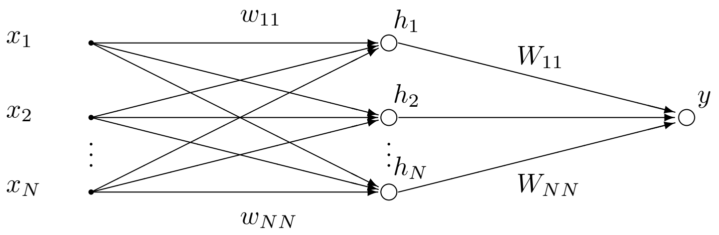
# Neural Networks and Learning Systems

## TBMI26, Exercise Collection

2017



# Neural Networks and Learning Systems
# Exercise Collection

## Contents

## Exercises

# Exercises

# 1. Introduction and k-Nearest Neighbors

## 1.1. Learning types

Machine learning can be divided into three main types. Which?

## 1.2. Feature

What is a *feature* in the context of machine learning?

## 1.3. Classification function

a) Mathematically, a classifier can be described as a function $f(\mathbf{x}; w_1, w_2, \ldots, w_n) \rightarrow \Omega$. What is $\mathbf{x}$, $w_1, w_2, \ldots, w_n$ and $\Omega$?

b) What is the difference between regression and classification?

c) How is learning of a classifier for a pattern recognition problem performed?

## 1.4. Generalization & overtraining

a) It is important that a learning method is able to *generalize*. What does this mean?

b) In supervised learning, we are usually training a classifier to minimize the error on training data. But what we really want is a low generalization error. How can we estimate the generalization error of a classifier?

c) Why is it important to divide the data set into a training set and a test set when training a learning system?

d) How can you notice if a supervised machine learning algorithm has overtrained?

## 1.5. Confusion matrices

You evaluate a classifier you have just trained and find the following *confusion matrix*. What is the classification accuracy?

| | | Classified label | |
|---|---|---|---|
| | | Class 1 | Class 2 |
| Actual label | Class 1 | 80 | 20 |
| | Class 2 | 30 | 90 |

## 1.6. Cross-validation

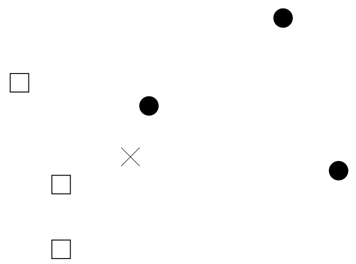
You have 900 labeled training samples and want to evaluate how well different supervised classification algorithms perform for this data. Explain/sketch how you would do this with *3-fold cross-validation*.

## 1.7. k-Nearest Neighbors

Mention one advantage and one disadvantage of the k-nearest neighbor classifier.

## 1.8. k-Nearest Neighbors

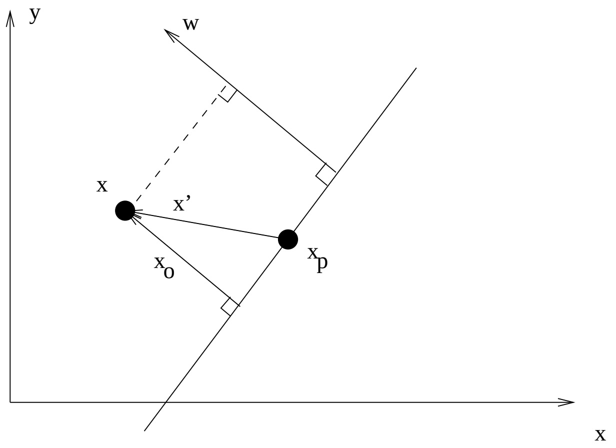
Which class does X belong to using a kNN-classifier using $k = 1$ and $k = 3$? why? How can we handle $k = 2$?



## 2. Linear classifiers

### 2.1. Linear algebra reminder

This exercise should work as a small repetition of planes, projections and the scalar products. Define the plane equation in vector format using the normal $\mathbf{w}$, then define how to calculate the distance from the plane to any point $\mathbf{x_0}$.



### 2.2. The perceptron

a) Write the mathematical expression for the perceptron.

b) What is the purpose of the "bias weight" in a perceptron?

c) What happens if the input to the bias weight in a perceptron is set to the constant value 2?

d) Construct a classification example with two classes which are linearly separable, but which requires a non-zero bias-weight for a perfect classification.

### 2.3. Batch vs. online learning

What is the difference between *batch learning* and *online learning*?

### 2.4. Partial derivatives of vectors

Find the derivative:

a) $\frac{\partial y}{\partial \mathbf{w}}$ if $y = \mathbf{w}^T\mathbf{x}$.

b) $\frac{\partial \mathbf{y}}{\partial \mathbf{W}}$ if $\mathbf{y} = \mathbf{W}\mathbf{x}$.

c) $\frac{\partial y}{\partial \mathbf{x}}$ if $y = \mathbf{x}^T\mathbf{W}\mathbf{x}$

d) $\frac{\partial y}{\partial \mathbf{w}}$ if $y = ||\mathbf{w}||^4$

e) $\frac{\partial \mathbf{y}}{\partial \mathbf{W}}$ if $\mathbf{y} = \mathbf{A}\mathbf{g}(\mathbf{s})$ where $\mathbf{g}(\mathbf{s}) = \exp(\mathbf{W}\mathbf{x})$

### 2.5. Optimization

Minimize the cost function $f$ with respect to $\mathbf{x}$:

a) if $f(\mathbf{x}) = 3||\mathbf{x}||^2 + \begin{bmatrix}1 & 1\end{bmatrix}\mathbf{x} - 4$

b) with the gradient descent method for (a) when $\eta = 0.05$ and $x_{old} = \begin{bmatrix}1\\1\end{bmatrix}$. Only two iterations need to be described.

### 2.6. Gradient descent step length

In gradient descent optimization, for example when training a neural network, what may be the effect of

- A too long step length (1p)
- A too short step length (1p)

Illustrate with figures!

### 2.7. Cost function

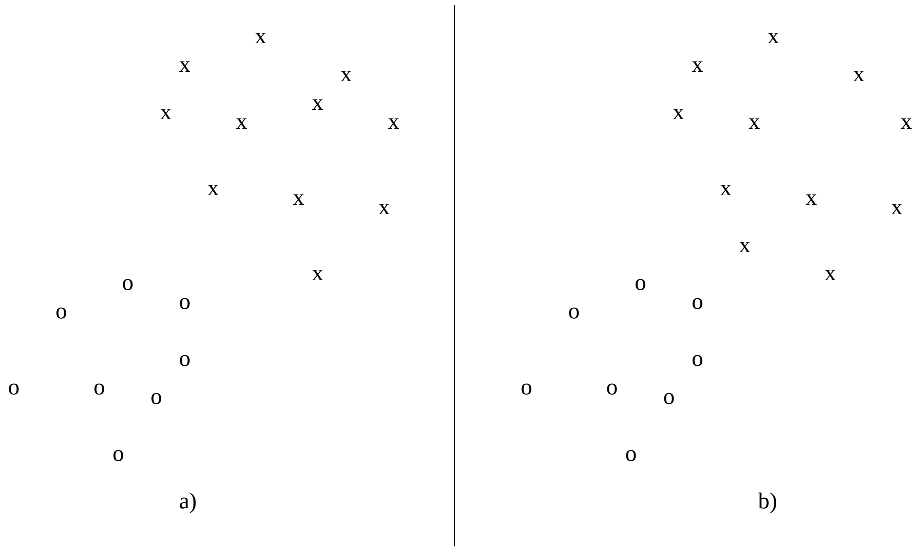
In supervised learning, why is it a problem to train a classifier by minimizing the number of false classifications using gradient descent? That is, to minimize $\sum_{k=1}^{N} I(z_k \neq y_k)$ where $z_k$ is the output from the classifier, $y_k$ the correct label, $I(z_k \neq y_k)$ is equal to 1 if $z_k \neq y_k$ and 0 otherwise, and $N$ is the number of training examples.

### 2.8. SVM

What is the *maximum margin* principle that is used for example in SVM?

### 2.9. SVM

Draw the optimal hyper plane (decision boundary) and mark the support vectors in the following two cases:



a)

b)

### 2.10. SVM

In SVM, an entity is minimized under the following constraint:

$$d_i(\mathbf{w}^T \mathbf{x}_i + w_0) \geq 1.$$

What is being minimized, and for which $\mathbf{x}_i$ are the constraint full-filled with equality?

### 2.11. SVM

Are SVM particularly useful when the training data set is rather small or when it is very large? Motivate!

# 3. Neural Networks

## 3.1. XOR

a) Sketch the so-called “xor-problem”.

b) Why cannot the “xor-problem” be solved by a one-layer perceptron?

## 3.2. Cover’s theorem

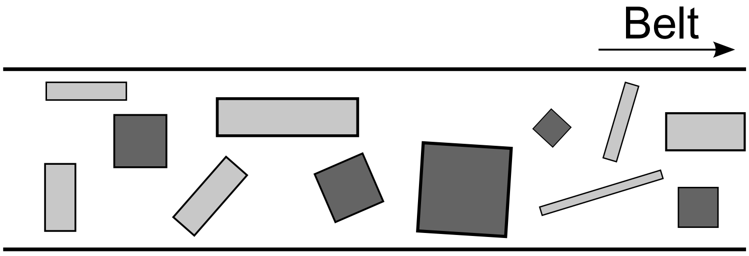
What is the meaning of Cover’s theorem in words?

## 3.3. Nonlinear classification

In an industrial application, two types of objects with square and rectangular (non-square) shapes respectively are transported on a belt. The size of the objects varies. A camera observes the objects and you have access to a function that measures the lengths $x_1$ and $x_2$ of two neighboring sides of each object. There is a small amount of noise in the measurement, so that $x_1$ and $x_2$ will be slightly different also for square objects.

Using the above measurements, your task is to classify each object as rectangular or square:

- Suggest features and sketch how the features distribute in the feature space for both classes (only the approximate structure is asked for!).
- Suggest a suitable classifier based on this.



*Note: The image is just for illustration, you are* **not** *required to measure in it!*

## 3.4. Prediction using a neural network

Describe briefly how a neural network could be trained to predict the temperature for the next day.

## 3.5. Activation functions

a) Is it appropriate to train the network with y = +/- 1 if we, for example, use tanh() as activation function?

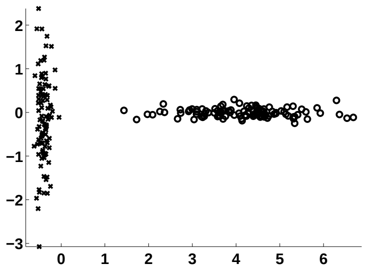
b) Would we gain anything by using a nonlinear activation function in the output layer, instead of a linear function?

c) Draw the approximate decision boundaries you would get if you trained a linear classifier using the error functions $\epsilon_1 = \sum (y_i - \mathbf{w}^T\mathbf{x}_i)^2$ and $\epsilon_2 = \sum (y_i - \tanh(\mathbf{w}^T\mathbf{x}_i))^2$, $\mathbf{x}_i$ are the feature vectors, $y_i = \pm 1$ the class labels, and $\mathbf{w}$ the boundary parameters. Explain how the difference in error functions affects the location of the boundary.

## 3.6. Neural network training

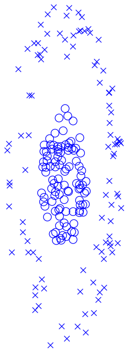
a) Describe (by an equation) how a momentum term is used in gradient descent, for example when training a neural network.

b) How can the problem of overfitting be avoided when training a neural network?



### 3.7. Metrics

Distance can be measured in a number of ways and is of great importance in classification. The most intuitive distance from a class with mean $\mu$ is the Euclidian distance $d_e = \sqrt{(x-\mu)^T(x-\mu)}$. Two other common distances are RBF $d_r = e^{-d_e^2/\sigma}$ and the Mahalanobis distance $d_m = \sqrt{(x-\mu)^T C^{-1}(x-\mu)}$, $C$ is the covariance matrix of the class. Which of these three metrics is most suitable for classification of the data in the figure below, and why?



### 3.8. Backpropagation derivation

a) Derive the batch training error gradient for a two-layer neural network with arbitrary number on neurons, classes an inputs. The hidden layer shall have an non-linear activation function while the output layer shall have a linear activation function.

   Hint: First do the derivation component-wise for the output layer then component-wise for the hidden layer. Last but not least transfer the result to matrix form.

   Also, derive the corresponding update rules for the hidden and output layers.

   You will implement this in the back-prop assignment.

b) Rewrite the previous result in on-line training format.

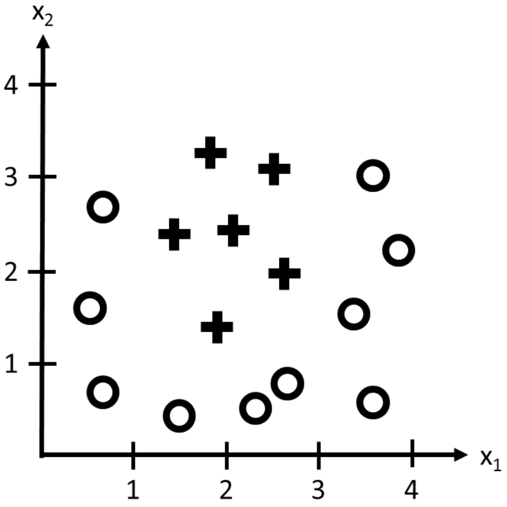
# 4. Ensemble Learning & Boosting

## 4.1. Decision tree & CART

a) What is a *decision tree / CART*?

b) What is the difference between a classification tree and a regression tree?

c) The figure below shows a feature space with two classes. Draw a Classification and Regression Tree (CART) that classifies the two classes and draw the classification boundaries in the figure.



## 4.2. Decision stump

a) What is a *decision stump*?

b) Which parameters are we optimizing? (Assume one dimension for now)

c) What is the cost function we want to minimize?

d) Why is this cost function always $\leq 0.5$ after optimization?

e) How many thresholds do we need to test?

## 4.3. Ensemble learning

What is the basic idea of Ensemble learning?

## 4.4. Boosting vs Bagging.

a) What is *Bagging*?

b) What is *Boosting*?

## 4.5. Weak classifier

a) What is a *weak classifier*?

b) Describe one weak classifier.

c) Construct/sketch a very simple ”Toy” classification task with two classes.

d) How will one instance of your weak classifier solve this classification task?

## 4.6. Boosting algorithm

Describe/sketch the flow of a general Boosting algorithm using pseudo-code. Omit details and equations but include the logic.

### 4.7. AdaBoost

You have the following data:

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \qquad \mathbf{Y}_a = \begin{bmatrix} 1 & 1 & -1 & 1 \end{bmatrix} \qquad \mathbf{Y}_b = \begin{bmatrix} 1 & -1 & -1 & 1 \end{bmatrix}$$

where $\mathbf{X}$ contains four 2d-samples (one per column), and $\mathbf{Y}_a \& \mathbf{Y}_b$ contain classification labels for the corresponding samples.

a) Perform the first AdaBoost iteration on the data $\mathbf{X}$ using the labels $\mathbf{Y}_a$. Sketch the classification problem. Use 'decision stumps' as weak classifiers. Does AdaBoost work well in this setting? (2p)

b) Perform the first AdaBoost iteration on the data $\mathbf{X}$ using the labels $\mathbf{Y}_b$. Sketch the classification problem. Use 'decision stumps' as weak classifiers. Does AdaBoost work well in this setting? (2p)

c) Why is outliers a problem for the standard AdaBoost method? (1p)

*Hint:* The standard way of updating the weights in the standard AdaBoost method is $d_{t+1}(i) \propto d_t(i)e^{-\alpha_t y_i h_t(\mathbf{x})}$, where $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$. *Note:* This was a question worth 5 points on the exam from 2012-03-22.

## 5. Deep Learning

Material will be available at the class.

# 6. Unsupervised Learning - Clustering

## 6.1. Cluster belonging

What is meant by *hard clustering* and *soft/fuzzy clustering* respectively?

## 6.2. Clustering algorithms

a) What is the difference between $k$-means clustering and Mixture of Gaussian clustering?

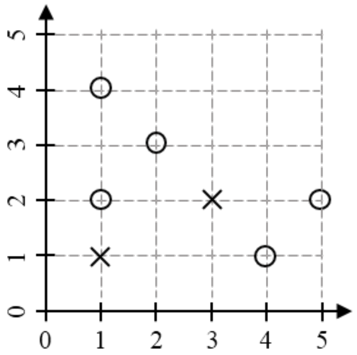
b) What is the optimization algorithm called that is used in k-Means Clustering and Mixture of Gaussian clustering.

