OST
Ostschweizer
Fachhochschule

Lab 7

FTP MachLe MSE
FS 2022

Support Vector Machines (SVM)

MSE MachLe
WÜRC

Lernziele/Kompetenzen

- You know the primal form of a *separating hyperplane classifier*:

$$f_{w,b}(x) = \text{sign}(\langle w, x\rangle + b)$$

- You can calculate the *geometric margin* γ_i of a given sample point x_i for a separating hyperplane classifier $f_{w,b}$ with given parameters w and b.

$$\gamma_i := y_i \cdot \left(\frac{\langle w, x_i\rangle + b}{\|w\|}\right)$$

- You can explain the progression of the SVM family from *maximal margin classifier* to *support vector classifier* (SVC) and finally to *support vector machines* (SVM) that use the kernel trick.
- You can use SVM successfully on tutorial style examples, including (cross-validated) parameter *grid search*.
- You know, that SVM use only a subset of training points in the decision function (called *support vectors*), so they are memory efficient. SVM are still effective in cases where number of dimensions d is greater than the number of samples N.
- You know what a *kernel function* $K(x_i, x_j)$ is and how it is related to a *feature transform* $\phi(x)$ (Mercer Theorem). Different kernel functions can be specified for the decision function of a SVM. Common kernels are the *radial basis function* kernel, the *linear* kernel and the *polynomial* kernel.

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j)\rangle_{\mathcal{H}}$$

- You can explain the effect of the C and γ parameters for a SVM with a radial basis function kernel (rbf-kernel). If you have a lot of noisy observations, C should be small. *Decreasing* C corresponds to *higher bias*, *less variance*, *larger margin*, *higher tolerance for misclassification*, less confidence in the dataset (more noise).

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$$

- You know that you can always *apply the kernel trick* whenever there appears an scalar product $\langle x_i, x_j\rangle$ of two feature vectors x_i and x_j in a given loss function $\mathcal{L}$. You know that by applying the kernel trick to the dual loss function of the separating hyperplane classifier linearly non-separable classes become separable if the chosen dimension is high enough.

1. Feature Transform ϕ of the Polynomial Kernel [A,I]

The inhomogeneous quadratic polynomial kernel $K(\vec{z}_1, \vec{z}_2)$ is given by:

$$K(\vec{z}_1, \vec{z}_2) = (\vec{z}_1 \cdot \vec{z}_2 + 1)^2$$

a) Calculate the feature transform $\phi(\vec{z}) : \mathbb{R}^2 \to \mathbb{R}$ associated with the inhomogeneous quadratic polynomial kernel $K(\vec{z}_1, \vec{z}_2)$ explicitely for two-dimensional ($d = 2$) features $\vec{z} = \begin{pmatrix} x \\ y \end{pmatrix}$, i.e. find a feature transform $\phi(\vec{z})$ and an associated HILBERT space $\mathcal{H}$ such that the kernel K can be expressed as a scalar product of the transformed features $\phi(\vec{z})$.

$K(\vec{z}_1, \vec{z}_2) = (\vec{z}_1 \cdot \vec{z}_2 + 1)^2 = \langle \phi(\vec{z}_1), \phi(\vec{z}_2) \rangle_{\mathcal{H}}$

b) What is the dimensionality $\dim(\mathcal{H})$ of the associated HILBERT space, i.e. the dimensionality D of the transformed features $\phi(\vec{z})$.

c) What is the dimensionality $\dim(\mathcal{H})$ of the associated HILBERT space for the radial basis function kernel?

2. Simple SVM example [A,I]

The lecture introduced two hyper parameters of SVM: C (to penalize misclassifications) and γ (in case of the Gaussian kernel). In this exercise, you will play around with an already complete Juypter notebook to experience the effect of these parameters.

In this exercise, you will follow an experiment that tries to characterize the **cross validation (CV)** and **leave-one-out cross validation (LOOCV)** method in terms of bias and variance. This exercise further serves to make you familiar with `Jupyter` notebooks. If you haven't used `Jupyter` notebooks yet, have a look at the cheat sheets and the quick introduction given in lecture one.

a) Open the IPython Notebook `ML07_A2_SVM_Simple.jpynb` in your browser. You find it on the moodle platform.

b) Make yourself familiar with the `scikit-learn` syntax of training and applying an (SVM-) classifier in cells `In[3]` and `In[4]` as well as with the evaluation in cell `In[5]`.

c) Try some parameter values for C on your own in cell `In[5]`. What is a good range of values? What is a good strategy to search for C, with respect to sampling a certain range of possible values?
A naïve strategy could be using equidistant points, e.g.: $C = 1, 2, 3, 4, 5, 6, 7, \ldots$. An often better strategy is to start with very small values and double/decuple/... the last value to use it as the next candidate. Other schemes such as *logarithmic* or *exponential* sequences are also possible.

d) Execute all cells until the end. How does the grid search parameter optimization in cell `In[6]` work?

- Does it find similar values as you did by hand?
- What strategy for sampling possible parameter values is reflected in the given values of the data structure `tuned_parameters` ?

e) Compare the CV accuracy on the training set (after cell `In[6]`) with the final scores on the test set (after cell `In[7]`).

- What is the quality of the estimated true error using CV for this particular data set?
- Which conclusions can you draw for the general case?

Further investigation [optional]: To deepen your understanding of SVM for practical applications, you may want to have a look at `scikit-learn`'s face recognition example: `https://scikit-learn.org/0.16/auto_examples/applications/face_recognition.html`. It contains code to load the *Labelled faces in the wild* data from the web, perform SVM parameter grid search, and evaluate the trained model. A complete run of the program takes approximately 5 minutes (with the most time spent on downloading the 233MB of images).

3. Predicting Diabetes using a SVM [A,II]

The data set `pima-indians-diabetes.csv` contains medical indicators for diabetes type II collected from 768 patients of the indigenous Pima tribe (Phoenix, AZ). The subjects were between 21 and 81 years old. The following characteristics were recorded:

`NumTimesPrg`	Number of pregnancies
`PlGlcConc`	Blood sugar 2h after oral uptake of sugar (oGTT) (mg/dl)
`BloodP`	Diastolic blood pressure (mm Hg)
`SkinThick`	thickness of the skin fold at the triceps (mm)
`TwoHourSerIns`	Insulin concentration after 2h oGTT (μIU/mg)
`BMI`:	Body Mass Index (kg/m^2)
`DiPedFunc`	Hereditary predisposition
`Age`	Age (years)
`HasDiabetes`	Diagnosis Diabetes Type II (target y)

The goal of the exercise is to create an optimal SVM classifier using a radial basis function kernel for the prediction of diabetes Type II. All features will be used for classification and the parameters of the radial basis function kernel (γ, C) will be tuned using a cross-validated grid search.

