac{1}{b-a}$, for $x \in [a, b]$

Conditional probability

- **Conditional probability** is the probability of some event provided that some other event has happened.
- Given two random variables x and y, the conditional probability that y has value y provided that we know that x has value x is given by

$$P(\text{y} = y \mid \text{x} = x) = \frac{P(\text{x,y})}{P(\text{x} = x)}$$

- Another way to see this formula is

$$P(\text{x,y}) = P(\text{x} = x)P(\text{y} = y \mid \text{x} = x)$$

i.e., the probability of x and y occurring together is equal to the probability of occurrence of x times the probability of y occurring provided x has occurred.

Expectation

- The **expectation** or **expected** value of a function $f(x)$ with respect to a probability distribution $P(x)$ is the average value of f over all values x assuming they are drawn from P

-
$$\mathbb{E}_{x \sim P}[f(x)] = \sum_x P(x)f(x)$$

-
$$\mathbb{E}_{x \sim P}[f(x)] = \int p(x)f(x)dx$$

- Linarity of expectations:

$$\mathbb{E}_x[\alpha f(x) + \beta g(x)] = \alpha \mathbb{E}_x[f(x)] + \beta \mathbb{E}_x[g(x)]$$

Variance and covariance

- The **variance** gives a measure of variation of the values of a random variable x

$$\text{Var}(f(x)) = \mathbb{E}[(f(x) - E[f(x)])^2]$$

Square root of the variance is called **standard deviation**.

- The **covariance** is a measure of linear relation as well as scale between

$$\text{Cov}(f(x), g(x)) = \mathbb{E}[f(x) - E[(f(x)])(g(x) - E[g(x)])]$$

Covariance matrix

- The **covariance matrix** of a random vector $\boldsymbol{x} \in \mathbb{R}^n$ is a $n \times n$ matrix with elements

$$\text{Cov}(\boldsymbol{x})_{i,j} = \text{Cov}(x_i, x_j)$$

- The diagonal elements of the matrix give the variance

$$\text{Cov}(x_i, x_i) = \text{Var}(x_i)$$

Bernouli Distribution

- A distribution over a single binary random variable
- Controlled by a single parameter $\phi \in [0, 1]$ which corresponds to the probability of the random variable taking the value 1
- Properties:

$$P(\mathrm{x}) = 1) = \phi$$
$$P(\mathrm{x} = 0) = 1 - \phi$$
$$P(\mathrm{x} = x) = \phi^x (1 - \phi)^{1-x}$$
$$\mathbb{E}_{\mathrm{x}}[\mathrm{x}] = \phi$$
$$Var(\mathrm{x}) = \phi(1 - \phi)$$

Gaussian distribution

- The most commonly used distribution, also called **normal distribution**.
- Controlled by two parameters $\mu \in \mathbb{R}$ (the **mean**) and $\sigma \in (0, \infty)$, (the **standard deviation**)

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

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sqrt{\frac{1}{(2\pi)^n \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right)$$

Gaussian distribution
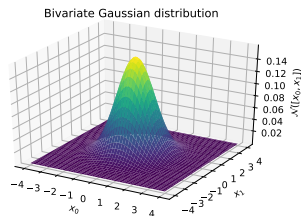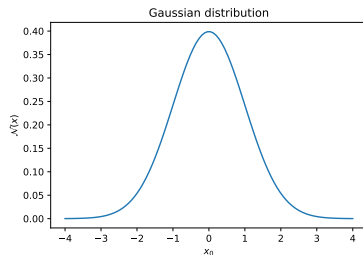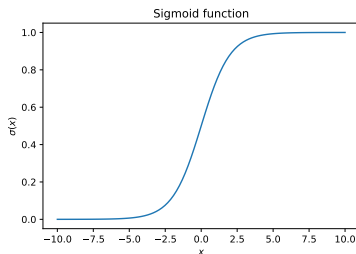
Logistic sigmoid

▶ A useful function that we are going to consider

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

▶ The Logistic (sigmoid) function is commonly used to parametrize Bernoulli distributions.

Sigmoid function

Bayes' rule

- Suppose know $P(y \mid x)$, but we actually need $P(x \mid y)$. If we know $P(x)$ then we can compute

$$P(x \mid y) = \frac{P(y \mid x)P(x)}{P(y)}$$

  Although it appears in the formula prior knowledge $P(y$ is not needed since usually it can be computed as $\sum_x P(y \mid x)P(x)$

- It can be straightforwardly derived from the conditional probability formula.

- It could have be named also after Laplace who independently found it, generalized it, and introduced it in practice.

Acknowledgements

References

📄 Friedman, J., T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning*. Springer series in statistics New York, 2001.

📄 Goodfellow, I., Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

📄 Kolter, Z. and C. Do. "Linear Algebra Review and Reference". In: (2015).