## 6.3. The parameter $k$

What is determined by the parameter *k* in the *k*-Nearest Neighbors and *k*-Means algorithms respectively?

## 6.4. $k$-means algorithm

Perform one iteration of the k-means algorithm on the data plotted below. Circles (o) are input data and crosses (x) are the current prototype vectors. How many more iterations must be made before the algorithm converges?



## 6.5. Multidimensional Gaussian function

a) What is the density function of the multidimensional Gaussian (normal) distribution?

b) Given a mean vector of $\mathbf{0}$ and covariance matrix $\begin{pmatrix} 3 & -1 \\ -1 & 3 \end{pmatrix}$, calculate the density function for $\mathbf{x_1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $\mathbf{x_2} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ and $\mathbf{x_3} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

## 6.6. $k$-means & Mixture of Gaussians

Assume the following six 2d-samples (written as columns): $\mathbf{X} = \begin{bmatrix} -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & 0 & 1 & -1 & 0 & 1 \end{bmatrix}$.

Assume we have two prototypes (k=2): $\mathbf{p}_1 = \begin{bmatrix} -0.5 \\ 0 \end{bmatrix}$ and $\mathbf{p}_2 = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}$.

a) Draw a sketch of the samples and prototypes.

b) Perform two iterations of k-Means.

c) Perform two iterations of Mixture of Gaussians (MoG). Assume initial $\mathbf{C_1} = \mathbf{C_2} = \mathbf{I}$.

d) This exercise is probably more fun in Matlab...

## 7. Unsupervised Learning - Dimensionality reduction

### 7.1. Statistical concepts: 1-dimensional

Give the expression for the following statistical concepts (one dimensional case):

a) Covariance (according to the definition, between the stochastic variables $X$ and $Y$ using the expectation operator.)

b) Covariance estimation (if you have N samples $x_i$ and $y_i$ respectively)

c) Variance (according to the definition, for the stochastic variables $X$ using the expectation operator.)

d) Variance (expressed as a covariance)

e) Variance estimation (if you have N samples $x_i$)

f) Correlation (expressed in terms of covariance and variance)

### 7.2. Statistical concepts: multi-dimensional

Give the expression for the following statistical concepts (multi dimensional case):

a) Covariance (according to the definition, for the stochastic variable $\mathbf{X}$ using the expectation operator.)

b) Covariance matrix estimation (if you have N samples $\mathbf{x_i}$)

c) Cross-Covariance (according to the definition, between the stochastic variables $\mathbf{X}$ and $\mathbf{Y}$ using the expectation operator.)

### 7.3. Covariance matrix interpretation

a) You have a covariance matrix $\mathbf{C}$ describing a set of three dimensional signal samples $\mathbf{x_i}$. Identify the elements of the matrix in terms of variances and covariances of the signal components. Symmetries?

b) You have the same matrix as above. How would you change it to get the correlation matrix?

c) You have calculated the covariance and correlation matrix $\text{Cov}(\mathbf{x})$ and $\text{Corr}(\mathbf{x})$ respectively. Describe $\text{Cov}(5\mathbf{x})$ and $\text{Corr}(5\mathbf{x})$.

d) Assume a (time varying) vector with signals $\mathbf{z}(t)$. If the covariance matrix $\mathbf{C}_{zz}$ is diagonal, what does this mean?

### 7.4. Covariance matrix estimation

Find the mean value $\mathbf{m}$ and the covariance matrix $\mathbf{C}$ which describes $\mathbf{x}$ when we have the data samples $\mathbf{x}_1 = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$, $\mathbf{x}_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ and $\mathbf{x}_3 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$.

### 7.5. Dimensionality reduction

You have a data set with 3-dimensional feature vectors and calculate the correlation matrix to be

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \tag{1}$$

Based on this information, explain what we can tell about the relationship between the features. Illustrate how this knowledge can be used.

### 7.6. Eigendecomposition of the covariance matrix

a) Find the eigenvalues and the eigenvectors to the covariance matrix $\mathbf{C} = \begin{bmatrix} 4 & \sqrt{3} \\ \sqrt{3} & 2 \end{bmatrix}$.

b) How are the eigenvectors related?

c) How are the eigenvalues interpreted?

### 7.7. Principal Component Analysis derivation

a) Derive the eigenvalue problem of PCA on the basis of maximizing the variance of the projection $\hat{\mathbf{w}}^T\mathbf{x}$. $\hat{\mathbf{w}}$ is the direction the data $\mathbf{x}$ should be projected upon.

b) Show that maximizing the variance of the projection is equal to minimizing the mean square error $|\mathbf{x} - (\hat{\mathbf{w}}^T\mathbf{x})\hat{\mathbf{w}}|^2$.

### 7.8. Principal direction

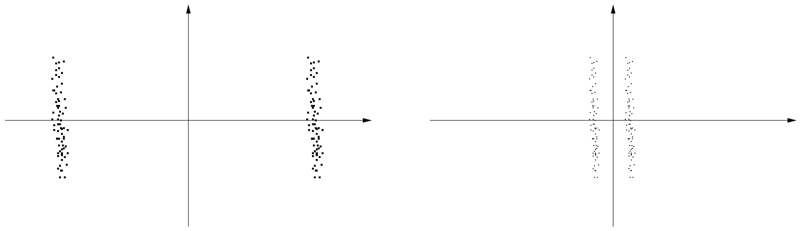
Assume data of three dimensions $\mathbf{z} = (z_1, z_2, z_3)^T$. We calculate the first principal direction (largest eigenvector of the covariance matrix) as $\mathbf{w}_1 = \frac{1}{3}(2, 1, 2)^T$. What does the result mean?

### 7.9. Shape of a Gaussian distribution

A Gaussian distribution of two dimensions has the mean $\mathbf{0}$ and the covariance $\begin{pmatrix} 3 & -1 \\ -1 & 3 \end{pmatrix}$.
Calculate the principal directions and draw, based on these directions, a sketch of the distribution with the help of contour lines. Include the principal directions in the sketch.

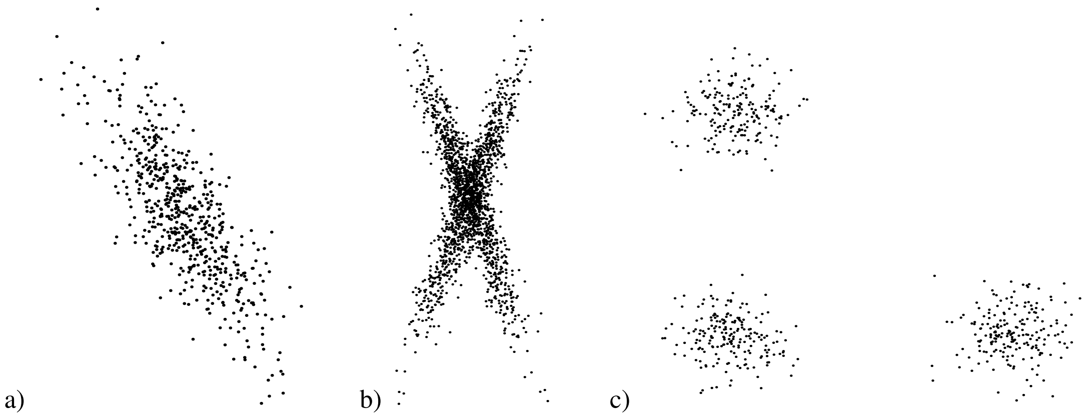
### 7.10. More principal directions

Draw the first principal direction in the two data sets below. What conclusions can you make about PCA as a preprocessing step before classification? The two data sets consist of samples from two classes, separated by the y-axis.



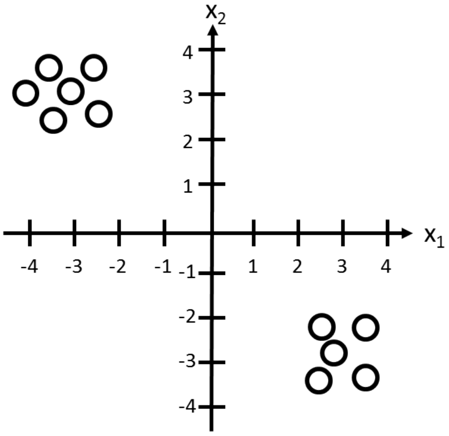
### 7.11. Even more principal directions

Assume three sources producing two dimensional data according to the figures below. Draw the principal directions.



### 7.12. PCA + k-Means

The figure below shows 2-dimensional training examples. Sketch and draw what happens if you first apply PCA to reduce the dimension of the training data to 1, and then apply k-Means clustering to the result. Also give approximate numbers of what would be the result of the k-Means algorithm.



## 7.13. Data compression

Assume we receive packets of 3D-data from a source $\mathbf{x} = (x_1,\, x_2,\, x_3)^T$. We should transmit these data on a channel only capable of transmitting packets of 1D-data. PCA has been used to find the first principal direction $\mathbf{e}_1 = (-1, 1, 1)^T$ also known on the receiver side.
Assume that we receive a packet with the following data

$$\mathbf{x} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}$$

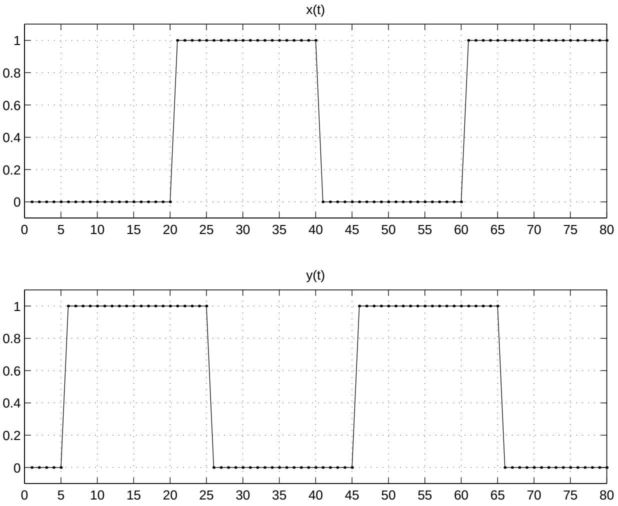
With the help of the first principal direction we can do a dimension reduction so that we can transmit this package.
Show how to do this!
In what sense is this an optimal method to reduce the dimensionality of the data?

## 7.14. PCA of signals

A source gives a two dimensional signal

$$\mathbf{s}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$$

$x(t)$ and $y(t)$ are shown in the figures below:



Find two normalized vectors $\hat{\mathbf{w}}_1$ and $\hat{\mathbf{w}}_2$ so that

$$\hat{s}_1(t) = \hat{\mathbf{w}}_1^T \mathbf{s}(t)$$

and

$$\hat{s}_2(t) = \hat{\mathbf{w}}_2^T \mathbf{s}(t)$$

get maximum and minimal variance respectively. Sketch the new signals.

### 7.15. More feature dimensions than training samples

Assume that we have $2000$ samples of data of $50$ dimensions. We represent the data with a matrix $\mathbf{X}$ of the size $50 \times 2000$ (one training sample per column). The covariance matrix $\mathbf{C}_{xx} = \mathbf{X}\mathbf{X}^T$ (when X is mean-centered) then has the form $50 \times 50$, which makes the eigenvalue problem in PCA easy to solve. If we instead have $50$ training samples of data of $2000$ dimensions, the size of the covariance matrix is $2000 \times 2000$, which isn't that fun to calculate. Show that it is ok to instead use the matrix of scalar products between the samples ($\mathbf{X}^T\mathbf{X}$) in order to find the principal directions.

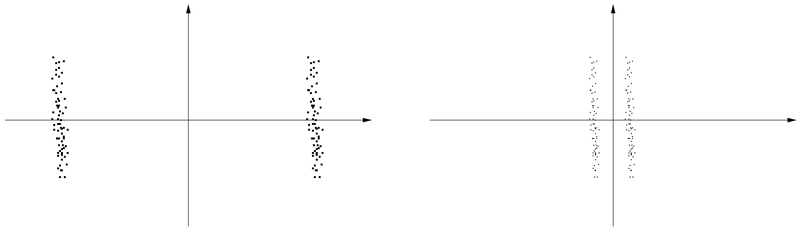
### 7.16. Fisher Linear Discriminant

a) Write the cost function being optimized with Fisher Linear Discriminant/Linear Discriminant Analysis. Explain the notations you use!

b) What assumption is made about the distributions of the two classes in linear discriminant analysis?

### 7.17. FLD vs. PCA

Fisher Linear Discriminant analysis results in a vector that can be used for classification or dimension reduction. Draw the resulting vectors in the two datasets below. Explain the principal differences between LDA and PCA, if any. The data represent two classes separated by the y-axis.



# 8. Genetic Algorithms

## 8.1. Genetic algorithm optimization

Describe the kind of optimization problems for which genetic algorithms may be a suitable method.

## 8.2. Cross-over

Show how a cross-over between the string representations 'ABCBC' and 'BBCAC' might look like in the context of genetic Algorithms

## 8.3. Mutation

Why is the mutation operation important in genetic algorithms?

## 8.4. Applying a genetic algorithm

The Sneaky Brothers have broken into the mansion of one of their wealthy neighbors. They intended to steal seven precious items, but the things are heavier than they thought and they must leave some stuff behind. The brothers want to maximize the value of the things they take with them. They decide to use a genetic algorithm in order to choose which of the items to steal.
The weight and value of the items are presented in the table below.

| Item | Weight (kg) | Value ($) |
|---|---|---|
| 1 | 5 | 200 |
| 2 | 10 | 1000 |
| 3 | 5 | 300 |
| 4 | 3 | 400 |
| 5 | 1 | 200 |
| 6 | 6 | 100 |
| 7 | 3 | 300 |

The brothers can only carry a maximum of 20 kg together without raising the suspicion of the neighborhood. Your task is to help the brothers design and use the genetic algorithm.

a) Describe a suitable solution representation, a 'fitness"-function and how the crossover and mutation operations can be performed.

b) Start with a population of three individuals and run the algorithm one iteration so the next generation is created. Assume a reproduction step where the two best fitted individuals are copied unaltered and they together create a new individual through the crossover operation you suggested in the first part of the question.

## 8.5. Crossover and mutation

We start of with a number of definitions in order to make further calculations easier. Define the order of a schema $S$, as the number of fixed positions, $o(S)$. The distance between the first and the last fixed position in the schema is denoted as $\delta(S)$.

a) Assume a crossover between two strings of the length $l$ is taking place be means of a two step process. First a random position $k$ is drawn from a rectangular distribution on the interval $\{1, l-1\}$. Then the strings swap the parts between and including position $k+1$ and $l$ with each other. Derive a lower bound for the probability, $p_s$, that a schema of the length $l$ survives a crossover given the probability of the crossover itself, $p_k$.

b) Let us also consider the possibility of mutation. The probability that a given position should be affected is assumed to be $p_m$. What is the lower bound for survivability of a schema now?

## 8.6. The schema theorem

The Schema Theorem can be expressed as:

$$m(S, t+1) \geq \frac{f(S)}{\langle f \rangle} m(S, t) \left(1 - p_k \frac{\delta(s)}{l-1} - O(S) p_m\right)$$

where $m(S,t)$ denotes the number of copies of the schema $S$ at the time $t$. $f(S), p_k$ and $p_m$ is the fitness of the schema and the probability of crossover and mutation respectively. Show "the Schema Theorem", i.e. the number of copies of a schema $S$ in a population will increase or decrease exponentially with respect to the relative fitness of the schema. Disregard crossover- and mutation effects, i.e. assume $p_k = p_m = 0$.

### 8.7. The survival of the fittest

A population contains strings with the following corresponding fitness:

| No. | String | Fitness |
|---|---|---|
| 1 | 10001 | 20 |
| 2 | 11100 | 10 |
| 3 | 00011 | 5 |
| 4 | 01110 | 15 |

The probability for mutation is $p_m = 0.01$ and the probability for crossover is $p_k = 1.0$. Calculate the expected number of schemata with the string $S_1 = 1****$ and $S_2 = 0**1*$ respectively in the next generation. Comments?

### 8.8. Live and let die

Let us in this exercise ignore the possibility of the destruction of a schemata due to crossovers and mutations.

a) A schema $S_1$ with one representative in the first generation has $25\%$ larger fitness value than the average in the population of 100 individuals. After how many generations will this schema appear in every individual in the population?

b) A schema $S_2$ appearing in 60 (first generation) of the 100 individuals has $10\%$ lower fitness value than the average. After how many generations will this schema be extinct?