Open the Jupyter notebook `ML07_A3_PimaIndians_TEMPLATE.jpynb` and execute the first three cells that load the dataset into a `pandas dataframe (df)` called `pima`. The classification goal is to make the diagnosis of type II diabetes based on the factors, i.e. to make a prediction model for the variable `y=HasDiabetes` using a support vector machine (SVM) with a radial basis function kernel. Before doing this, some *data preprocessing* will be necessary.

a) Calculate the percentage of patients with diabetes and display a statistics using `pima.describe`.

b) Exclude samples with zero entries or missing values. Replace the zero values with `np.nan` and print again a statstical description of the dataset using `df.describe()`. Then drop `np.nan` values using `df.dropna()`.

c) Plot a *histogram* of the of each feature and the target using `df.hist()`.

d) *Split* the data in 80% training and 20% test data. Use `train_test_split` from `sklearn.model_sele` If you feed a `pandas.Dataframe` as an input to the method, you will also get `pandas.Dataframes` as output for the training and test features.

e) *Standardize* the features using the `StandardScaler` from `sklearn.preprocessing` and display the histograms of the features again.

f) Train a SVM classifier with a radial basis function kernel ($\gamma = 1, C = 1$) and determine the accuracy on the test data.

g) Perform a *cross validated grid search* `GridSearchCV` to find the best parameters for γ and C and determine the best parameters and score.

- vary the value of C in the range in a logarithmic scale from 10^{-3} to 10^{+3} (7 steps)
- vary the value of γ in the range in a logarithmic scale from 10^{-3} to 10^{+3} (7 steps)
- print a classification report and a confusion matrix of the best classifier using `classification_report` and `confusion_matrix` from `sklearn.metrics`.

h) *Plot the resulting decision boundary* and margins of the SVM classifier for different values of γ and C as contour plot in 2D. Use the following two features as the only features for the prediction such that we can display the decision boundaries in a 2D plot.

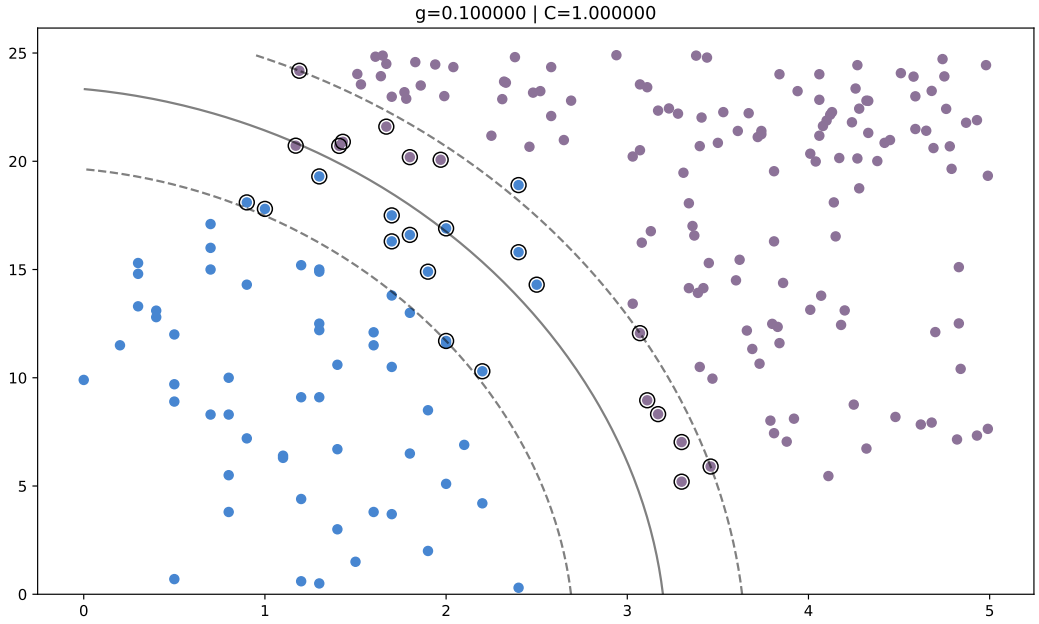
- feature 1: 'TwoHourSerIns' (x1-axis)
- feature 2: 'Age' (x2-axis)

Vary the C parameters in the following ranges: `C_range = [1e-2, 1, 1e2]` and `gamma_range = [1e-1, 1, 1e1]` and visualize the decision boundary and the margins. You can use the helper function `PlotDecisionBoundary(model, X2D, y)` which is given in the template.

4. Polynomial Kernel SVM [A,I]

Some data is almost linearly separable. The goal of this exercise is to generate nearly linearly-separable data and fit a polynomial kernel support vector machine tho the dataset and visualize the misclasifications, the decision boundary as well as the margins as function of the two parameters γ and C.

Open the IPython Notebook `ML07_A4_PolynomialKernel_TEMPLATE.jpynb` in your browser. You find it on the moodle platform. Cells 1-3 generate an almost linearly separable dataset $\mathcal{S} = \{X, y\}_{i=1...M}$ with $N = 100$.



a) Split the data in 70% training and 30% test data using `train_test_split` from `sklearn.model_selection`.

b) Plot the training data as scatter plot using different colors for both classes.

c) Fit an SVM with a second-degree polynomial kernel using $\gamma = 0.1$ and $C = 1$.