### 8.9. 1D optimization

We want to use a simple genetic algorithm in order to maximize the function $f(x) = x^2$ on the interval $x \in [-15, 16]$ (-15 is represented by 00000 and 16 by 11111). We code the variable $x$ using a binary string with 5 bits where the leftmost bit is the most significant.
At the beginning, at the time $t = 0$, we have the following population:

| String nr. | Population at $t = 0$. |
|---|---|
| 1 | 01100 |
| 2 | 11000 |
| 3 | 01000 |
| 4 | 10011 |

a) We use the function we want to maximize as our fitness function. Then, we select individuals for reproduction with a probability proportional to their relative fitness. How many copies of the different individuals can we expect to be selected for reproduction? How good/fit is the initial population on average?

b) Assume that two copies of individual 2, one copy of individual 3 and one copy of individual 4 are selected for reproduction. At the reproduction, we randomly choose pairs for the crossover operation. The crossover is implemented by choosing a string position at random, and then let the individuals swap substrings after this position. Let us assume that a crossover operation is performed on a copy of individual 2 and a copy of individual 3 at string position 4. Another crossover is performed on a copy of individual 2 and a copy of individual 4 as string position 2. Describe the population after these operations. How good/fit is the new population on average?

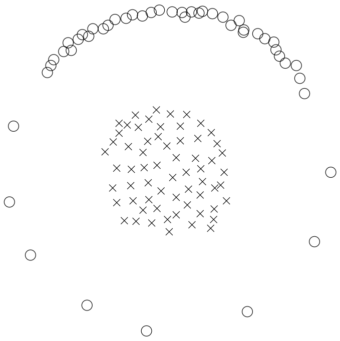
c) Calculate how many individuals we can expect to carry the schema $S = (1****)$ in the next generation, i.e. at the time $t = 1$.

NOTE: This is not the type of optimization problem that one typically would use a genetic algorithm for.

## Kernel Methods

### 8.10. Classification with a nonlinear mapping

The figure below shows the data for a classification problem in the xy-plane. Draw the optimal decision boundaries for a linear classifier that have access to $1, x_1, x_2$. Also draw the optimal boundaries for a classifier based on a linear combination of $1, x_1, x_2, x_1x_2, x_1^2, x_2^2$. The optimal solution is the one with least number of misclassifications and largest margins if possible.



### 8.11. Scalar product

a) What is the scalar product (a.k.a. as dot product or inner product) between two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$?

b) Show how the length of a vector $\mathbf{x}$ can be expressed in terms of the scalar product.

c) Show how the distance between two points $\mathbf{x}_1$ and $\mathbf{x}_2$ can be expressed in terms of the scalar product.

d) Show how the angle between $\mathbf{x}_1$ and $\mathbf{x}_2$ can be expressed in terms of the scalar product.

The conclusion of this exercise is that if we know the values of the scalar products between vectors, we also know how these vectors are geometrically positioned relative to each other.

### 8.12. Kernel definition

What is defined by a kernel function?

### 8.13. Using the kernel function

Consider a Gaussian kernel function $\kappa(\mathbf{x}_1, \mathbf{x}_2) = e^{-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{4}}$. What is the distance between two feature vectors

$$\mathbf{x}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \text{ and } \mathbf{x}_2 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

in the new feature space defined by this kernel function?

### 8.14. Explicit mapping vs. implicit mapping with kernel

Consider the following non-linear mapping of the input data $\mathbf{x}$:

$$\begin{aligned} \varphi_1(\mathbf{x}) &= x_1^2 \\ \varphi_2(\mathbf{x}) &= x_2^2 \\ \varphi_3(\mathbf{x}) &= \sqrt{2} \cdot x_1 x_2 \end{aligned}$$

You want to analyse this data with a kernel method. How is the scalar product $\varphi(\mathbf{x}_1)^T \varphi(\mathbf{x}_2)$ expressed in the input data space?

### 8.15. Kernel matrix

a) What is the kernel matrix $\mathbf{K}$ (a.k.a. Gram matrix or similarity matrix)?

b) We have 10 training samples in a 20-dimensional space that we want to analyse with a kernel method. How large is the kernel matrix?

### 8.16. Interpreting the kernel matrix geometrically

You get the following kernel matrix

$$\mathbf{K} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 2 & 2 \\ 0 & 2 & 4 \end{pmatrix}.$$

Plot the training data vectors that could have generated this kernel matrix.

### 8.17. Removing the mean

Suppose we have a quadratic mapping of the input signal $\mathbf{x}$ to a high-dimensional feature space $\mathbf{x} \rightarrow \varphi(\mathbf{x})$ according to $\varphi(\mathbf{x}) = \mathbf{x} \times \mathbf{x}$ where "$\times$" means that you take the outer product and then make a vector of the resulting matrix, which will contain all products between the components of the input vectors.

a) What will the kernel matrix look like if we have the following three data vectors?

$$\begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

b) In some algorithms, such as PCA, it is required to remove the mean from the data. However, the kernel matrix above corresponds to the original and non-centered samples in the feature space. Show that you get a kernel matrix that corresponds to the samples being centered in the feature space by, from the non-centered kernel matrix subtracting the column mean from each column, then subtract the row mean from each row, and finally add the total mean value of the whole matrix, i.e.

$$k'_{ij} = k_{ij} - \frac{1}{n}\sum_i k_{ij} - \frac{1}{n}\sum_j k_{ij} + \frac{1}{n^2}\sum_{ij} k_{ij}$$

where $k'$ are the components in the centered kernel matrix.

### 8.18. Kernel trick

What is required of an optimization problem in order to be able to solve it with kernel methods?

### 8.19. Kernel trick applied to SVM

The optimal plane separating two linearly separable classes is in a Support Vector Machine found by optimizing the cost function

$$\min \|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w}$$

$$\text{subject to the constraint } y_i\left(\mathbf{w}^T\mathbf{x}_i + w_0\right) \geq 1 \text{ for all } i,$$

where $\{\mathbf{x}_i, y_i\}$ are the training examples and $y_i = \pm 1$. Show how the kernel trick is applied to optimize a nonlinear SVM classifier.

### 8.20. Support vectors

After training an SVM, the resulting discriminant function $f(\varphi)$ is given by:

$$f(\varphi) = 4\varphi_1 + 9\varphi_2 + 4\varphi_3 - 16\varphi_4$$

The training samples $\mathbf{x}$ have been mapped according to: $\varphi_1(\mathbf{x}) = x_1^2$, $\varphi_2(\mathbf{x}) = x_2^2$, $\varphi_3(\mathbf{x}) = x_1x_2$, and $\varphi_4(\mathbf{x}) = 1$.
Is any of the two samples $\mathbf{x} = (1 \ \ 1)^T$ and $\mathbf{y} = (1 \ \ -1)^T$ a support vector? Why/why not?

# 9. Reinforcement learning

### 9.1. Feedback

Describe the difference in feedback between supervised and reinforcement learning.

### 9.2. Delayed feedback

Describe the “temporal credit assignment problem”?

### 9.3. V vs. Q

What is the difference between the value function V and the Q-function in Reinforcement learning?

### 9.4. Discount factor

What does the *discount factor* control in Reinforcement Learning?

### 9.5. Exploration vs. exploitation

Explain the principle of *$\epsilon$-greedy exploration*.

### 9.6. Get the V- and Q-functions 1

The figure shows three different deterministic state models and their corresponding reward functions. The states are numbered from 1 to 5 and arrows represent actions denoted as ”up”, ”same” and ”right”. The numbers close to the arrows show the reward. If the system reaches a state called ”End” no more rewards are given, i.e. the V-function is defined as 0 in these states. With optimal means the policy that maximizes the reward.

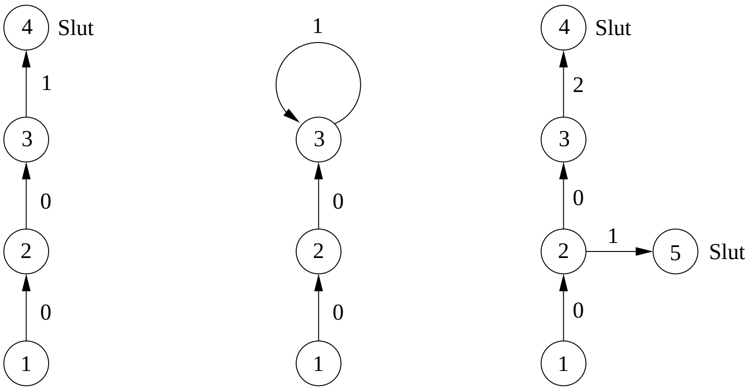


Figure 1: The state models A, B and C.

a) Calculate the optimal Q- and V-function for system A as a function of $\gamma$. (1p)

b) Calculate the optimal Q- and V-function for system B as a function of $\gamma$. Interpret the result in words. (2p)

c) Calculate the optimal Q- and V-function for system C as a function of $\gamma$. Interpret the result in words. (2p)

### 9.7. Get the V- and Q-functions 2

The figure shows two different deterministic state models and their corresponding reward functions. The states are numbered from 1 to 6 and arrows represent actions denoted as ”forward” and ”shortcut”. The numbers close to the arrows show the reward.

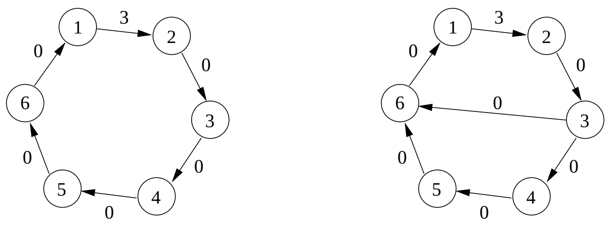


Figure 2: The state models A and B.

a) Describe the optimal policy for system A and B respectively. (1p)
b) Calculate the optimal Q- and V-functions for system A as a function of $\gamma$ $(0 < \gamma < 1)$. (2p)
c) Calculate the optimal Q- and V-functions for system B as a function of $\gamma$ $(0 < \gamma < 1)$. (2p)

### 9.8. Get the V- and Q-functions 3

The figure shows two different deterministic state models and their corresponding stochastic reward functions. The states are numbered from 1 to 6 and arrows represent actions denoted as "up" and "left". The numbers close to the arrows show the reward. In the cases where the reward is stochastic a percentage states the probability of a certain reward.

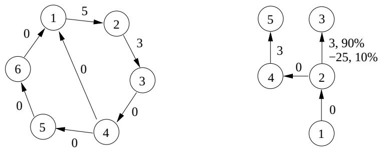


Figure 3: The state models A and B.

a) Calculate the optimal Q- and V-functions for system A as a function of $\gamma$ $(0 < \gamma < 1)$. (2p)
b) Calculate the optimal Q- and V-functions for system B as a function of $\gamma$ $(0 < \gamma < 1)$. (3p)

### 9.9. Get the V- and Q-functions 4

The figure below shows a state model of a system were the task is to get from state "1" to an end node with maximal reward over time. Find the optimal path using Q-learning with the allowed actions "up","down" and "right". When trying to moving right from state "1" there is a chance $p$ of ending up in state "3", and a chance $1 - p$ of ending up in state "4" $(0 \leq p \leq 1)$.

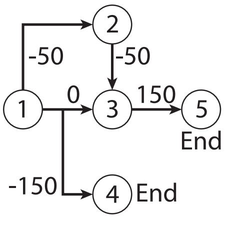


Figure 4: The state model. The numbers next to the arrows are the rewards for moving from one state to another.

a) Calculate the expected Q-function after a very large number of iterations with a learning rate $0 < \alpha < 1$ and discount factor $0 < \gamma \leq 1$.

b) Calculate the expected V function using $\gamma = 1$.

c) Why can we not use $\alpha = 1$ when we train this system?

# Solutions

# 1. Introduction and kNN

## 1.1. Answer

Supervised, Unsupervised and Reinforcement Learning

## 1.2. Answer

A feature describes a certain aspect of an object or phenomenon that we would like to classify. It is usually expressed as a numerical measurement.

## 1.3. Answer

a) $\mathbf{x}$ is a vector of input features, $w_1, w_2, \ldots, w_n$ are weights or parameters of the classifier that are adjusted in the training phase, and $\Omega$ is a set of discrete class labels.

b) In classification we want to learn to predict a discrete class label. In regression we want to learn to predict a continuous variable, for example a temperature, probability, stock price etc.

c) The parameters $w_1, w_2, \ldots, w_n$ are adjusted by an optimization procedure so that the system obtains the optimal skill to classify training data.

## 1.4. Answer

a) A learning method generalizes well if it performs well on previously unseen data, for example, that it can generalize what it has learned on training data to new data.

b) The generalization error can be estimated by testing the classifier on a test data set that was not used in the training.

c) To avoid over-fitting.

d) The algorithm has overtrained if it performs much better on the training data than on a test data set which it has not seen before.

## 1.5. Answer

The accuracy is the number of correctly classified examples divided with the total number of samples, i.e., (80+90)/(80+90+20+30) = 0.77.

## 1.6. Answer

Divide the data set into equal parts P1, P2 and P3 with 300 samples each. Then train and evaluate 3 times as follows:
Train using P1 and P2 and evaluate using P3.
Train using P1 and P3 and evaluate using P2.
Train using P2 and P3 and evaluate using P1.
The total performance is the average classification accuracy of the 3 runs.

## 1.7. Answer

Advantages are that k-nearest neighbor requires no training and it is easy to implement. Disadvantages is that one must store all training examples and it takes a long time to classify if the training data set is large, as the distance to all training examples must be calculated.

## 1.8. Answer

In the case of $k = 1$ X will belong to a black dot, since this is the closest sample. When $k = 3$ X will be classified as a square because two of the three closest samples are squares. In the case of $k = 2$ the votes is even between the two classes and some confusion occurs. One solution is to classify the X as a dot since this is the closest sample.

# 2. Linear Classifiers

## 2.1. Answer

A (hyper-)plane can be defined with the equation

$$\mathbf{w^T x} + b = 0, \tag{2}$$

where $\mathbf{w}$ is a normal vector to the plane (i.e. orthogonal to the plane). $\mathbf{x_p}$ is a point in the plane (any point will do) and $b$ is a scalar adjusting the offset of the plane from the origin. If $b = 0$, the plane goes through the origin.
How should we interpret this equation? Consider a point $\mathbf{x_p}$ lying somewhere in the plane. Also consider a vector $\mathbf{x}'$ from $\mathbf{x_p}$ to an arbitrary point $\mathbf{x}$. If $\mathbf{x}$ lies in the plane, $\mathbf{x}'$ should not have any projection on the normal of the plane (right?). This gives us

$$\mathbf{w^T}(\mathbf{x} - \mathbf{x_p}) = 0 \tag{3}$$

This is also an equation for the plane; We just need to know the normal vector and an arbitrary point in the plane for the plane to be well defined. We can however rewrite the equation as:

$$\mathbf{w^T x} - \mathbf{w^T x_p} = 0 \tag{4}$$

If we then name $-\mathbf{w^T x_p}$ as $b$, we have returned to the original equation. We can consequently calculate $b$ as:

$$b = -\mathbf{w^T x_p} \tag{5}$$

If we anew consider another arbitrary point $\mathbf{x}$ and would like to know the distance to the plane, we can proceed like this:
Let $\mathbf{x}'$ be a vector from $\mathbf{x_p}$ to $\mathbf{x}$, i.e. $\mathbf{x}' = \mathbf{x} - \mathbf{x_p}$. (In this context, a point is the same thing as a vector from the origin to the coordinate of the point). Also let $\mathbf{x_o}$, be a vector orthogonal to the plane, going from the plane to $\mathbf{x}$. It is the length of this vector we seek!. This vector is parallel to $\mathbf{w}$ and therefore projects its whole length on $\mathbf{w}$. As we can see in the picture, it is also obvious this projection on $\mathbf{w}$ has the same length as the projection of $\mathbf{x}'$ on $\mathbf{w}$. Consequently, we only need to calculate this projection:

$$\mathbf{x_o} = \frac{\mathbf{x'^T w}}{\mathbf{w^T w}}\mathbf{w} \tag{6}$$

where the denominator is for the normalization of $\mathbf{w}$.
The distance to the plane, i.e. the length of this vector is:

$$|\mathbf{x_o}| = \frac{\mathbf{x'^T w}}{|\mathbf{w}|^2}|\mathbf{w}| = \frac{\mathbf{x'^T w}}{|\mathbf{w}|} = \frac{(\mathbf{x} - \mathbf{x_p})^\mathbf{T}\mathbf{w}}{|\mathbf{w}|} \tag{7}$$

Note that the sign of the projection $\mathbf{x'^T w}$ tells us what side of the plane the point lies. If the projection is greater than zero we are on the side in the direction of the normal vector $\mathbf{w}$. If it is less than zero, we are on the other side. If the projection is zero, we are exactly in the plane.