d) Plot the margins and decision boundary of the classifier.
Use the function `PlotDecisionBoundary(model,X,y)` that is provided below. The input arguments the instance of the trained `model`, the 2D array of the featues X and the 1D array of the target `y`.

```
def PlotDecisionBoundary(model, X,y):

#plot decision boundary for model in case of 2D feature space
x1=X[:,0]
x2=X[:,1]
# Create grid to evaluate model
xx = np.linspace(min(x1), max(x1), len(x1))
yy = np.linspace(min(x1), max(x2), len(x2))
YY, XX = np.meshgrid(yy, xx)
xy = np.vstack([XX.ravel(), YY.ravel()]).T

train_size = len(x1)

# Assigning different colors to the classes
colors = y
colors = np.where(colors == 1, '#8C7298', '#4786D1')

# Get the separating hyperplane
```

```
Z = model.decision_function(xy).reshape(XX.shape)

fig, ax = plt.subplots(figsize=(12, 7))
ax.scatter(x1, x2, c=colors)

# Draw the decision boundary and margins
ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1], alpha=0.5,
                                    linestyles=['--', '-', '--'])

# Highlight support vectors with a circle around them
ax.scatter(model.support_vectors_[:, 0], model.support_vectors_[:, 1],
                                     s=100,
linewidth=1, facecolors='none', edgecolors='k')
plt.title('g=%f | C=%f' % (model.gamma,model.C))
plt.show()
```

e) Vary the hyperparameter γ logarithmically in the range from 10^{-2} to 10^{+2} in 5 steps. Set $C = 1$ and plot for each value of the parameter γ the decision boundary and the marigns.

f) Vary the hyperparameter C logarithmically in the range from 10^{0} to 10^{4} in 5 steps. Set $\gamma = 0.01$ and plot for each value of the parameter C the decision boundary and the marigns.

5. General Questions about SVM [A,I]

Try to answer the following questions with 2-4 sentences.

a) What is the fundamental idea behind SVMs?

b) What is a *support vector*?

c) Why is it important to *scale* the inputs when using SVM?

d) Can an SVM output a confidence score when it classifies an instance? What about a probability?

e) Should you use the *primal* or the *dual* form of the SVM problem to train a model on a training set with millions of instances and hundreds of features?

f) Say you trained an SVM classifier with an RBF kernel. It seems to underfit the training set: should you increase or decrease γ? What about C?

OST
Ostschweizer
Fachhochschule

Lab 8

Probabilistic Reasoning - Gaussian distribution and Bayes Theorem

FTP MachLe MSE
FS 2022

Machine Learning
WÜRC

The central *paradigm of probabilistic reasoning* is to identify all relevant variables $x_1, \ldots, x_N$ in the environment,and make a probabilistic model $p(x_1, \ldots, x_N)$ of their interaction. Reasoning (*inference*) is then performed by introducing *evidence* that sets variables in known states, and subsequently computing probabilities of interest, *conditioned on this evidence*. The rules of probability, combined with Bayes' rule make a reasoning system complete.

After this unit, ...

Lernziele/Kompetenzen

- you have repeated the *basic rules of probability theory*.
- you know the difference between a *joint* and a *conditional probability* distribution.
- you know how to apply *Bayes Theorem* to calculate the *posterior* probability distribution for simple discrete examples. You can name the *prior* probability distribution, the *likelihood function*, the *evidence*, and you know how to *marginalize* over a joint probability distribution.
- you know the basic properties of a *multivariate Gaussian* probability distribution. You can plot a 2D Gaussian probability distribution given the *mean vector* $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$.
- you can *sample* data points from a given multivariate gaussian distribution.
- you can explain the *naïve Bayes classifier* to your classmates and to your teacher.

1. Supervised Bayesian Learining [M,II]

The table below contains the result of a market survey for a promotion for different items such as a magazine, a watch and a life insurance and a credit card insurance. 10 people were interviewed and asked whether they would buy such items.

Use this count table for supervised Bayesian learning. The output attribute is *sex* with possible values `male` and `female`. Consider an individual who has said *no* to the life insurance promotion, *yes* to the magazine promotion, *yes* to the watch promotion and *yes* to the credit card insurance. Use the values in the table together with the Naive Bayes classifier to determine which of a,b,c or d represents the probability that this individual is male. $p(E)$ is the marginal distribution.

	Magazine		Watch		Life Insurance		Credit Card	
	`male`	`female`	`male`	`female`	`male`	`female`	`male`	`female`
yes	4	3	2	2	2	3	2	1
no	2	1	4	2	4	1	4	3

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) $p(\texttt{sex} = \texttt{male}\|...) = \frac{4}{6} \cdot \frac{2}{6} \cdot \frac{2}{6} \cdot \frac{2}{6} \cdot \frac{6}{10} \cdot \frac{1}{p(E)}$	○	○
b) $p(\texttt{sex} = \texttt{male}\|...) = \frac{4}{6} \cdot \frac{2}{6} \cdot \frac{3}{4} \cdot \frac{2}{6} \cdot \frac{3}{4} \cdot \frac{1}{p(E)}$	○	○
c) $p(\texttt{sex} = \texttt{male}\|...) = \frac{4}{6} \cdot \frac{4}{6} \cdot \frac{2}{6} \cdot \frac{2}{6} \cdot \frac{6}{10} \cdot \frac{1}{p(E)}$	○	○
d) $p(\texttt{sex} = \texttt{male}\|...) = \frac{2}{6} \cdot \frac{4}{6} \cdot \frac{4}{6} \cdot \frac{2}{6} \cdot \frac{4}{10} \cdot \frac{1}{p(E)}$	○	○

2. Hamburger and Bayes Rule [A,II]

Consider the following fictitious scientic information: Doctors find that people with Kreuzfeld-Jacob disease (KJ) almost invariably ate hamburgers (Hamburger Eater, HE) , thus $p(\mathsf{HE}|\mathsf{KJ}) = 0.9$. The probability of an individual having KJ is currently rather low, about one in 100′000.

a) Assuming eating lots of hamburgers is rather widespread, say $p(\mathsf{HE}) = 0.5$, what is the probability that a hamburger eater will have Kreuzfeld-Jacob disease? Determine the prior, the likelihood function and the posterior probability.

b) If the fraction of people eating hamburgers was rather small, $p(\mathsf{HE}) = 0,001$, what is the probability that a regular hamburger eater will have Kreuzfeld-Jacob disease?