## 2.2. Answer

a) See lecture notes!

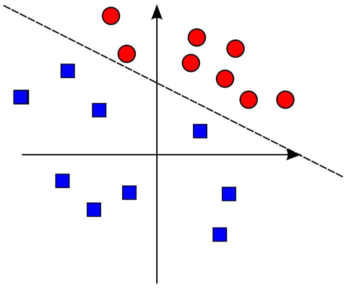
b) The "bias weight" makes it possible to move the discriminating hyperplane away from the origin.

c) The input value to the bias weight is typically 1, but can be any value except zero.

d) The bias weight offsets the decision boundary from the origin, so we need an example where the optimal boundary does not go through the origin, for example

## 2.3. Answer

In *batch learning*, all the training samples are used for updating the classifier. In *online learning*, the classifier is updated using one training sample at the time.



### 2.4. Answer

a) $\mathbf{w}^T = [w_1 w_2 \dots w_n]\,, \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

$$\mathbf{y} = \mathbf{w}^T\mathbf{x} = w_1x_1 + w_2x_2 + \ldots + w_nx_n = \sum w_ix_i$$

$$\frac{\partial y}{\partial w_i} = x_i\,, \qquad \frac{\partial y}{\partial \mathbf{w}} = \mathbf{x}$$

b) $\mathbf{y} = \begin{bmatrix} w_{11}\dots w_{1m} \\ \vdots \ddots \vdots \\ w_{11}\dots w_{1m} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} w_{11}x_1 + \ldots + w_{1m}x_m \\ \vdots \\ w_{n1}x_1 + \ldots + w_{nm}x_m \end{bmatrix}$

$$\frac{\partial y}{\partial \mathbf{W}} = \begin{bmatrix} \frac{\partial y}{\partial W_{11}} \cdots \frac{\partial y}{\partial W_{1m}} \\ \vdots \ddots \vdots \\ \frac{\partial y}{\partial W_{n1}} \cdots \frac{\partial y}{\partial W_{nm}} \end{bmatrix} = \begin{bmatrix} x_1 \dots x_m \\ \vdots \ddots \vdots \\ x_1 \dots x_m \end{bmatrix} = \begin{bmatrix} \mathbf{x}^T \\ \vdots \\ \mathbf{x}^T \end{bmatrix}$$

c) $y = \sum_i \sum_j x_iW_{ij}x_j\,, \qquad \frac{\partial y}{\partial x_k} = \sum_i x_iW_{ik} + \sum_j x_jW_{kj}$

$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \sum_i x_iW_{i1} + \sum_j x_jW_{1j} \\ \vdots \\ \sum_i x_iW_{in} + \sum_j x_jW_{nj} \end{bmatrix} = \mathbf{Wx} + \mathbf{W}^T\mathbf{x}$. Can be reduced to $2\mathbf{Wx}$ if $\mathbf{W}$ is symmetric.

d) $y = \left(\sqrt{w_1^2 + \ldots + w_n^2}\right)^4 = \left(w_1^2 + \ldots + w_n^2\right)^2$

$$\frac{\partial y}{\partial w_i} = 2(w_1^2 + \ldots + w_n^2) \cdot 2w_i = 4(\mathbf{w}^T\mathbf{w}) \cdot w_i$$

$$\frac{\partial y}{\partial \mathbf{w}} = 4(\mathbf{w}^T\mathbf{w})\mathbf{w}$$

e) $\frac{\partial \mathbf{y}}{\partial \mathbf{W}} = \mathbf{A}\frac{\partial \mathbf{g}}{\mathbf{s}}\frac{\partial \mathbf{s}}{\mathbf{W}} = \mathbf{A}\exp(\mathbf{Wx}) \begin{bmatrix} \mathbf{x}^T \\ \vdots \\ \mathbf{x}^T \end{bmatrix}$

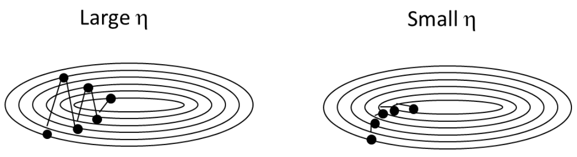
### 2.5. Answer

a) $\frac{\partial f}{\partial \mathbf{x}} = 6\mathbf{x} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ which gives $\mathbf{x} = \begin{bmatrix} -\frac{1}{6} \\ -\frac{1}{6} \end{bmatrix}$.

b) $\mathbf{x}_t = \mathbf{x}_{t-1} - \eta\frac{\partial f}{\partial \mathbf{x}}$ where $\frac{\partial f}{\partial \mathbf{x}} = 6\mathbf{x}_{t-1} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. We then get $\mathbf{x}_1 = \begin{bmatrix} 0.65 \\ 0.65 \end{bmatrix}$ and $\mathbf{x}_2 = \begin{bmatrix} 0.4050 \\ 0.4050 \end{bmatrix}$

### 2.6. Answer

The effect of a too long step length may be an oscillating behaviour that may have the effect that we never converge to a local minimum. The effect of a too small step length is that we move very slowly towards the local minimum, i.e., the training takes a long time.



### 2.7. Answer

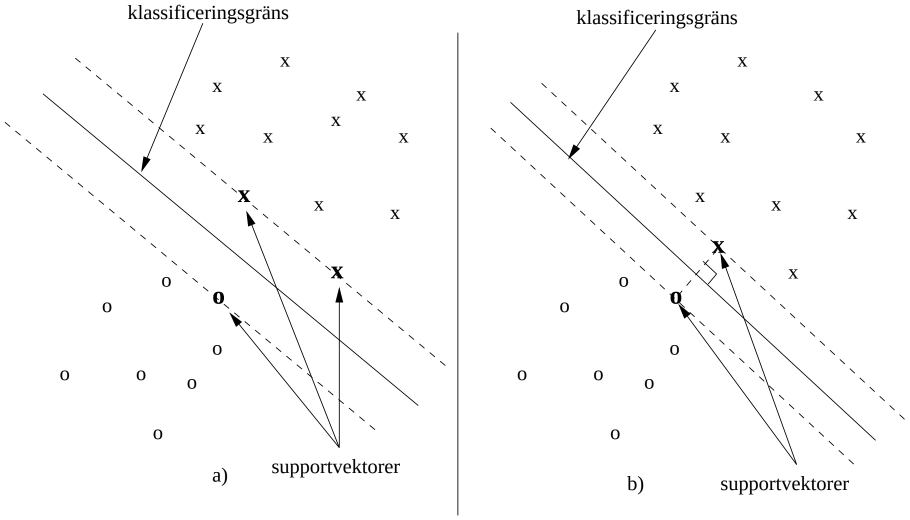
The cost function $\sum_{k=1}^{N} I(z_i \neq y_i)$ is piecewise flat and not differentiable at all points, so that we do not get any information from the gradient, which is required in gradient descent.

### 2.8. Answer

The maximum margin principle says that the decision boundary between two classes should be placed so that the distance between the boundary and the training data should be as large as possible, i.e., the error-margin should be maximized.

### 2.9. Answer

The optimal hyper plane is the line tat best separates the to classes, i.e. the line that is as far away from the closest points as possible. The support vectors are the samples that are closest to this plane.



### 2.10. Answer

$\|\mathbf{w}\|$ is minimized and the equation holds with equality for the *support vectors*.

### 2.11. Answer

SVM have good generalization properties and are usually better than e.g. back-propagation networks on small datasets. On large datasets, however, it gets computationally heavy since the kernel matrix grows quadratically with the number of samples.

## 3. Neural Networks and Non-Linear Classifiers

### 3.1. Answer

a) The “xor-problem” has one class defined as the points (-1,1) and (1,-1) and the other class at the points (1,1) and (-1,-1). These classes cannot be separated with a linear classifier.

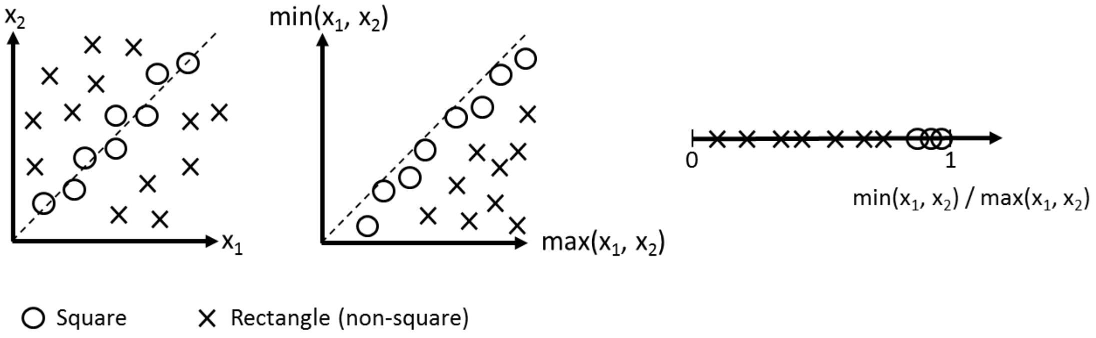
b) It is not linearly separable.

### 3.2. Answer

The probability that classes are linearly separable increases when the features are nonlinearly mapped to a higher dimensional feature space.

### 3.3. Answer

The figure shows three different ways this can be done.



- One can keep $x_1$ and $x_2$ as they are as features. The squares will the be located along the diagonal line because $x_1 \approx x_2$, while the non-square rectangles will be located away from the diagonal in the feature space. A non-linear classifier is required for this case, e.g., a neural network, SVM, ...
- One can create two new features $y_1 = \max(x_1, x_2)$ and $y_2 = \min(x_1, x_2)$. The feature space will then look like the middle plot, and a linear classifier will be enough.
- Another option is to calculate a new 1D-feature as the ratio $y = \frac{\min(x_1,x_2)}{\max(x_1,x_2)}$, shown in the right figure. Again, a simple linear classifier will do (i.e., a single threshold in this case).

### 3.4. Answer

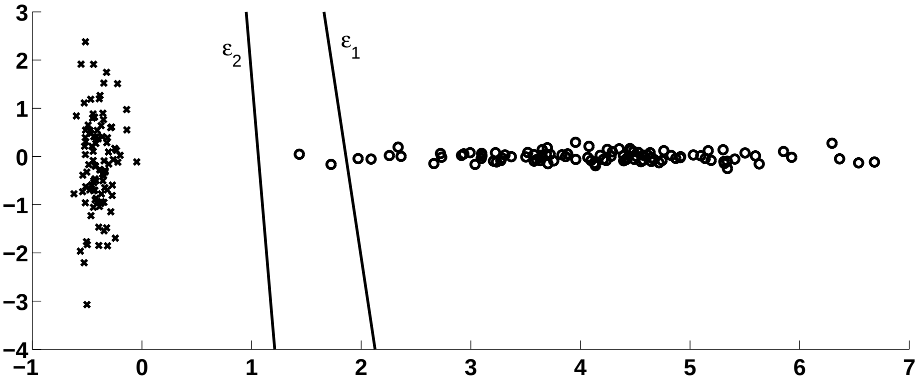
For example, one may used a neural network with 5 input nodes, one for the temperature of each of the 5 past days. Given historical data over 6-day periods, one can use the first 5 days as training data and the 6th day as the correct answer. The neural network can in this way be trained to predict tomorrows temperature based on the past 5 days.

### 3.5. Answer

a) Not really, tanh(x) never becomes $\pm 1$, just very close. If you try to use $+1$ (or $-1$) you may force the weights to very large values and risk the robustness of the network. Try using a slightly smaller value. (In practice $\pm 1$ often work though...)

b) The activation function in the output layer can not provide more advanced class boundaries or such. However it can bound the output values which is a good protection against outliers; e.g. very deviating training samples will not affect the solution as much with some kind of sigmoid activation function.

c) The problem here is that we have two linearly separable classes whom can be classified using a simple threshold on the horizontal axis. They do however have different distributions along this axis, a factor that can confuse our training if we are not careful. The first error function $\epsilon_1$ is sensitive to the distance of the samples. The effect is that the right most cluster will have a higher impact in the total error, and the boundary will shift closer to the right class. This is somewhat fixed with $\epsilon_2$ were $\tanh$ quickly saturates to about $\pm 1$, thus limiting the effect of the different distributions. The figure below shows the result after training a single neuron using the two functions. Here we can see that the theory holds true.



### 3.6. Answer

a)

$$\Delta w(t) = \alpha \Delta w(t-1) - \eta \frac{\partial \epsilon(t)}{\partial w}$$

b) You have to answer this during the back-prop assignment.

### 3.7. Answer

Since both the Euclidian and RBF metrics given above are radially symmetric around $\mu$ they have problems with the radial asymmetric form of the data. By including the covariance matrix the contributions of differences in x or y can be tuned in such a way that the metric better fits the data.

### 3.8. Answer

a) We want to find the gradient direction in which the error decreases the most. This direction tells us how much we should change each of the individual parameters (weights) $V_{jk}$ and $W_{ij}$. The notation is based on the lecture. There is also an index $\mu$ in superscript , e.g. $\mathbf{x}^\mu$ denotes the $\mu$:th training sample (a column vector). We assume $N$ training samples with the dimension $D$, $M$ neurons in the hidden layer and $C$ neurons($= C$ classes) in the output layer. In order to train the network we have to run all training samples (in the off-line/batch version) through it and for each sample calculate every $z_i, h_j$ and so on. When we have done that, we will calculate the error gradient and update our weights $W$ and $V$ by taking a small step in the negative gradient direction, just as in the one layer case. (We negate the error gradient because we want to decrease the error, not increase it!) *NB! Please note the indexing of the weights! All input weights connecting to a certain neuron lies along a row in the weight matrix. When we discussed the simple perceptron, the weights were stored in a column vector...so we use the transposed version here!*

In order to calculate the error gradient, let us first line up the whole chain of functions involved:

$$\varepsilon = \frac{1}{N}\sum_{i=1}^{C}\sum_{\mu=1}^{N}\varepsilon_i^\mu \tag{8}$$

$$= \frac{1}{N}\sum_{i=1}^{C}\sum_{\mu=1}^{N}(y_i^\mu - z_i^\mu)^2 \tag{9}$$

$$z_i^\mu = \sum_{j=1}^{M} W_{ij}h_j^\mu \tag{10}$$

$$h_j^\mu = \sigma(s_j^\mu) \tag{11}$$

$$s_j^\mu = \sum_{k=1}^{D} V_{jk}x_k^\mu \tag{12}$$

Now, we want to calculate each of the partial derivatives $\frac{\partial\varepsilon}{\partial W_{ij}}$ and $\frac{\partial\varepsilon}{\partial V_{jk}}$. If we stack the components into a vector, we have found our error gradient.

*Don't let all the indices scare you!* Look at the graph and concentrate on how the weight connects to other nodes. Also observe how the output error propagates through the network from the output to the inputs.

The chain rule gives us for $W$ (the weights in the output layer):

$$\frac{\partial\varepsilon}{\partial W_{ij}} = \frac{1}{N}\sum_{\mu=1}^{N}\sum_{n=1}^{C}\frac{\partial\varepsilon_n^\mu}{\partial z_n^\mu}\frac{\partial z_n^\mu}{\partial W_{nj}} = \frac{2}{N}\sum_{\mu=1}^{N}(y_i^\mu - z_i^\mu)(-1)h_j^\mu = \tag{13}$$

$$= \frac{2}{N}\sum_{\mu=1}^{N}(z_i^\mu - y_i^\mu)\,h_j^\mu = \frac{2}{N}(\mathbf{z_i} - \mathbf{y_i})\mathbf{h_j}^T \tag{14}$$

Note that the sum over the number of output neurons has changed index name from $i$ to $n$, so that we don't confuse it with the index in the differentiation variable $W_{ij}$. In addition, at the second equal sign this sum vanishes because the only term contributing is when $n = i$. If you look at the figure you can see this as it is clear that only the error at the output node $n = i$ is dependent on the weight $W_{ij}$. The last step is a conversion to vector form. Note that $\mathbf{z_i}$ and $\mathbf{y_i}$ here is *row*-vectors(horizontal) with $N$ components. The output of the expression is therefore simply a scalar. In order to make this efficient, we would like to calculate the update of the whole $W$ at the same time. We can rewrite the expression above in that way, even if it can be a bit cumbersome. Let us rewrite the expression in two steps. First we eliminate the $j$-index ($j$ denotes a certain neuron in the hidden layer, remember?):

$$\frac{\partial\varepsilon}{\partial\mathbf{w_i}} = \frac{2}{N}(\mathbf{z_i} - \mathbf{y_i})H^T \tag{15}$$

$H$ is a matrix with output signals from each of the neurons in the hidden layer. It has the dimension of $MxN$, that is $M$ rows and $N$ columns. What remains is to eliminate the $i$-index ($i$ denotes a certain neuron in the output layer):

$$\nabla_W = \frac{\partial \varepsilon}{\partial \mathbf{W}} = \frac{2}{N}(Z - Y)H^T \tag{16}$$

$H$ and $Y$ is matrices of the dimension $CxN$ and hold the output value and the desired (training output) value respectively for all samples and all output neurons (classes). The dimension of V becomes $CxM$.

For $V$ we proceed in the same way. Look at the figure and follow the errors back through the net to the weight $V_{jk}$.

$$\frac{\partial \varepsilon}{\partial V_{jk}} = \frac{1}{N} \sum_{\mu=1}^{N} \sum_{i=1}^{C} \frac{\partial \varepsilon_i^\mu}{\partial z_i^\mu} \frac{\partial z_i^\mu}{\partial h_j^\mu} \frac{\partial h_j^\mu}{\partial s_j^\mu} \frac{\partial s_j^\mu}{\partial V_{jk}^\mu} = \tag{17}$$

$$= \frac{1}{N} \sum_{\mu=1}^{N} \sum_{i=1}^{C} 2(y_i^\mu - z_i^\mu)(-1) \left(W_{ij}\sigma'(s_j^\mu)x_k^\mu\right) = \tag{18}$$

$$= \frac{2}{N} \sum_{\mu=1}^{N} x_k^\mu \sum_{i=1}^{C} (z_i^\mu - y_i^\mu) W_{ij} \sigma'(s_j^\mu) \tag{19}$$

What remains is just(?) to rewrite this into matrix form. A somewhat cumbersome process leads to

$$\nabla_V = \frac{2}{N}((W^T(Z - Y)). * \sigma'(S))X^T \tag{20}$$

where $S$ has the same form as $H$ ($MxN$) and $V$ has the form $MxD$. Finally, we have to deal with the bias weights....In order not to force a number of planes to involuntary go through the origin, we have to add a bias column to $V$ and $W$. In addition we have to add ones to $X$ and $H$ (where?). This part you have to find out on your own when you do the lab.

b) The difference between online and offline mode is that in offline mode we have access to all samples before we start to train. In online mode we get access to one sample at the time. This means that we get a training sample, run it through the network, calculate the error gradient and update the weights after each arriving sample.

The error measure is the same as before but without the sum over the training samples. (Or you can use the same measure as before and set N=1):

$$\varepsilon = \sum_{i=1}^{C} (y_i - z_i)^2 \tag{21}$$

Then you just have to differentiate as usual. Note that when we train online it is a good idea to let the learning rate decrease with the number of iterations(epochs). Otherwise the last samples will have unproportional influence over the properties of the planes. You also have to make sure that training samples from different classes arrive in a balanced fashion. If all samples from one class arrive first and then all from another, we might get very bad solutions.

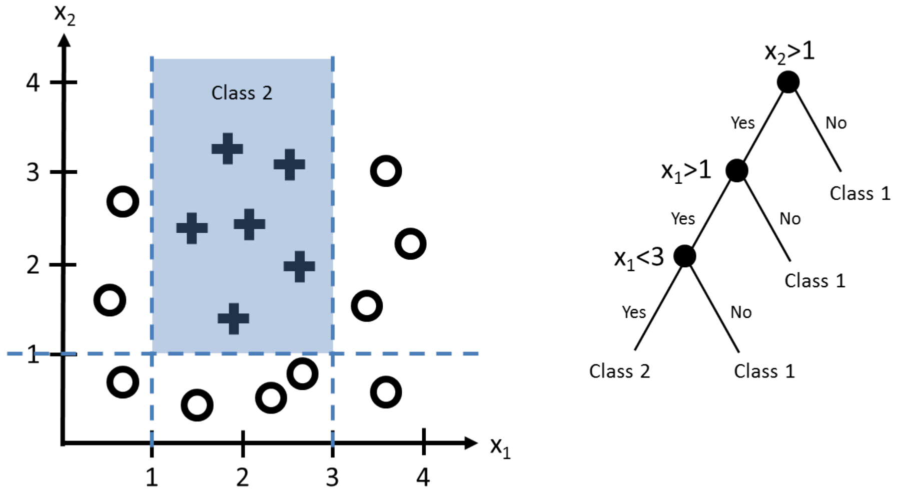
There is no point in iterating an online solution. When all samples have arrived we might just as well use the offline method and avoid all pitfalls.

Conclusion: Avoid online mode if you can. Even if you have to use online mode then you might consider alternatives; one example is to use offline mode training on all the samples that have arrived so far.

## 4. Ensemble Learning & Boosting

### Answer 4.1

- One of the many models are selected, and the choice of model is a function of the input variables. In the *decision tree*, a sequence of binary decisions (ordered as a binary tree) yields the chosen model. The models are in the ”leafs” of the tree and are often very simple, e.g. ”+1”.
- The leafs of the regression tree hold real values, not class labels.
- The figure below shows one possible solution. Note that there are other similar solutions.



### Answer 4.2

- A decision stump is a decision tree with a single node. It partitions the input space into two regions. The linear decision surface is perpendicular to one of the feature axes.
- The polarity and the threshold.
- $\min_{\tau,p} \epsilon(\tau, p) = \sum_{i=1}^{M} d_i I(y_i \neq h(x_i; \tau, p))$
- If the performance $\epsilon$ of the classifier is not better than 0.5, we can always change the polarity (flip the signs) so that we get $(1 - \epsilon)$ in performance.
- We only need to test thresholds in relation to the training data: $M$ thresholds

### Answer 4.3

Earlier in the course we have tried to make *one* classifier/regressor to perform as good as possible. Ensemble learning and Boosting, in contrast, combine *multiple* models to achieve this goal.

### Answer 4.4

a) *Bagging*, short for Bootstrap Aggregation, is a committee method where the individual models are trained on separate *bootstrap* datasets that are created from the original dataset using random sampling with replacement.

b) *Boosting* is also a committee method but the base models are trained in sequence, and each base model is trained using a weighted form of the data set in which the weighting coefficient associated with each data sample depends on the performance of the previous classifiers.

**Answer 4.5**

- A weak classifier is a simple classifier that may perform only slightly better than a random classifier. That is, for two classes, a weak classifier may give a classification accuracy of only slightly above 50%. The opposite of a weak classifier is a strong classifier, that aims to have a very high classification accuracy.
- Small decision trees or even decision stumps are typical weak classifiers.
- The XOR classification problem, see figure 5.
- See figure 5.

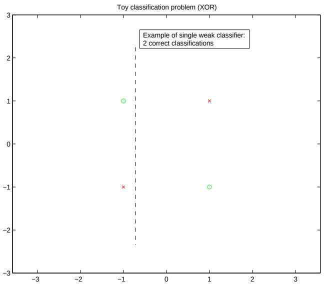


Figure 5: Toy example

**Answer 4.6**

See the lecture.

```
a) Set weights $d_1 := 1/N; i := 1;$
b) Train weak classifier $h_i(x)$ using weights $d_i$
c) Increase and decrease weight for wrongly and correctly classified training
   examples respectively -> $d_{i+1}$
d) $i := i + 1;$
e) Repeat from b) until $i > M$
```

**Answer 4.7**

a) In figure 6 the classification problem is sketched along with the initial weights. All vertical/horizontal decision stumps within the square of the data samples give the error $\epsilon = \frac{1}{4}$. In figure 6 we have chosen one of these lines as an solution example. We get $\alpha_1 = \frac{1}{2}\ln\frac{1-\epsilon_1}{\epsilon_1} = \frac{1}{2}\ln\frac{3/4}{1/4} = \frac{\ln 3}{2}$. Now we update (decreasing) the weights of the correctly classified samples with $e^{-\alpha_1} = \frac{1}{\sqrt{3}}$. The weight of the erroneous classified sample (upper right in the solution example) is instead increased with $e^{\alpha_1} = \sqrt{3}$. The sum of the weights are now $3\frac{1}{4\sqrt{3}} + \frac{\sqrt{3}}{4} = \frac{\sqrt{3}}{2}$. After normalizing the weights (dividing with the sum) we get the resulting weights as shown in the right part of figure 6. As we can see, AdaBoost works well here.

b) In figure 7 the classification problem is sketched along with the initial weights. All vertical/horizontal decision stumps within the square of the data samples give the error $\epsilon = \frac{1}{2}$. In figure 7 we have chosen one of these lines as an solution example. We get $\alpha_1 = \frac{1}{2}\ln\frac{1-\epsilon_1}{\epsilon_1} = \frac{1}{2}\ln\frac{1/2}{1/2} = \frac{\ln 1}{2} = 0$. If we try to update the weights, we get $e^{\pm\alpha_1} = 1$, i.e. the weights don't change. This means that the weak classifier in iteration two will face the same problem as in iteration 1. As we can see, AdaBoost does not work in this setting. One solution is to allow other types of weak classifiers.

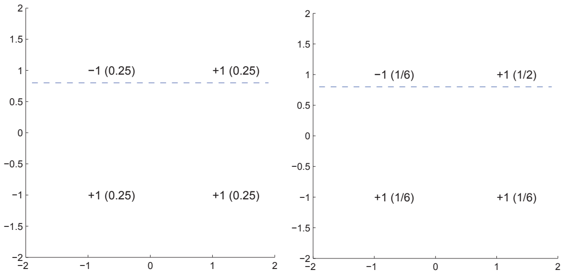


Figure 6:

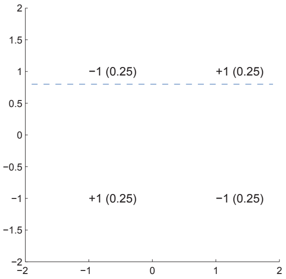


Figure 7:

c) Outliers gain weight exponentially which will eventually result in bad weak classifiers. This may ruin the composite strong classifier. Solutions to this problem include various forms of weight trimming and alternative weight update approaches.

## 5. Deep Learning

Material will be available at the class.

## 6. Unsupervised Learning - Clustering

### Answer 6.1

Hard clustering means that each training sample is assigned to only one cluster, whereas soft/fuzzy clustering means that a training sample can belong to several clusters to certain degrees.

### Answer 6.2

a) In $k$-means clustering, only the centers of the clusters are modelled, whereas in Mixture of Gaussian clustering, also the cluster shape is modelled using the covariance matrix, which can model circular and elliptical cluster shapes.

b) Expectation Maximization, commonly known as the EM-algorithm.

### Answer 6.3

k-NN: $k$ is the number of the stored data vectors that vote for the classification decision.
k-Means: $k$ is the number of clusters.

### Answer 6.4

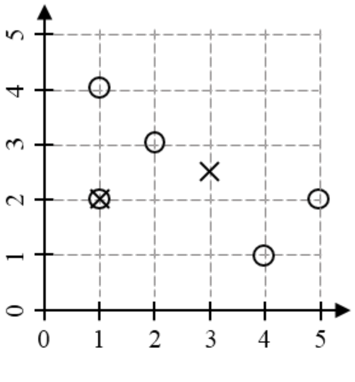
The prototype vectors are updated to the means of the closest data points, i.e.,

$$\mathbf{p}_1 = \frac{1}{4}\left[\begin{pmatrix}1\\4\end{pmatrix} + \begin{pmatrix}2\\3\end{pmatrix} + \begin{pmatrix}4\\1\end{pmatrix} + \begin{pmatrix}5\\2\end{pmatrix}\right] = \begin{pmatrix}3\\2.5\end{pmatrix}$$

and

$$\mathbf{p}_2 = \begin{pmatrix}1\\2\end{pmatrix}$$

as shown in the figure.



The algorithm must perform 2 more iteration before no more points change clusters.

### Answer 6.5

a) $N$-dim. normal (Gaussian) distribution:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{N/2}\sqrt{\det \mathbf{C}}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{m})\right]$$

b) Remember that $\mathbf{A^{-1}} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{\det(\mathbf{A})}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$.
Inserting the values in a) we get $p(x_1) = \frac{1}{2\pi\sqrt{8}} \approx 0.06, p(x_2) \approx 0.04, p(x_3) \approx 0.03$ respectively. $x_2$ is in the direction of the largest variance, i.e. the Gaussian has a wider support in that direction. $x_3$ on the other hand is in the direction of the smallest variance, i.e. the probability density decays faster in this direction.

**Answer 6.6**

Please see the Matlab demo code published on the course web site.

## 7. Unsupervised Learning - Structure Discovery

### Answer 7.1

a) $m_X = E(X)$, $m_Y = E(Y)$ and Cov($x, y$)= $C(X,Y) = E((X - m_X)(Y - m_Y)) = E(XY) - E(X)E(Y)$

b) $m_x \approx \frac{1}{N}\sum_{i=1}^{N} x_i$ and $C(x,y) \approx \frac{1}{N}\sum_{i=1}^{N}(x_i - m_x)(y_i - m_y)$

c) $m_X = E(X)$, and Var($x$)= $\sigma^2 = E((X - m_X)^2) = E(X^2) - (E(X))^2$

d) Var($x$)=Cov($x, x$)

e) $m_x \approx \frac{1}{N}\sum_{i=1}^{N} x_i$ and Var($x$)=$C(x,x) \approx \frac{1}{N}\sum_{i=1}^{N}(x_i - m_x)^2$

f) Corr($x, y$)=$\rho(x,y) = \frac{C(x,y)}{\sqrt{\sigma_x^2\sigma_y^2}}$ where $-1 \leq \rho \leq 1$

### Answer 7.2

a) $\mathbf{m_X} = E(\mathbf{X})$, and Cov($\mathbf{x}$)= $\mathbf{C}(\mathbf{X}) = E((\mathbf{X} - \mathbf{m_X})(\mathbf{X} - \mathbf{m_X})^T)$

b) $\mathbf{m_x} \approx \frac{1}{N}\sum_{i=1}^{N} \mathbf{x_i}$ and $\mathbf{C}(\mathbf{x}) \approx \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x_i} - \mathbf{m_x})(\mathbf{x_i} - \mathbf{m_x})^T$

c) $\mathbf{m_X} = E(\mathbf{X})$, $\mathbf{m_Y} = E(\mathbf{Y})$, and Cov($\mathbf{x}, \mathbf{y}$)= $\mathbf{C}(\mathbf{X}, \mathbf{Y}) = E((\mathbf{X} - \mathbf{m_X})(\mathbf{Y} - \mathbf{m_Y})^T)$

### Answer 7.3

a) $\mathbf{C} = \begin{bmatrix} var(x_1) = cov(x_1, x_1) & cov(x_1, x_2) & cov(x_1, x_3) \\ cov(x_2, x_1) & var(x_2) = cov(x_2, x_2) & cov(x_2, x_3) \\ cov(x_3, x_1) & cov(x_3, x_2) & var(x_3) = cov(x_3, x_3) \end{bmatrix}$.
Since $cov(x_i, x_j) = cov(x_j, x_i)$, $\mathbf{C}$ is symmetric.

b) $\mathbf{C}_{\text{corr}} = \begin{bmatrix} 1 & corr(x_1, x_2) & corr(x_1, x_3) \\ corr(x_2, x_1) & 1 & corr(x_2, x_3) \\ corr(x_3, x_1) & corr(x_3, x_2) & 1 \end{bmatrix}$.
Since $corr(x_i, x_j) = corr(x_j, x_i)$, $\mathbf{C}_{\text{corr}}$ also symmetric.

c) The covariance matrix will be scaled with a factor $5^2 = 25$. The correlation matrix is invariant to signal scaling.

d) A diagonal covariance matrix imply uncorrelated components in $\mathbf{z}(t)$.

### Answer 7.4

a) $m_{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i$ and $C(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - m_{\mathbf{x}})(\mathbf{x}_i - m_{\mathbf{x}})^T$
According to the above, we then get $m_{\mathbf{x}} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $C(\mathbf{x}) = \begin{bmatrix} 2 & -1 \\ -1 & \frac{2}{3} \end{bmatrix}$. In practice you need a lot more than three samples in order to get a good estimate of the mean value and the covariance.