3. Naïve Bayes Classifier [A,II]

In order to reduce my email load, I decide to implement a machine learning algorithm to decide whether or not I should read an email, or simply file it away instead. To train my model, I obtain the following data set of binary-valued *features* about each email, including whether I know the author or not, whether the email is long or short, and whether it has any of several key words, along with my final decision about whether to read it ($y = +1$ for 'read', $y = -1$

for 'discard').

x_1	x_2	x_3	x_4	x_5	y
know `author`?	is `long`?	has `research`	has `grade`	has `lottery`	$\Longrightarrow$ read?
0	0	1	1	0	-1
1	1	0	1	0	-1
0	1	1	1	1	-1
1	1	1	1	0	-1
0	1	0	0	0	-1
1	0	1	1	1	+1
0	0	1	0	0	+1
1	0	0	0	0	+1
1	0	1	1	0	+1
1	1	1	1	1	-1

a) Compute all the probabilities necessary for a naïve Bayes classifier, i.e. the class probability $p(y)$ and all the individual feature probabilities $p(x_i|y)$, for each class y and feature x_i.

b) Which class would be predicted for $x = \{00000\}$?
What about for $x = \{11010\}$?

c) Compute the *posterior probability* that $y = +1$ given the observation $x = \{00000\}$. Also compute the posterior probability that $y = +1$ given the observation $x = \{11010\}$.

d) Why should we probably not use a *'joint' Bayes classifier* (using the joint probability of the features x, as opposed to the conditional independencies assumed by naïve Bayes) for these data?

e) Suppose that before we make our predictions, we lose access to my address book, so that we cannot tell whether the email author is known. Do we need to re-train the model to classify based solely on the other four features? If so, how? *Hint:* How do the parameters of a naïve Bayes model over only features $x_2, \ldots, x_5$ differ?

4. Passenger Scanner [A, II]

A secret government agency has developed a scanner which determines whether a person is a terrorist. The scanner is fairly reliable: 95% of all scanned terrorists are identied as terrorists, and 95% of all upstanding citizens are identied as such (i.e. as non-terrorists). An informant tells the agency that exactly one passenger of 100 aboard an aeroplane in which you are seated is a terrorist. The police haul off the plane the first person for which the scanner tests positive. What is the probability that this person is a terrorist?

5. Bivariate Gaussian Distribution [A,II]

The probability density function (pdf) of a multivariate normal with $\mathbf{x} = \begin{pmatrix} x_a \\ x_b \end{pmatrix}$ and $\boldsymbol{\mu} = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix}$ is given by:

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}\sqrt{\det(\boldsymbol{\Sigma})}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

Consider a bivariate normal distribution $p(\mathbf{x}) = p(x_a, x_b)$ with $\mu_a = 0$, $\mu_b = 2$, $\Sigma_{aa} = 2$, $\Sigma_{bb} = 1$ and $\Sigma_{ab} = \Sigma_{ba} = \frac{\sqrt{2}}{2}$.

a) Calculate the *precision matrix* $\boldsymbol{\Lambda}$, the *inverse* $\boldsymbol{\Sigma}^{-1}$ of the covariance matrix $\boldsymbol{\Sigma}$.

b) Write out the squared generalized distance expression, the *Mahalanobis distance*

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

as a function of x_a and x_b.

c) Write out the *bivariate normal density* $p(\mathbf{x}) = p(x_a, x_b)$ (joint probability density).

d) Calculate the *eigenvalues* $\lambda_{1,2}$ and the *eigenvectors* $\mathbf{u}_{1,2}$ of the covariance matrix $\boldsymbol{\Sigma}$ using Python (`numpy.linalg.eig`).

e) Plot the joint probability distribution $p(x_a, x_b)$ using Python (`scipy.stats.multivariate_normal`) and sample $N = 10'000$ points from this distribution.

f) Calculate the *conditional probability* $p(x_a|x_b)$.

6. Weather in London [A,II]

The weather in London can be summarised as: if it rains one day there's a 70% chance it will rain the following day; if it's sunny one day there's a 40% chance it will be sunny the following day.

$$p(\mathsf{today} = rain \mid \mathsf{yesterday} = rain) = 70\%$$

$$p(\mathsf{today} = sun \mid \mathsf{yesterday} = sun) = 40\%$$

From these likelihoods, we can *infer* the following:

$$p(\mathsf{today} = sun \mid \mathsf{yesterday} = rain) = 30\%$$

$$p(\mathsf{today} = rain \mid \mathsf{yesterday} = sun) = 60\%$$

a) Assuming that the prior probability it rained yesterday is 0.5, what is the probability that it was raining yesterday given that it's sunny today?

b) If the weather follows the same pattern as above, day after day, what is the probability that it will rain on any day (based on an effectively infinte number of days of observing the weather)?

c) Use the result from b) above as a new prior probability of rain yesterday and recompute the proba- bility that it was raining yesterday given that it's sunny today.

7. Inspector Clouseau [A,II]

Inspector Clouseau arrives at the scene of a crime. The victim lies dead in the room alongside the *possible* murder weapon, a knife. The Butler (B) and Maid (M) are the inspector's main suspects and the inspector has a prior belief of 0.6 that the Butler is the murderer, and a prior belief of 0.2 that the Maid is the murderer. These beliefs are independent in the sense that $p(B, M) = p(B)p(M)$. (It is possible that both the Butler and the Maid murdered the victim or neither). The inspector's prior criminal knowledge can be formulated mathematically as follows:

$$\mathrm{dom}(B) = \mathrm{dom}(M) = \{\mathsf{murderer}, \mathsf{not\ murderer}\}$$

$$\mathrm{dom}(K) = \{\mathsf{knife\ used}, \mathsf{knife\ not\ used}\}$$

$$p(B = \mathsf{murderer}) = 0.6, \qquad p(M = \mathsf{murderer}) = 0.2$$

$$\begin{aligned} p(\mathsf{knife\ used}|B = \mathsf{not\ murderer}, &\quad M = \mathsf{not\ murderer}) &&= 0.3 \\ p(\mathsf{knife\ used}|B = \mathsf{not\ murderer}, &\quad M = \mathsf{murderer}) &&= 0.2 \\ p(\mathsf{knife\ used}|B = \mathsf{murderer}, &\quad M = \mathsf{not\ murderer}) &&= 0.6 \\ p(\mathsf{knife\ used}|B = \mathsf{murderer}, &\quad M = \mathsf{murderer}) &&= 0.1 \end{aligned}$$

In addition $p(K, B, M) = p(K|B, M) \cdot p(B) \cdot p(M)$.

a) Assuming that the knife is the murder weapon, what is the probability that the Butler is the murderer, i.e. what is $p(B|K)$?
Hint: Remember that it might be that neither is the murderer. Use Bayes rule and marginalize (sum) over the possible states of B and M with $\mathrm{dom}(B) = \mathrm{dom}(M) = \{\mathsf{murderer}, \mathsf{not\ murderer}\}$).

8. Factorization of a multivariate probability distribution [A,II]

By using the definition of conditional probability, show that any multivariate joint distribution of N random variables has the following trivial factorization:

$$p(x_1, x_2, \ldots, x_N) = p(x_1 \mid x_2, \ldots, x_N) \cdot p(x_2 \mid x_3, \ldots, x_N) \cdot \cdots \cdot p(x_N) \tag{2}$$

9. Conditional distribution of a bivariate Gaussian distribution [A,II]

The bivariate normal distribution is given by:

$$\mathcal{N}(\mathbf{x} \mid \mu, \mathbf{\Sigma}) = \frac{1}{\sqrt{(2\pi)^N \cdot |\mathbf{\Sigma}|}} \cdot \exp\left[-\frac{1}{2}(\mathbf{x} - \mu)^{\mathbf{T}}\mathbf{\Sigma}^{-\mathbf{1}}(\mathbf{x} - \mu)\right] \tag{3}$$

where:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{4}$$

$$\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \tag{5}$$

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix} \tag{6}$$

The parameter ρ is called *correlation coefficient*. By using the definition of conditional probability, show that the *conditional* distribution $p(x_1 \mid x_2)$ can be written as a *normal* distribution of the form $N(x_1 \mid \tilde{\mu}, \tilde{\sigma})$ where

$$\tilde{\mu} = \mu_1 + \rho\frac{\sigma_1}{\sigma_2} \cdot (x_2 - \mu_2) \tag{7}$$

$$\tilde{\sigma} = (1 - \rho^2) \cdot \sigma_1^2 \tag{8}$$

Hint: Only look first at the quadratic form in the exponential of the joint probability distribution $p(x_1, x_2)$ and make use of the fact that x_2 is constant for the conditional probability distribution $p(x_1 \mid x_2)$. Sort out all terms with x_1.

OST
Ostschweizer
Fachhochschule

Lab 9

FTP MachLe MSE
FS 2022

Gaussian Processes

Machine Learning
WÜRC

Essentially, all models are wrong, but some are useful.
George E.T. Box

After this unit, ...

Lernziele/Kompetenzen

- you know the principle of maximum likelihood (`ML`) and and maximum a posteriori probability (`MAP`) and you know their difference.
- you know (K1), that both the *conditionals* $p(x|y)$ and the *marginals* $p(x)$ of a joint Gaussian $p(x, y)$ are again Gaussian.
- you know (K1) that a *Gaussian process* $\mathcal{GP}(\mu, k)$ is a generalization of a multivariate Gaussian distribution to infinitely many variables. A Gaussian process is a *prior* over *functions* $p(f)$ which can be used for Bayesian regression. Sampling from a Gaussian process means sampling *functions* (instead of samples of a random variable) out of a pool of functions characterized by a mean function μ and a covariance function $k(x, x')$.
- you know (K1), that every model relies on (explicit or implicit) *assumptions*. We discriminate *knowledge*, *assumptions* and *simplifying assumptions*. In Bayesian reasoning, assumptions are formulated as *prior distribution* $p(\theta)$ over the parameters θ of a model. Using Bayes rule, one can calculate the posterior parameter distribution $p(\theta|x, y)$ given the data (x, y) and the model assumptions.

$$\text{posterior} = p(\theta|x, y) = \frac{p(y|x, \theta) \cdot p(\theta)}{\int_\theta p(y|x, \theta) \cdot p(\theta) d\theta} = \frac{\text{likelihood} \cdot \text{prior}}{\text{marginal}} \tag{1}$$

- your are able to formulate *probabilitstic models* that use *priors* to express knowlege (or beliefs) about aspects of the model. You can formulate a *probabilistic model* for a process $f(x, \theta)$ with additive Gaussian noise ε. You can derive the *likelihood function* $p(y|x, \varepsilon, \theta)$ for this model given the parameters θ.
- you are able (K3) to *sample functions* from a Gaussian Process $\mathcal{GP}(\mu, k)$ with given mean $\mu(x)$ and covariance function $k(x, x')$ using the `GaussianProcessRegressor` of the class `sklearn.gaussian_process`.

- you are able (K3) to *fit* n-dimensional data using a Gaussian Process, i.e. you are able to *infer* hyperparameters of the model from given data using the `GaussianProcessRegressor` of the class `sklearn.gaussian_process`.
- you are able (K3) to *make predictions* using the `GaussianProcessRegressor` of the class `sklearn.gaussian_process`.
- you know (K1) that the *predictive distribution* which is used for making predictions for unknown data (x^*, y^*) can be calculated by *marginalizing* (integrating or averaging) over the parameter distribution.

$$p(y^*|x^*, x, y) = \int_\theta p(y^*|x^*, x, y, \theta) \cdot p(\theta|x, y) d\theta \tag{2}$$

- you know (K1) the most important covariance functions (kernels) $k(x, x')$, namely the *constant* kernel, the *Gaussian* kernel, the *RBF*-kernel (radial basis function), the *Dot-Product* kernel and the *sine-exponential* kernel.
- you are able (K3) to apply *kernel operations* (namely sum and product) in order to construct a probabilistic model adapted to a given dataset.