### Answer 7.5

Let the feature vector be defined by

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

The correlation matrix tells us that the correlation between $x_1$ and $x_3$ is -1, which means that there is an inverse linear relationship $x_1 = -k\, x_3 + m$ with some $k > 0$ and some arbitrary difference in mean value $m$. The correlation between $x_1$ and $x_2$ is 0, as is the correlation between $x_3$ and $x_2$, i.e., there is no linear relationship between $x_2$ and the other features. This information can be used for dimensionality reduction, for example to make a new reduced feature vector

$$\tilde{\mathbf{x}} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

because $x_1$ and $x_3$ carry the same information (up to a scaling and mean value, which usually is not interesting).

**Answer 7.6**

a) In order to find the eigenvectors $\mathbf{e}$ to the matrix $\mathbf{C}$ we have to solve $\mathbf{Ce} = \lambda\mathbf{e}$. First, find the eigenvalues using the characteristic (secular) equation:
det$(\mathbf{C} - \lambda\mathbf{I}) = 0$
We then get the eigenvalues $\lambda_1 = 5$, $\lambda_2 = 1$. Insert these values and solve for the corresponding eigenvectors. We get $\mathbf{e}_1 = \frac{1}{2}\begin{bmatrix}\sqrt{3}\\1\end{bmatrix}$, $\mathbf{e}_2 = \frac{1}{2}\begin{bmatrix}1\\-\sqrt{3}\end{bmatrix}$

b) $\mathbf{e}_1$ and $\mathbf{e}_2$ are orthogonal, i.e., $\mathbf{e}_1^T\mathbf{e}_2 = 0$. For symmetric matrices, like the covariance matrix, the eigenvectors are always orthogonal as proven by the so-called Spectral Theorem in linear algebra.

c) $\lambda_1$ the variance in of the projected data onto $\mathbf{e}_1$, i.e., $\lambda_1 = \text{Var}(\mathbf{x}^T\mathbf{e}_1)$, where $\mathbf{x}$ is the 2-dimensional data vectors whose covariance matrix is $\mathbf{C}$.

**Answer 7.7**

a) $V = Var[\hat{\mathbf{w}}^T\mathbf{x}] = E[(\hat{\mathbf{w}}^T\mathbf{x})^T(\hat{\mathbf{w}}^T\mathbf{x})] = E[\hat{\mathbf{w}}^T\mathbf{x}\mathbf{x}^T\hat{\mathbf{w}}] = \hat{\mathbf{w}}^T\mathbf{C}_{xx}\hat{\mathbf{w}} = \frac{\mathbf{w}^T\mathbf{C}_{xx}\mathbf{w}}{\mathbf{w}^T\mathbf{w}}$
$\frac{\partial V}{\partial \mathbf{w}} = \frac{2\mathbf{C}_{xx}\mathbf{w}\cdot\mathbf{w}^T\mathbf{w} - 2\mathbf{w}^T\mathbf{C}_{xx}\mathbf{w}\cdot\mathbf{w}}{(\mathbf{w}^T\mathbf{w})^2} = 0$
$\mathbf{C}_{xx}\mathbf{w}\cdot\mathbf{w}^T\mathbf{w} = \mathbf{w}^T\mathbf{C}_{xx}\mathbf{w}\cdot\mathbf{w}$
$\mathbf{C}_{xx}\mathbf{w} = \frac{\mathbf{w}^T\mathbf{C}_{xx}\mathbf{w}}{\mathbf{w}^T\mathbf{w}}\cdot\mathbf{w} = V\mathbf{w}$, which is our eigenvalue problem.

b) $E[|\mathbf{x} - (\mathbf{w}^T\mathbf{x})\mathbf{w}|^2] = E[(\mathbf{x}^T - \mathbf{w}^T(\mathbf{x}^T\mathbf{w}))(\mathbf{x} - (\mathbf{w}^T\mathbf{x})\mathbf{w})] = E[\mathbf{x}^T\mathbf{x}] - 2E[(\mathbf{w}^T\mathbf{x})(\mathbf{x}^T\mathbf{w})] + E[(\mathbf{x}^T\mathbf{w})(\mathbf{w}^T\mathbf{x})\mathbf{w}^T\mathbf{w}] = E[\mathbf{x}^T\mathbf{x}] - E[(\mathbf{w}^T\mathbf{x})^T(\mathbf{w}^T\mathbf{x})] = E[\mathbf{x}^T\mathbf{x}] - V$, having its minimum where V is having its maximum.

**Answer 7.8**

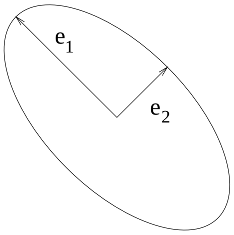
This means that $\frac{2}{3}z_1 + \frac{1}{3}z_2 + \frac{2}{3}z_3$ is the linear combination of the data components that has the highest variance among all possible projection directions.

**Answer 7.9**

The principal directions are the same as the eigenvectors of the covariance matrix. These are

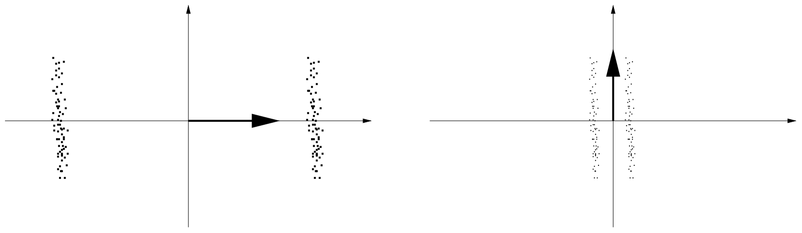
$$\hat{\mathbf{e}}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix}-1\\1\end{bmatrix} \text{ with the eigenvalue } \lambda_1 = 4$$
$$\hat{\mathbf{e}}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} \text{ with the eigenvalue } \lambda_2 = 2$$

and the eigenvalues specifies the variance in respective direction. A sketch of the distribution could therefore look something like this:



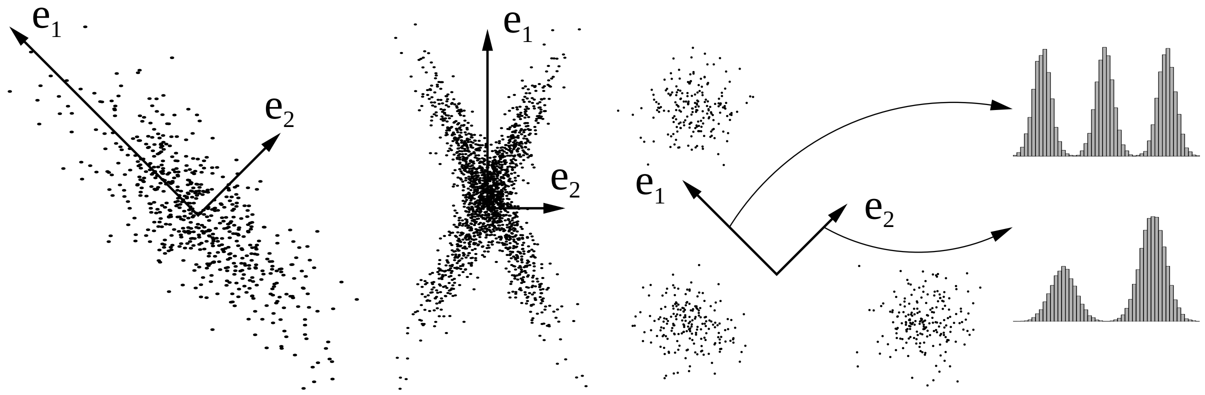
### Answer 7.10

The principal directions are shown in the figure. Since the variance of the projection is maximized, PCA is dependent on the scaling of the variables. This means the method is an appropriate preprocessing step in the first case but not in the second.



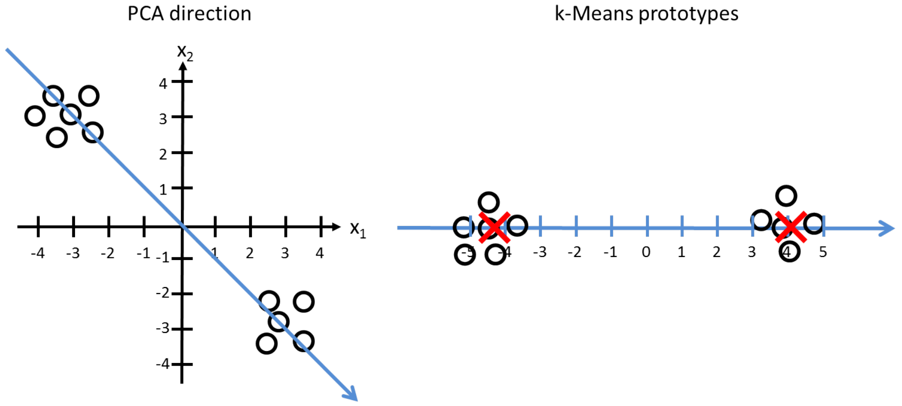
### Answer 7.11

The principal directions are those with maximal and minimal variance respectively (in the 2D-case).



### Answer 7.12

The left image below show the major PCA direction for dimensionality reduction. The right image shows the prototype vector positions at the center of the clusters (if we use k=2 clusters). The original clusters are centered around (-3,3) and (3,-3). After projection on the PCA direction, the clusters are centered around $\sqrt{3^2+3^2} \approx 4.2$ and -4.2, which should be the output from the k-Means algorithm


PCA direction

k-Means prototypes

**Answer 7.13**

a) The principal directions are orthogonal and if we normalize them they constitute an ON-base for the space. This means we can write $\mathbf{x}$ as a linear combination of the principal directions $\hat{\mathbf{e}}_1$, $\hat{\mathbf{e}}_2$ and $\hat{\mathbf{e}}_3$:

$$\mathbf{x} = \beta_1\hat{\mathbf{e}}_1 + \beta_2\hat{\mathbf{e}}_2 + \beta_3\hat{\mathbf{e}}_3 \tag{22}$$

We can reduce the number of dimensions from 3 to 1 by only transmitting the coordinate $\beta_1$ for the first principal direction $\hat{\mathbf{e}}_1 = \frac{1}{\sqrt{3}}(-1,1,1)^T$. Due to the orthonormality we can calculate this coordinate as

$$\beta_1 = \mathbf{x}^T\hat{\mathbf{e}}_1$$

This dimension reduction is optimal in the sense of transmitting as much of the signal variance as possible.

For the vector in the exercise we therefore transmit:

$$\beta_1 = \frac{1}{\sqrt{3}}\begin{pmatrix}1 & 2 & 0\end{pmatrix}\begin{pmatrix}-1\\1\\1\end{pmatrix} = \frac{1}{\sqrt{3}}$$

On the receiver side it can be reconstructed into $\frac{1}{\sqrt{3}}\hat{\mathbf{e}}_1 = \frac{1}{3}(-1,1,1)^T$

**Answer 7.14**

Maximal and minimal variance are found in the principal directions. We start with calculating the covariance matrix

$$C = \begin{pmatrix}Var(x) & Cov(x,y)\\Cov(x,y) & Var(y)\end{pmatrix} = \begin{pmatrix}s_x^2 & c_{x,y}\\c_{y,x} & s_y^2\end{pmatrix}$$

where

$$\begin{aligned}
s_x^2 &= \frac{1}{80-1}\sum_{i=1}^{80}(x_i-\bar{x})^2 = \frac{1}{79}\sum_{i=1}^{80}\frac{1}{4} = \frac{20}{79}\\
s_y^2 &= \ldots = \frac{20}{79}\\
c_{x,y} &= \frac{1}{80-1}\sum_{i=1}^{80}(x_i-\bar{x})(y_i-\bar{y}) = \frac{1}{79}\left(20\frac{1}{4} - 60\frac{1}{4}\right) = -\frac{10}{79}\\
c_{y,x} &= c_{x,y}
\end{aligned}$$

so

$$C = \frac{10}{79}\begin{pmatrix}2 & -1\\-1 & 2\end{pmatrix}$$

The principal directions (the eigenvectors) of the covariance matrix are

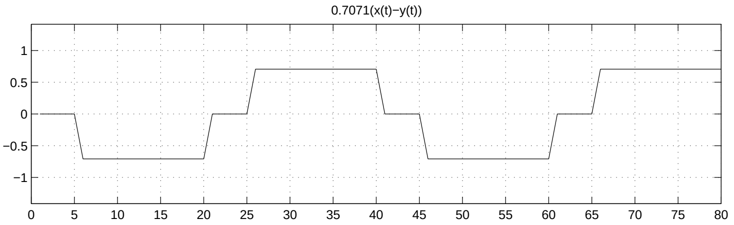
$$\hat{\mathbf{w}}_1 = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} \text{ and } \hat{\mathbf{w}}_2 = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}$$

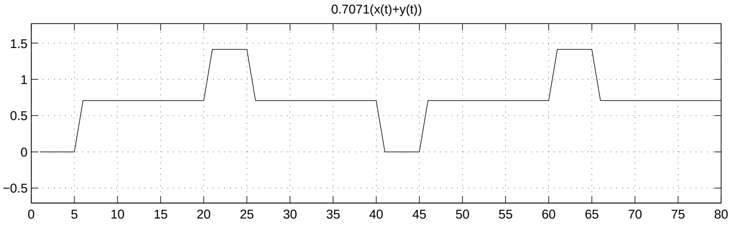
and the principal components become

$$\hat{s}_1(t) = \hat{\mathbf{w}}_1^T\begin{pmatrix}x(t)\\y(t)\end{pmatrix} = \frac{1}{\sqrt{2}}(x(t)-y(t))$$

and

$$\hat{s}_2(t) = \hat{\mathbf{w}}_2^T\begin{pmatrix}x(t)\\y(t)\end{pmatrix} = \frac{1}{\sqrt{2}}(x(t)+y(t))$$





### Answer 7.15

The ordinary eigenvalue problem is

$$\mathbf{X}\mathbf{X}^T\hat{\mathbf{w}} = \lambda\hat{\mathbf{w}}$$

If we multiply with $\mathbf{X}^T$ from the left, we get

$$\mathbf{X}^T\mathbf{X}\mathbf{X}^T\hat{\mathbf{w}} = \lambda\mathbf{X}^T\hat{\mathbf{w}}$$

which can be written as

$$\mathbf{X^T X}\hat{\mathbf{v}} = \lambda\hat{\mathbf{v}}$$

if we let $\hat{\mathbf{v}} = \mathbf{X}^T\hat{\mathbf{w}}$. Now we can solve the eigenvalue problem to obtain $\hat{\mathbf{v}}$. By using the relation above in the left step in the upmost equation, we get

$$\mathbf{X}\hat{\mathbf{v}} = \lambda\hat{\mathbf{w}}$$

### Answer 7.16

- See the lecture notes for details!
- That both classes have the same covariance matrix, i.e. that the shapes of the distributions are identical.

### Answer 7.17

Both vectors point in the X-direction, thus a classification can be made by thresholding the projection of the data onto that vector. While PCA maximizes the variance globally LDA maximizes the ratio of the between class and within class variances. This also means that LDA need some sort of preclassified training data.

## 8. Genetic Algorithms

### Answer 8.1

Genetic algorithms can be an alternative for discrete optimization problems, i.e., when the variable to optimize only can adopt discrete values so that we cannot calculate derivatives and use gradient-based methods (like gradient descent). Discrete problems are in general much more difficult to solve than continuous problems because of the lack of derivative information. Typically, to consider to use a Genetic algorithm, the number of variables should be fairly large so that it is not feasible to perform a brute-force exhaustive search that tests all possible solutions. When using a Genetic algorithm, we typically even do not know if we have found a local minimum/maximum (as we would know in the continuous case). As a Genetic algorithm is a kind of guided random search, it is also a brute force method, and it will not be feasible if the cost function is computationally costly to evaluate, because this is done many times. An advantage of Genetic algorithms is that that they are relatively simple to implement.

### Answer 8.2

For example, if a 1-point cross-over takes place after the second letter 'AB|CBC' and 'BB|CAC', the children will be 'ABCAC' and 'BBCBC'.

### Answer 8.3

Mutation is important to avoid getting trapped in a local minimum, as a mutation can take the solution to a completely different place in the search space.

### Answer 8.4

a) A natural representation is a string of seven zeros/ones, where a '1' represents that the corresponding item should be stolen and a '0' that the item should be left behind. The "Fitness"-function can, e.g., be chosen as the sum of the value of the things to bring if the weight is sufficiently low, and zero otherwise. Crossover can ,e.g., be implemented by copying genes before a certain position from the first parent and genes after this position are copied from the other parent. Mutation can, e.g., be implemented by changing the value at a random position from '1' to '0', or the opposite.

b) Given the choices above and assuming that the three original individuals are 1101001, 0100101 and 1010111, we have:

| Individual | Weight (kg) | Value ($) | "Fitness" |
|---|---:|---:|---:|
| 1101001 | 21 | 1900 | 0 |
| 0100101 | 14 | 1500 | 1500 |
| 1010111 | 20 | 1100 | 1100 |

The two last individuals will be copied and through crossover of these two, a new one will be created. The exact configuration of this individual depends on the definition of the crossover operator, but e.g. 0100111, having the weight 20, value 1600 and "fitness" 1600.