---

1. Medical Inference (Bayes Theorem) [M,I]

Breast cancer facts:

- 1 % of scanned women have breast cancer
- 80 % of women with breast cancer get positive mammography scans
- 9.6 % of women without breast cancer also get positive mammography scans

Question: A woman gets a scan, and it is positive. what is the probability that she has breast cancer?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) less than 1 %	○	○
b) less than 10 %	○	○
c) around 80 %	○	○
d) around 90 %	○	○

2. Likelihood function, MAP and linear regression [L, II]

Assume a linear model $y = \theta_1 + \theta_2 \cdot x$ for observed data points $\{\mathbf{X}, \mathbf{Y}\} = \{x_i, y_i\}$ $(i = 1 \dots N)$ superposed by Gaussian noise ε with zero mean value and variance σ_n^2. The parameters $\theta = \{\theta_1, \theta_2\}$ of the model are the offset θ_1 and the slope θ_2 and

$$\begin{aligned} y &= f(x|\theta) + \varepsilon \\ y &= \theta_1 + \theta_2 \cdot x + \varepsilon \\ \varepsilon &\sim \mathcal{N}(0, \sigma_n^2) \end{aligned}$$

a) Write down the *likelihood function* $p(\mathbf{Y}|\mathbf{X}, \theta)$.

b) Show that the *log-likelihood* takes the form of the sum of the squared errors $\mathrm{SSE}(\theta)$ between the model and the data points.

$$\log\left[p\left(\mathbf{Y}|\mathbf{X}, \theta\right)\right] = -\frac{1}{2\sigma_n^2} \sum_{i=1}^{N} \left(y_i - f(x_i|\theta)\right)^2 \qquad -\frac{N}{2} \cdot \log\left(2\pi\sigma_n^2\right) \tag{3}$$

$$= -\frac{1}{2\sigma_n^2} \left\|\mathbf{Y} - f(\mathbf{X}|\theta)\right\|^2 \qquad -\frac{N}{2} \cdot \log\left(2\pi\sigma_n^2\right) \tag{4}$$

$$= -\frac{1}{2\sigma_n^2} \mathrm{SSE}(\theta) \qquad -\frac{N}{2} \cdot \log\left(2\pi\sigma_n^2\right) \tag{5}$$

c) Show that in the case of a *linear* model in the parameter θ, we can write the sum of squared error $\mathrm{SSE}(\theta)$ in matrix form:

$$\begin{aligned} \mathrm{SSE}(\theta) &= \left(\mathbf{Y} - f(\mathbf{X}|\theta)\right)^T \cdot \left(\mathbf{Y} - f(\mathbf{X}|\theta)\right) \\ &= \left(\mathbf{Y} - \Phi \cdot \theta\right)^T \cdot \left(\mathbf{Y} - \Phi \cdot \theta\right) \end{aligned}$$

where

$$\theta = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} \qquad \Phi = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{pmatrix} = \left(\mathbb{1}_N \, \mathbf{X}\right) \tag{6}$$

d) To calculate the model parameters $\hat{\theta}_{\mathrm{ML}}$ that maximize the likelihood function, we could set the derivative of the likelihood function $p(\mathbf{Y}|\mathbf{X}, \theta)$ with respect to θ to zero and solve for θ. But instead of calculating the derivative of the likelihood function, it is easier to calculate the derivative of the log-likelihood function. The logarithm is monotonous, so the postitions of the extrema are not changed. Since the loglikelihood is proporitonal to the negative sum of squared errors, $\mathrm{SSE}(\theta)$, we can also look for a *minimum* in $\mathrm{SSE}(\theta)$.

Show that minimizing $\mathrm{SSE}(\theta)$ leads - in case of a linear model - to the normal equations:

$$\hat{\theta}_{\mathrm{ML}} = \left[\Phi^T \Phi\right]^{-1} \Phi^T \mathbf{Y} \tag{7}$$

3. Prior samples and posterior distributions from differnt kernels of a $\mathcal{GP}$ [A,II]

Open the Jupyter notebook `plot_gpr_prior_posterior.jpynb` and execute the first cell. The goal of this exercise is that you get familiar with the Gaussian process class `GaussianProcessRegressor` and the implemented `kernels` from `sckit-learn`.

a) Analyze the code and try to understand what is done at each line. At which line is the `posterior distribution` calculated for the given evidence (data). What is the prior used for calculating the posterior?

b) Have a look at the samples drawn from the given Gaussian processes for each kernel: `RBF, Matern, RationalQuadratic, ExpSineSquared, DotProduct and ConstantKernel`. Give an example for data that could be described by these covariance functions.
Hint:

- Check the kernel cookbook from `http://www.cs.toronto.edu/~duvenaud/cookbook/index.html`.
- Interested readers may have a look at the paper of Zoubin Ghahramani at `http://www.cs.toronto.edu/~duvenaud/cookbook/index.html`,
- or chapter 4 of C.E Rasmussens book available at `http://www.gaussianprocess.org/gpml/chapters/RW4.pdf`.

c) How would you model a periodic process with noise?

d) How would you set up a model for a polynomial regression using a Gaussian process?

e) For which cases would you use a Matérn kernel?

4. Model fitting, prediction and noise estimation using a $\mathcal{GP}$ [A,II]

Open the Jupyter notebook `FitGPModel_NoiseEstimation.jpynb` and execute the first cell. We generate a synthetic dataset of 300 samples that represent a noisy sine wave. The goal of this problem is to find an appropriate model for the data, to estimate the hyperparameters of the model using a Gaussian process, to make predictions and to get an estimate of the noise level in the data.

a) Plot the data in a scatter plot using `matplotlib.pyplot`. Try to estimate the noise level and the periodicity of the data. This will be used as staring point for the optimization of the *hyperparameters*.

```
nSamples=300;
rng = np.random.RandomState(0)
X = rng.uniform(0, 5, nSamples)[:, np.newaxis]
y = 0.5 * np.sin(3 * X[:, 0]) + rng.normal(0, 0.3, X.shape[0])
```

b) Generate a `kernel` for the covariance function of the Gaussian process as the sum of a `RBF`-kernel and a `WhiteKernel`. Import the *kernels* from `sklearn.gaussian_process`. Select appropriate length scales, noise levels and the corresponding lower and upper bounds for the optimizer.

```
from sklearn.gaussian_process import kernels
kernel = 1.0 * kernels.RBF(length_scale=..., length_scale_bounds=(...,
                                                    ...)) \
+ kernels.WhiteKernel(noise_level=..., noise_level_bounds=(..., ...))
```

c) Generate a Gaussian process `gp` as instance of the class `GaussianProcessRegressor` using the above `kernel` and fit the model to the data using the `.fit(X,y)` method.

```
gp = GaussianProcessRegressor(kernel=kernel,alpha=1e-5).fit(X, y)
```

d) Make *predictions* with the infered paramers of the model within the interval $X^* = [0, 10]$ using the `.predict` method of the `GaussianProcessRegressor`.
Set the option `return_cov=True` to get the covariance matrix.

```
X_ = np.linspace(0, 10, 100)
y_mean, y_cov = gp.predict(X_[:, np.newaxis], return_cov=True)
```