### Answer 8.5

a) The probability of such a crossover destroying a schema is given by the probability of the randomized position ending up somewhere between the fixed positions in the schema. The number of such positions are given by the length of the schema, $\delta(S)$. The position for the crossover is drawn from a rectangular distribution over the possible sites in the interval $\{1, l-1\}$ and we get the probability for survival as:

$$p_s = 1 - \delta(S)/(l-1).$$

If $p_k$ is the probability of using the crossover operator, the lower bound for a schema to survive a crossover becomes :

$$p_s \geq 1 - p_k\delta(S)/(l-1).$$

This is a lower bound because the schema might live on with another individual in the population and we have not taken that into consideration.

b) In order for the schema to avoid damage from mutation, all fixed positions in the schema must come through. The number of fixed positions are given by the order of the schema, $o(S)$. The probability to survive mutation is then:

$$p_m = (1 - p_m)^{o(S)} \approx 1 - o(S)p_m,$$

where the last approximation applies when $p_m \ll 1$. The lower bound in total for surviving both crossover and mutation becomes

$$p_s \geq 1 - p_k \frac{\delta(S)}{l-1} - o(S)p_m,$$

if we neglect the influence from the second order terms.

**Answer 8.6**

At reproduction an individual is chosen with a its relative fitness as probability, $f_i / \sum f_i$. A schema is therefore chosen with the probability $(m(S,t)f(S))/\sum f_i$, where $f(S)$ is the mean value of the fitness values for all individuals in the population having the schema. If we look at the expected number of representatives of a schema $S$ in the next generation given the number of representatives in the current generation $m(S,t)$, we get:

$$m(S, t+1) = m(S,t) \frac{f(S)}{\sum f_i} n.$$

because the size of the population is $n$ and we consequently make $n$ random samples. We can rewrite this expression with the help of $f_{ave}$, i.e. the mean fitness of the entire population:

$$m(S, t+1) = m(S,t) \frac{f(S)}{f_{ave}}.$$

Now, if a schema in average has $c \cdot f_{ave}$ greater fitness than the average in the population, this schema will grow according to the recursive expression:

$$\begin{aligned} m(S,t+1) &= m(S,t) \frac{f_{ave} + c \cdot f_{ave}}{f_{ave}} \\ &= m(S,t)\,(1+c) \\ m(S,t) &= m(S,0)\,(1+c)^t. \end{aligned}$$

I.e. the genetic algorithm leads to an exponential growth of such a schema. With the same line of reasoning we see that schemata with less fitness than average will die off from the population according to the same exponential function.

**Answer 8.7**

By combining “the schemata theorem” with the survivability calculations from exercise 8.6 we see that the expected number of representatives for a schema $S$ in the next generation are given by:

$$m(S,t+1) \geq m(S,t) \frac{f(S)}{f_{ave}} \left(1 - \frac{\delta(S)}{l-1} p_k - o(S)p_m\right).$$

Some of the parameters are given for all schemata, $p_k = 1.0$, $p_m = 0.01$, $l = 5$ and $f_{ave} = 12.5$. For the remaining parameters we can set up a table:

| Schema | $f(S)$ | $\delta(S)$ | $o(S)$ | $m(S,t)$ |
|---|---|---|---|---|
| 1 | 15 | 0 | 1 | 2 |
| 2 | 10 | 3 | 2 | 2 |

Inserted in the recursion expression we get the expected number of both the schemata as $m(S_1, t+1) = 2.376$ and $m(S_2, t+1) = 0.368$. We see that schema number two will be reduced drastically due to its length, low fitness and having many fixed positions. The opposite applies for schema number one; it's not affected by crossover, it has a low probability for being affected by mutation and it has a fitness value larger than the average.

**Answer 8.8**

a) Again we use "the Schema theorem" and we assume that schema $S_1$ has taken over the population when more than $99.5\%$ of its individuals have been equipped with this schema. According to the exercise we start with one individual having schema $S_1$, i.e. $m(S_1, 0) = 1$. In addition we know that this schema is $25\%$ better than average, which gives us $c = 0.25$. Inserting these numbers then gives us:

$$\begin{aligned} m(S_1, t) &= m(S_1, 0)\,(1+c)^t \\ 99.5 &< 1 \cdot (1+0.25)^t \\ t &> \frac{\ln 99.5}{\ln 1.25} \approx 20.6, \end{aligned}$$

i.e. the expected number of generations before all individuals in the population carries this schema is $21$.

b) Since it will be an exponential decrease of a bad schema, we hold that $S_2$ is extinct when less than $0.5\%$ of the individuals of the population carry this schema. According to the exercise we start with $60$ individuals carrying this schema, i.e. $m(S_2, 0) = 60$. In addition we know that this schema is $10\%$ worse than average, which gives us $c = -0.10$. Inserting these numbers then gives us:

$$\begin{aligned} m(S_2, t) &= m(S_2, 0)\,(1+c)^t \\ 0.5 &> 60 \cdot (1-0.1)^t \\ t &> \frac{\ln(0.5/60)}{\ln 0.99} \approx 45.4, \end{aligned}$$

i.e. the expected number of generations before no individuals in the population carries this schema is $46$.

**Answer 8.9**

a) We select as many individuals for reproduction as we have individuals in the population. The selection probability is based on their relative fitness $f(x)/\sum f(x)$. This gives us the following table which describes the performance of the initial population:

| Str. nr. | Init. pop. | $x$-value | $f(x)$ | Rel. fit. | $E\{\text{copies}\}$ |
|---|---|---|---|---|---|
| 1 | 01100 | -3 | 9 | 0.06 | 0.23 |
| 2 | 11000 | 9 | 81 | 0.52 | 2.09 |
| 3 | 01000 | -7 | 49 | 0.32 | 1.26 |
| 4 | 10011 | 4 | 16 | 0.10 | 0.41 |
| Sum | | | 155 | 1.0 | 4.0 |
| Average | | | 38.75 | 0.25 | 1.00 |
| Max | | | 81 | 0.52 | 2.09 |

The expected number of copies of the different individuals is given by the underlined values in the rightmost column in the table above. The average fitness of the initial population is 38.75, see the table above.

b) Another table shows the performance of the population after the reproduction and crossover operations:

| Str. nr. | After repr. | Partner | Cross. pos. | New pop. | $x$-value | $f(x)$ |
|---|---|---|---|---|---|---|
| 1 | 1100\|0 | 3 | 4 | 11000 | 9 | 81 |
| 2 | 11\|000 | 4 | 2 | 11011 | 12 | 144 |
| 3 | 0100\|0 | 1 | 4 | 01000 | -7 | 49 |
| 4 | 10\|011 | 2 | 2 | 10000 | 1 | 1 |
| Sum | | | | | | 275 |
| Average | | | | | | 68.8 |
| Max | | | | | | 144 |

The new table shows that both the average and maximum fitness values have improved from the first to the second generation.

c) According to the Schema Theorem we have

$$m(S, t+1) = m(S, t)\,\frac{f(S)}{f_{ave}}.$$

where $m(S, t+1)$ is the *expected* number of individuals with the schema at the time $t+1$. $m(S, t)$ is the *observed* number of individuals with the schema at the time $t$. $f(S)$ is the average fitness for the individuals with the schema at the time $t$. $f_{ave}$ is the average fitness for the whole population at the time $t$.

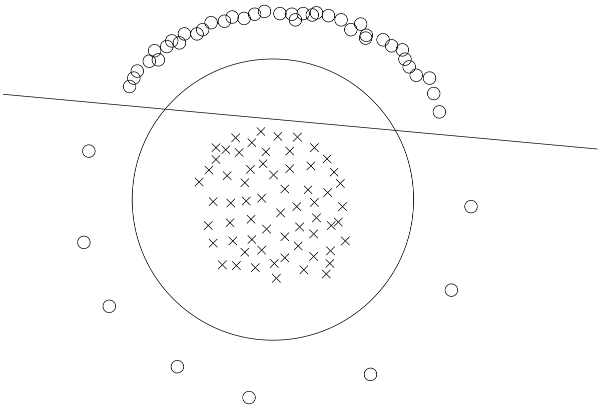
According to the solution a) above (see first table for $t = 0$), we have $f_{ave} = 38.75$. Two individuals, string number 2 and 4, have the schema $S$. Their average fitness are $f(S) = (81 + 16)/2 = 48.5$. This gives us for $t = 1$:

$$m(S, 1) = m(S, 0)\,\frac{f(S)}{f_{ave}} = 2 \cdot \frac{48.5}{38.75} \approx 2.5$$

## Kernel Methods

### Answer 8.10

Since the linear classifier produce a line the placement is pretty straight forward and some misclassifications will occur. With access to the squared feature values $\varphi_1 = x_1^2$ and $\varphi_2 = x_2^2$, a linear classifier with a circular boundary in the original space is easily generated: $f(\varphi_1, \varphi_2) = sign(w_1\varphi_1 + w_2\varphi_2) = sign(w_1 x_1^2 + w_2 x_2^2)$ that provides perfect classification.



### Answer 8.11

a) The scalar product is defined as $\mathbf{x}_1 \cdot \mathbf{x}_2 = \mathbf{x}_1^T \mathbf{x}_2$.

b) The length of the vector $\|\mathbf{x}\| = \sqrt{\|\mathbf{x}\|^2} = \sqrt{\mathbf{x}^T\mathbf{x}} = \sqrt{\mathbf{x} \cdot \mathbf{x}}$.

c) The distance $\|\mathbf{x}_1 - \mathbf{x}_2\| = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T(\mathbf{x}_1 - \mathbf{x}_2)} = \sqrt{\mathbf{x}_1^T\mathbf{x}_1 + \mathbf{x}_2^T\mathbf{x}_2 - 2\mathbf{x}_1^T\mathbf{x}_2} = \sqrt{\mathbf{x}_1 \cdot \mathbf{x}_1 + \mathbf{x}_2 \cdot \mathbf{x}_2 - 2\mathbf{x}_1 \cdot \mathbf{x}_2}$

d) The vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ span a triangle. The Law of Cosines from trigonometry states that if the lengths of the triangle sides are $a$, $b$ and $c$, the following relation holds: $c^2 = a^2 + b^2 - 2ab\cos\theta$. If $\mathbf{x}_1$ represents $a$ and $\mathbf{x}_2$ represents $b$, $c$ is $\mathbf{x}_1 - \mathbf{x}_2$. The Law of Cosines then gives $\|\mathbf{x}_1 - \mathbf{x}_2\|^2 = \|\mathbf{x}_1\|^2 + \|\mathbf{x}_2\|^2 - 2\|\mathbf{x}_1\|\|\mathbf{x}_2\|\cos\theta$. Rearranging and using the results above gives the angle $\theta$ between $\mathbf{x}_1$ and $\mathbf{x}_2$ in terms of scalar products $\cos\theta = \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{\sqrt{\mathbf{x}_1 \cdot \mathbf{x}_1}\sqrt{\mathbf{x}_2 \cdot \mathbf{x}_2}}$.

### Answer 8.12

The kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ defines a scalar product between two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ that have been mapped to a (usually high-dimensional) feature space via a function $\varphi()$.

$$\varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j) = \text{ some function.}$$

The clue here is that we never deal with $\varphi(\mathbf{x}_i)$ and $\varphi(\mathbf{x}_j)$ explicitly, i.e., we never calculate them or store them in the computer, we only deal the scalar products defined by $k(\mathbf{x}_i, \mathbf{x}_j)$. As the kernel function defines the scalar product, which in turn defines the distance between vectors according to 8.11, the kernel function can also be seen as a similarity function.

### Answer 8.13

A kernel function defines the inner product $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \Phi(\mathbf{x}_1)^T\Phi(\mathbf{x}_2)$ in the new feature space. Thus, $\kappa(\mathbf{x}_1, \mathbf{x}_2)$ specifies the feature space by defining how distances and angles are measured, instead of explicitly stating the mapping function $\Phi(\mathbf{x})$.

The distance between $\mathbf{x}_1$ and $\mathbf{x}_2$ in the new feature space is

$$\begin{aligned}\|\Phi(\mathbf{x}_1)-\Phi(\mathbf{x}_2)\| &= \sqrt{(\Phi(\mathbf{x}_1)-\Phi(\mathbf{x}_2))^T\,(\Phi(\mathbf{x}_1)-\Phi(\mathbf{x}_2))} \\ &= \sqrt{\Phi(\mathbf{x}_1)^T\Phi(\mathbf{x}_1)-2\Phi(\mathbf{x}_1)^T\Phi(\mathbf{x}_2)+\Phi(\mathbf{x}_2)^T\Phi(\mathbf{x}_2)} \\ &= \sqrt{\kappa(\mathbf{x}_1,\mathbf{x}_1)-2\,\kappa(\mathbf{x}_1,\mathbf{x}_2)+\kappa(\mathbf{x}_2,\mathbf{x}_2)} \\ &= \sqrt{1-2\,e^{-\frac{1}{4}}+1} = 0.6651\end{aligned}$$

**Answer 8.14**

$$\varphi(\mathbf{x})^T\varphi(\mathbf{y}) = x_1^2y_1^2+x_2^2y_2^2+2x_1x_2y_1y_2 = (x_1y_1+x_2y_2)^2 = \left(\mathbf{x}^T\mathbf{y}\right)^2 \tag{23}$$

**Answer 8.15**

a) The kernel matrix $\mathbf{K}$ contains all scalar products between all training data feature vectors (generally mapped through a nonlinear function $\varphi()$). For example, if we have three training data samples $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$, the kernel matrix is

$$\mathbf{K} = \begin{pmatrix}\varphi(\mathbf{x}_1)^T\varphi(\mathbf{x}_1) & \varphi(\mathbf{x}_1)^T\varphi(\mathbf{x}_2) & \varphi(\mathbf{x}_1)^T\varphi(\mathbf{x}_3)\\ \varphi(\mathbf{x}_2)^T\varphi(\mathbf{x}_1) & \varphi(\mathbf{x}_2)^T\varphi(\mathbf{x}_2) & \varphi(\mathbf{x}_2)^T\varphi(\mathbf{x}_3)\\ \varphi(\mathbf{x}_3)^T\varphi(\mathbf{x}_1) & \varphi(\mathbf{x}_3)^T\varphi(\mathbf{x}_2) & \varphi(\mathbf{x}_3)^T\varphi(\mathbf{x}_3)\end{pmatrix} = \begin{pmatrix}k(\mathbf{x}_1,\mathbf{x}_1) & k(\mathbf{x}_1,\mathbf{x}_2) & k(\mathbf{x}_1,\mathbf{x}_3)\\ k(\mathbf{x}_2,\mathbf{x}_1) & k(\mathbf{x}_2,\mathbf{x}_2) & k(\mathbf{x}_2,\mathbf{x}_3)\\ k(\mathbf{x}_3,\mathbf{x}_1) & k(\mathbf{x}_3,\mathbf{x}_2) & k(\mathbf{x}_3,\mathbf{x}_3)\end{pmatrix}.$$

One can note that $\mathbf{K}$ is symmetric as $\mathbf{x}_i^T\mathbf{x}_j = \mathbf{x}_j^T\mathbf{x}_i$.

b) $10 \times 10$.

**Answer 8.16**

We have the following kernel matrix

$$\mathbf{K} = \begin{pmatrix}1 & 1 & 0\\ 1 & 2 & 2\\ 0 & 2 & 4\end{pmatrix} = \begin{pmatrix}\mathbf{x}_1^T\mathbf{x}_1 & \mathbf{x}_1^T\mathbf{x}_2 & \mathbf{x}_1^T\mathbf{x}_3\\ \mathbf{x}_2^T\mathbf{x}_1 & \mathbf{x}_2^T\mathbf{x}_2 & \mathbf{x}_2^T\mathbf{x}_3\\ \mathbf{x}_3^T\mathbf{x}_1 & \mathbf{x}_3^T\mathbf{x}_2 & \mathbf{x}_3^T\mathbf{x}_3\end{pmatrix}.$$

We have three training data vectors that we denote $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$. Using the derivations in 8.11 above, we can extract the following information from $\mathbf{K}$:

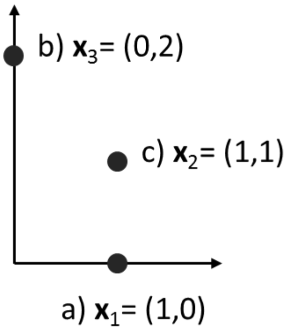
- $\|\mathbf{x}_1\| = \sqrt{\mathbf{x}_1\cdot\mathbf{x}_1} = 1$, i.e., this point is 1 unit length from the origin of the space. Similarly, $\|\mathbf{x}_2\| = \sqrt{2}$ and $\|\mathbf{x}_3\| = 2$.
- The distances $\|\mathbf{x}_1-\mathbf{x}_2\| = 1$, $\|\mathbf{x}_1-\mathbf{x}_3\| = \sqrt{5}$ and $\|\mathbf{x}_2-\mathbf{x}_3\| = \sqrt{2}$.
- The angle between $\mathbf{x}_1$ and $\mathbf{x}_2$ is 45°. The angle between $\mathbf{x}_1$ and $\mathbf{x}_3$ is 90° (orthogonal as the scalar product is 0).

Now we can reconstruct the data as follows (see figure c below):

a) Let's begin by placing $\mathbf{x}_1$ at any point with distance 1 from the origin.

b) Next, place $\mathbf{x}_3$ on an axis orthogonal (angle 90°) to $\mathbf{x}_1$, a distance 2 from the origin.

c) Finally, $\mathbf{x}_2$ must be at one of 2 possible positions that are at an angle 45° and a distance 1 relative $\mathbf{x}_1$. As the distance between $\mathbf{x}_2$ and $\mathbf{x}_3$ must also be $\sqrt{2}$, there is only one possible point left. Note that there is an infinite number of solutions depending on where we start with $\mathbf{x}_1$.



**Answer 8.17**

a)

$$\mathbf{K} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

b) The vectors in the feature space become

$$\varphi(\mathbf{x}_1) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \varphi(\mathbf{x}_2) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \varphi(\mathbf{x}_3) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

The average vector in the feature space then becomes

$$\overline{\varphi}(\mathbf{x}) = \frac{1}{3} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

The centered feature vectors the become $\varphi'(\mathbf{x}_i) = \varphi(\mathbf{x}_i) - \overline{\varphi}(\mathbf{x})$:

$$\varphi'(\mathbf{x}_1) = \frac{1}{3} \begin{bmatrix} 2 \\ 0 \\ 0 \\ -1 \end{bmatrix}, \quad \varphi'(\mathbf{x}_2) = \frac{1}{3} \begin{bmatrix} -1 \\ 0 \\ 0 \\ -1 \end{bmatrix}, \quad \varphi'(\mathbf{x}_3) = \frac{1}{3} \begin{bmatrix} -1 \\ 0 \\ 0 \\ 2 \end{bmatrix}$$

The centered kernel matrix then becomes

$$\text{K'} = \frac{1}{9} \begin{bmatrix} 5 & -1 & -4 \\ -1 & 2 & -1 \\ -4 & -1 & 5 \end{bmatrix}$$

This is the same matrix as we get if we subtract the row- and column averages from K and add the total average.

**Answer 8.18**

The problem must be possible to formulate in terms of scalar products between the samples so that we can swap in a nonlinear kernel function here, the so-called kernel trick. A first step to do this is typically to show that the optimal solution can be written as a linear combination of the training data vectors $\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i$ for some values of the $\alpha$'s.

**Answer 8.19**

The original cost function is:

$$\min \|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w}$$

subject to the constraint $y_i \left(\mathbf{w}^T\mathbf{x}_i + w_0\right) \geq 1$ for all $i$.

The kernelized function can be derived for problems for which it can be shown that the optimal solution $\mathbf{w}^*$ lies in the span of the input training data, i.e., $\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i$, where $\alpha_i$ are some number, $\mathbf{x}_i$ is a traing data example and $N$ is the number of training examples. Inserting this relationship in the original cost function yields $\mathbf{w}^T\mathbf{w} = \sum_{i=1}^{N} \sum_{i=j}^{N} \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j$ (which can also be written in a vector form $\boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha}$ using a kernel matrix $\mathbf{K}$). The entire kernelized cost function is

$$\min \sum_{i=1}^{N} \sum_{i=j}^{N} \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to the constraint $y_i \left( \alpha_i \mathbf{x}_i^T \mathbf{x}_i + \alpha_0 \right) \geq 1$ for all $i$.

In non-linear kernel methods, the inner products $\mathbf{x}_i^T \mathbf{x}_j$ is replaced with a non-linear kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j)$.

### Answer 8.20

The classification function in SVM has the value ±1 for the support vectors. Hence, $\mathbf{x}$ is a support vector since $f(\mathbf{x}) = 1$

## 9. Reinforcement learning

### Answer 9.1

In supervised learning, the feedback is the correct action or class for every input, situation or state. In reinforcement learning, a scalar feedback (reward or punishment) is obtained for taking certain actions or reaching certain states, but the correct or optimal action is not given to the learning system.

### Answer 9.2

The "temporal credit assignment problem" refers to the problem of assessing individual actions in a sequence of actions when the reward/punishment comes delayed. For example, which were the winning moves that led to the victory in a game?

### Answer 9.3

The V-function describes the value (expected reward) that will be obtained in the future when being in certain state, when following a certain policy (which action to take in every state). Each policy has a different value function. The Q-function describes the value for each action in each state, given that the optimal policy is followed after the action has been taken.

### Answer 9.4

The discount factor controls the trade-off between optimizing for immediate rewards and long term rewards.

### Answer 9.5

*$\epsilon$-greedy exploration* means that a random action (exploration) is made with probability $\epsilon$ and a greedy action (exploitation), i.e., an action that maximizes the reward, is made with probability $1 - \epsilon$.

### Answer 9.6

The definition of the optimal Q-function is given by:

$$\begin{aligned} Q^*(x,a) &= r(x,a) + \gamma V^*(g(x,\mu^*(x))) \\ &= r(x,a) + \gamma \min_b Q^*(g(x,\mu^*(x)),b) \end{aligned}$$

We can find a solution by traversing the state graph recursively starting from the end in the first exercise, since we know the value of $V^*(4)$.

a) According to the state graph, $V^*(4) = 0$.

$$\begin{aligned} Q^*(3,\mathit{up}) = 1 + \gamma V^*(4) = 1 = V^*(3) \\ Q^*(2,\mathit{up}) = 0 + \gamma V^*(3) = \gamma = V^*(2) \\ Q^*(1,\mathit{up}) = 0 + \gamma V^*(2) = \gamma^2 = V^*(1) \end{aligned}$$

b) In this exercise there is no end. The system will however always end up in state 3 and stay there. For $\gamma \neq 1$ we therefore have

$$\begin{aligned} Q^*(3,\mathit{same}) = 1 + \gamma Q^*(3,\mathit{same}) \Leftrightarrow \\ Q^*(3,\mathit{same}) = 1/(1-\gamma) = V^*(3) \\ Q^*(2,\mathit{up}) = 0 + \gamma V^*(3) = \gamma/(1-\gamma) = V^*(2) \\ Q^*(1,\mathit{up}) = 0 + \gamma V^*(2) = \gamma^2/(1-\gamma) = V^*(1) \end{aligned}$$

We can consequently conclude that we have to use a discounting factor $\gamma \neq 1$ in order to avoid infinite Q- and V-functions.

c) According to the state graph, $V^*(4) = 0$ and $V^*(5) = 0$. We can therefore calculate (in order)

$$\begin{aligned}
Q^*(3, up) = 2 + \gamma V^*(4) = 2 = V^*(3) \\
Q^*(2, up) = 0 + \gamma V^*(3) = 2\gamma \\
Q^*(2, right) = 1 + \gamma V^*(5) = 1 \\
V^*(2) = \max_a Q^*(2, a) = \left\{ \begin{array}{ll} 2\gamma & \text{om } \gamma > \frac{1}{2} \\ 1 & \text{om } \gamma \leq \frac{1}{2} \end{array} \right. \\
Q^*(1, up) = 0 + \gamma V * (2) = \left\{ \begin{array}{ll} 2\gamma^2 & \text{om } \gamma > \frac{1}{2} \\ \gamma & \text{om } \gamma \leq \frac{1}{2} \end{array} \right. = V^*(1)
\end{aligned}$$

For this exercise we can conclude that we get different optimal policies depending on the value of $\gamma$. A low $\gamma$ value corresponds to upweighted rewards in the short term and the system therefore turns right immediately in spite of this causing a lower unweighted reward sum over time for this policy.

**Answer 9.7**

The definition of the optimal Q-function is given by:

$$\begin{aligned}
Q^*(x, a) &= r(x, a) + \gamma V^*(g(x, \mu^*(x))) \\
&= r(x, a) + \gamma \min_b Q^*(g(x, \mu^*(x)), b)
\end{aligned}$$

a) For system A, there is only one choice of policy, to continue forward. For system B, we can see that the optimal policy is to get to the upper loop as soon as possible, and then continue around the loop, since this will maximize the rewards over time.

b) Since there is only one action in every state, the Q- and V-functions will be equal. By application of the definition of the optimal Q-function, we get the following:

$$\begin{aligned}
V^*(1) = Q^*(1, forward) = 3 + \gamma V^*(2) \\
V^*(2) = Q^*(2, forward) = 0 + \gamma V^*(3) \\
V^*(3) = Q^*(3, forward) = 0 + \gamma V^*(4) \\
V^*(4) = Q^*(4, forward) = 0 + \gamma V^*(5) \\
V^*(5) = Q^*(5, forward) = 0 + \gamma V^*(6) \\
V^*(6) = Q^*(6, forward) = 0 + \gamma V^*(1)
\end{aligned}$$

Which gives

$$\begin{aligned}
V^*(1) = Q^*(1, forward) = \frac{3}{1 - \gamma^6} \\
V^*(2) = Q^*(2, forward) = \frac{3\gamma^5}{1 - \gamma^6} \\
V^*(3) = Q^*(3, forward) = \frac{3\gamma^4}{1 - \gamma^6} \\
V^*(4) = Q^*(4, forward) = \frac{3\gamma^3}{1 - \gamma^6} \\
V^*(5) = Q^*(5, forward) = \frac{3\gamma^2}{1 - \gamma^6} \\
V^*(6) = Q^*(6, forward) = \frac{3\gamma^1}{1 - \gamma^6}
\end{aligned}$$

c) From exercise A we know that the optimal policy when standing in some of the states in the upper loop, i.e. 1, 2, 3 or 6, is to go around the loop. We therefore start calculating the V- and Q-functions in this loop.

$$
\begin{aligned}
V^*(1) = Q^*(1, forward) = 3 + \gamma V^*(2) \\
V^*(2) = Q^*(2, forward) = 0 + \gamma V^*(3) \\
V^*(3) = Q^*(3, shortcut) = 0 + \gamma V^*(6) \\
V^*(6) = Q^*(6, forward) = 0 + \gamma V^*(1)
\end{aligned}
$$

Which in the same way as in B gives

$$
\begin{aligned}
V^*(1) = Q^*(1, forward) &= \frac{3}{1-\gamma^4} \\
V^*(2) = Q^*(2, forward) &= \frac{3\gamma^3}{1-\gamma^4} \\
V^*(3) = Q^*(3, shortcut) &= \frac{3\gamma^2}{1-\gamma^4} \\
V^*(6) = Q^*(6, forward) &= \frac{3\gamma^1}{1-\gamma^4}
\end{aligned}
$$

Now when we know the value of the V-function for state 6, we can calculate the V- and Q-functions for the state 5,4 and 3 recursively.

$$
\begin{aligned}
V^*(5) = Q^*(5, forward) = 0 + \gamma V^*(6) &= \frac{3\gamma^2}{1-\gamma^4} \\
V^*(4) = Q^*(4, forward) = 0 + \gamma V^*(5) &= \frac{3\gamma^3}{1-\gamma^4} \\
Q^*(3, forward) = 0 + \gamma V^*(4) &= \frac{3\gamma^4}{1-\gamma^4}
\end{aligned}
$$

**Answer 9.8**

Since we are dealing with stochastic reward functions we are forced to include a expectation value in our expression for optimal Q-function. (In the lab we solved this by choosing a low value for the learning rate $\eta$.)
The definition of the optimal Q-function is given by:

$$
\begin{aligned}
Q^*(x, a) &= E\{r(x, a)\} + \gamma V^*(g(x, \mu^*(x))) \\
&= E\{r(x, a)\} + \gamma \min_b Q^*(g(x, \mu^*(x)), b)
\end{aligned}
$$

We can find a solution by traversing the state graph recursively starting from the end in the first exercise, since we know the value of $V^*(4)$.

a) According to the state graph, $V^*(4) = 0$.

$$
\begin{aligned}
Q^*(3, up) = 3 + \gamma V^*(4) = 3 = V^*(3) \\
Q^*(2, up) = (3 \cdot 0.70 + 1 \cdot 0.30) + \gamma V^*(3) = 2.4 + 3\gamma = V^*(2) \\
Q^*(1, up) = 0 + \gamma V^*(2) = 2.4\gamma + 3\gamma^2 = V^*(1)
\end{aligned}
$$

b) According to the state graph, $V^*(3) = 0$ and $V^*(5) = 0$. We can therefore calculate (in order)

$$\begin{aligned} Q^*(4, up) &= 3 + \gamma V^*(5) = 3 = V^*(4) \\ Q^*(2, up) &= (3 \cdot 0.90 - 25 \cdot 0.10) + \gamma V^*(3) = 0.2 \\ Q^*(2, left) &= 0 + \gamma V^*(4) = 3\gamma \\ V^*(2) &= \max_a Q^*(2, a) = \begin{cases} 0.2 & \text{om } \gamma \le \frac{2}{30} \\ 3\gamma & \text{om } \gamma > \frac{2}{30} \end{cases} \\ Q^*(1, up) = 0 + \gamma V * (2) &= \begin{cases} 0.2\gamma & \text{om } \gamma \le \frac{2}{30} \\ 3\gamma^2 & \text{om } \gamma > \frac{2}{30} \end{cases} = V^*(1) \end{aligned}$$

For this exercise we can conclude that we get different optimal policies depending on the value of $\gamma$. A low $\gamma$ value corresponds to upweighted rewards in the short term and the system therefore goes straight up in spite of this causing a lower unweighted reward sum over time for this policy than if it had turned left in the crossing.

**Answer 9.9**

The formula for the optimal Q-function is given by:

$$\begin{aligned} Q^*(x, a) &= r(x, a) + \gamma V^*(g(x, \mu^*(x))) \\ &= r(x, a) + \gamma \max_b Q^*(g(x, \mu^*(x)), b) \end{aligned}$$

a) Counting backwards we get:

$$\begin{aligned} Q^*(3, right) &= 150 + 0 \cdot \gamma = 150 \\ Q^*(2, down) &= \gamma Q^*(3, right) - 50 = 150\gamma - 50 \\ Q^*(1, up) &= \gamma Q^*(2, down) - 50 = 150\gamma^2 - 50\gamma - 50 \\ Q^*(1, right) &= p \cdot \gamma Q^*(3, right) + 0 - 150 \cdot (1 - p) \end{aligned}$$

b) Using the result from A we get:

$$\begin{aligned} V^*(5) &= 0 \\ V^*(4) &= 0 \\ V^*(3) &= Q^*(3, right) = 150 + 0 = 150 \\ V^*(2) &= Q^*(2, down) = Q^*(3, right) - 50 = 150 - 50 = 100 \\ p \le \frac{2}{3} \Rightarrow V^*(1) &= Q^*(1, up) = Q^*(2, down) - 50 = 150 - 50 - 50 = 50 \\ p \ge \frac{2}{3} \Rightarrow V^*(1) &= Q^*(1, right) = p \cdot Q^*(3, right) + 0 - 150 \cdot (1 - p) = (2p - 1) \cdot 150 \end{aligned}$$

c) If $0 < \alpha < 1$ the system will eventually converge to the average reward for the action ”right” from ”1”, giving us an optimal policy. But if $\alpha = 1$ only the reward of last action taken will be remembered, and the policy might never converge to a stable state as long as we are exploring. For example when $2/3 < p < 1$ it is optimal to move right from node 1, since this is the path where we can expect the maximum reward over time, but sometimes there will be a huge negative reward. If we were to use $\alpha = 1$ in this example the policy will shift every time the we try to move right from ”1” and end up where we did not expect, and thus the final policy will only be dependent on when we stop training and not on which path is optimal.