e) Plot the mean of the estimated model in the interval $X^* = [0, 10]$ using the `y_mean` output of the `.predict method`. Plot the confidence interval as $\pm$ one standard deviation from the mean value. The standard deviation can be obtained from `y_cov` by `stdv=np.sqrt(np.diag(y_cov))`. If you like, you can use `plt.fill_between` to paint the range within the standard deviation in gray. Explain how the model behaves in the interval where there were no data points available. Check and print the values of the *fitted hyperparameters*, especially the estimated noise level.

```
plt.fill_between(X_, y_mean - np.sqrt(np.diag(y_cov)),
y_mean + np.sqrt(np.diag(y_cov)),
alpha=0.5, color='k')
```

f) Repeat now the same process using the `ExpSineSquared kernel`. It has an additional parameter called `periodicity` that you have to specify including the lower and upper bounds.

```
kernel = 0.5 * kernels.ExpSineSquared(length_scale=..., periodicity=
                                      ...,
length_scale_bounds=(..., ...),
periodicity_bounds=(..., ...)) \
+ kernels.WhiteKernel(noise_level=..., noise_level_bounds=(..., ...))
```

Repeat the above steps from b) to e) using this kernel.

OST
Ostschweizer
Fachhochschule

Lab 10

FTP MachLe MSE
FS 2022

Reinforcement Learning

MSE MachLe
WÜRC

Lernziele/Kompetenzen

- You know the definition of a finite *Markov decision process* (MDP) and the definition of a *Markov Reward Process* (MRP).
- You understand the two basic algorithms for iteratively approximating the *dynamic programming* task for finite MDPs in the special cases of one-step, tabular, modelfree TD (time-difference) methods, namely *value iteration* and *Q-learning*.
- You know the difference between *on-policy* and *off-policy* learning.
- You can explain the difference between *value iteration* and *policy iteration*.
- You know the trade-off between *exploitation* and *exploration*.

---

1. General Questions: Reinforcement Learning [A, II]

a) Is the MDP framework adequate to usefully represent *all* goal-directed learning tasks? Can you think of any clear exceptions?

b) Imagine that you are designing a robot to run a maze. You decide to give it a reward of +1 for escaping from the maze and a reward of zero at all other times. The task seems to break down naturally into episodes - the successive runs through the maze - so you decide to treat it as an episodic task, where the goal is to maximize expected total reward. After running the learning agent for a while, you find that it is showing no improvement in escaping from the maze. What is going wrong? Have you effectively communicated to the agent what you want it to achieve?

c) Suppose $\gamma = 0.5$ and the following sequence of rewards is received $R_1 = -1$, $R_2 = 2$, $R_3 = 6$, $R_4 = 3$ and $R_5 = 3$, with $T = 5$. What are $G_0, G_1, \ldots, G_4$? *Hint*: Work backwards.

d) Suppose $\gamma = 0.9$ and the reward sequence is $R_1 = 2$ followed by an infinite sequence of 7s. What are G_1 and G_0?

e) Suppose the reinforcement learning player was greedy, that is, it always played the move that brought it to the position that it rated the best. Would it learn to play better, or worse, than a non-greedy player? What problems might occur?

2. Value Iteration on the Grid-World [A,II]
Gridworld is the most basic as well as classic problem in reinforcement learning and by implementing it on your own is the best way to understand the basis of reinforcement learning.

The rule is simple.

- Your agent/robot starts at the left-bottom corner (the 'start' sign) and ends at either +1 or -1 which is the corresponding reward.
- At each step, the agent has 4 possible actions including up, down, left and right, whereas the black block is a wall where your agent won't be able to penetrate through.
- If the agent hits the wall, it will remain at the same position.

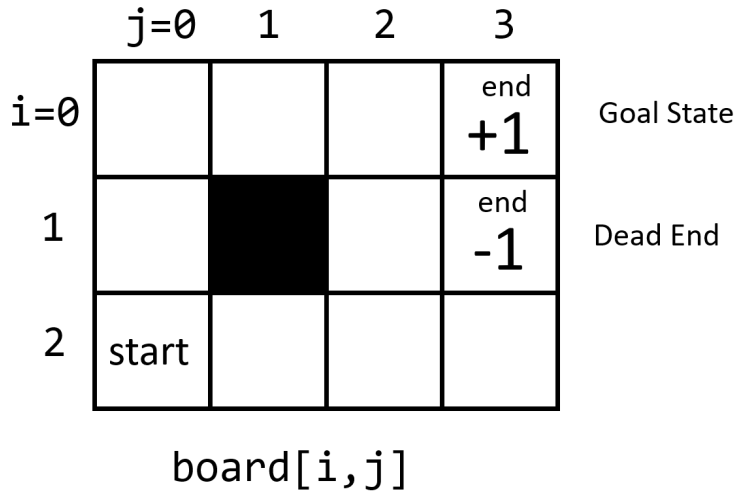


Abbildung 1: Simple Grid World

a) Open the Jupyter notebook `Lab10_A2_ValueIteration_GridWorld_TEMPLATE.jpynb`.
The python code defines two classes. The class `Environment` defines the board of the grid world and its possible legal actions. The agent can move in all four directions. If it hits a wall, it stays at the place.

The start, lose and win positions are defined as global variables:

```
BOARD_ROWS = 3
BOARD_COLS = 4
WIN_STATE = (0, 3)
LOSE_STATE = (1, 3)
START = (2, 0)
DETERMINISTIC = True
```

What our agent will finally learn is a policy π. A policy is a mapping from state s to action a, simply instructs what the agent should do at each state. In our case, instead of learning a mapping from state to action, we will leverage *value iteration* to learn a mapping of state to value (which is the estimated reward) and based on the estimation, at each state, our agent will choose the best action that gives the highest estimated reward.

$$\delta_t \leftarrow R + \gamma \cdot V(s_{t+1}) - V(s_t) \tag{1}$$
$$V(s_t) \leftarrow V(s_t) + \alpha \cdot \delta_t \tag{2}$$

This is the essence of *value iteration*. This formula almost applies to all reinforcement learning problems. Value iteration, just as its name, updates its value (estimated reward) at each iteration (end of game).

At first, our agent knows nothing about the grid world (environment), so it would simply initialise all rewards as 0. Then, it starts to explore the world by randomly walking around, surely it will endure lots of failure at the beginning, but that is totally fine. Once it reaches end of the game, either reward +1 or reward -1, the whole game reset and the reward *propagates in a backward fashion and eventually the estimated value of all states along the way will be updated based on the formula above.*

The $V(s_t)$ on the left is the updated value of that state, and the right one is the current non-updated value and α is the *learning rate*, γ is the discount factor. The formula is simply saying that the updated value of a state equals to the current value plus a *temporal difference*, which is what the agent learned from this iteration of game playing minus the previous estimate.

b) Run the Jupyter notebook `Lab10_A2_ValueIteration_GridWorld_TEMPLATE.jpynb`, get familiar with it and find the lines of code, where the update rule for the value function $V(s)$ (1) is implemented. Extend the formula such that a discounted reward with discount factor γ can be calculated. Why does the immediate reward R not appear in the formula?

c) Write a method `showValues()` for the class `Agent` that prints a table of the value function $V(s)$ after the iteration through the epochs.

d) Implement a method `SetParameters()` for the class `Agent` to change the learning rate α and the discount factor γ. Don't forget to initialize the table (dictionary) of the value function when updating one of these parameters.

e) Implement a method for the class `Agent` that calculates the ℓ_2-norm of the value function $\|V(s)\|_2$ as function of the epochs i and plot the norm of the value function $\|V(s)\|_2$ as function of the epochs i. We want to analyze the convergence behavior.

f) Plot the norm of the value function $\|V(s)\|_2$ for 150 epochs for the following learning rates α, discount factors γ and exploration rates ε:

- $\{\alpha, \gamma, \varepsilon\} = \{0.2, 1.0, 0.2\}$
- $\{\alpha, \gamma, \varepsilon\} = \{0.2, 0.8, 0.2\}$
- $\{\alpha, \gamma, \varepsilon\} = \{0.02, 0.8, 0.2\}$

What do you observe? Play with the values.

g) If you have fun, you could extend `GridWorld` by

- increasing the board size
- adding rewards on all cells of the board
- adding more obstacles
- ...

3. Q-Learning on the Grid-World [A,II]

The result of grid world using value iteration was that we get a estimated value $V(s)$ for each state, but to have our policy $\pi(s, a)$, which is a mapping from state s to action a, we need to go one step further by choosing the action that could get into the maximum value of next state. However, in Q-function, state and action are paired in the first place, which means when one has the optimal Q-function, he has the optimal action of that state.

Besides the Q-function, we are going to add more fun to our game:

- The agent action is *non-deterministic*, i.e. there is a transition probability $P_{ss'}$ from one state s to another state s'. Non-deterministic means that the agent will not be able to go where it intends to go. When it takes an action, it will has a probability to crash in a different action.
- Reward decays with ratio γ.

One of the early breakthroughs in reinforcement learning was the development of an off-policy TD control algorithm known as Q-learning (Watkins, 1989, `https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.80.7501&rep=rep1&type=pdf`), defined by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot \Big[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\Big]$$

In this case, the learned action-value function, Q, directly approximates Q^*, the optimal action-value function, *independent of the policy being followed*. This dramatically simplifies the analysis of the algorithm and enabled early convergence proofs. The policy still has an effect in that it determines which state–action pairs are visited and updated. The policy can directly be derived from Q, e.g. using an ε-greedy approach.

a) Open the Jupyter notebook `Lab10_A3_QLearning_GridBoard_TEMPLATE.jpynb` an get familiar with the code.

b) Generate a table of the corresponding value function $V^*(s)$

c) Write a method `ShowPolicy` for the class `Agent` that displays a table of the learned policy with arrows pointing into the greedy directions on each field of the board or using the words 'up','down','left' and 'right'.

4. Tic Tac Toe (optional) [A,I]

In general game theory, methods (e.g. the min-max algorithm) always assume a perfect opponent who is so rational, that at each step it will maximise its reward and minimise our agents reward. In reinforcement learning, the algorithm does not even presume a model of the opponent, and the result could still show a surprising performance. By considering the opponent as part of the environment which the agent can interact with over iterations, the agent is able to plan ahead without any model of the agent or environment, and without the need to conduct a search of possible future actions or states. The advantage is that the method saves a complex mathematical deduction and the exploration of a large search space. Nevertheless, reinforcement algorithms are still able to achieve state-of-art skills by simply trial and learn.

a) Open the Jupyter notebook `Lab10_A4_TicTacToe.jpynb`.

b) Train 2 agents to play against each other and save their policy.

c) Load the policy and make the agent to play against human.

d) Many tic-tac-toe positions appear different but are really the same because of symmetries. How might we amend the reinforcement learning algorithm described above to take advantage of this? In what ways would this improve it? Now think again. Suppose the opponent did not take advantage of symmetries. In that case, should we? Is it true, then, that symmetrically equivalent positions should necessarily have the same value?

e) (optional) Amongst several striking ideas, Peter Abbeel prominently mentions *self play* as one important ingredient of future successful AI agents trained with reinforcement learning (RL) in his recent summary talk at EECS [1], where he places this topic in larger context of *learning to learn*. Make yourself acquainted with Prof. Abbeel's ideas from the abovementioned talk on *learning to learn* [3] with a specific focus on how the *self play* he proposes [1][2] works. Compare it with the work of David Silver et al. on the game of Go [4].

[1.] Peter Abbeel, *Learning to Learn Robotic Control*, EECS colloquium, Oct 11, 2017, available online: `https://www.youtube.com/watch?v=TERCdog1ddE&t=3575s` (Feb 19, 2018)

[2.] Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, Pieter Abbeel, *Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments*, Oct 10, 2017, available online: `https://arxiv.org/abs/1710.03641` (Feb 19, 2018)

[3.] Chelsea Finn, *Learning to learn*, BAIR blog, Jul 18, 2017, available online: `http://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/` (Feb 19, 2018)

[4.] David Silver, Julian Schrittwieser, Karen Simonyan, et al., *Mastering the game of Go without human knowledge*, Nature, 550(7676), 354, 2017, available online: `https://www.nature.com/articles/nature24270` (Sep 04, 2018)

OST
Ostschweizer
Fachhochschule

Lab 11

FTP MachLe MSE
FS 2022

Dimensionality Reduction

MSE MachLe
WÜRC

Lernziele/Kompetenzen

- You know the main motivations and applications and drawbacks of *dimensionality reduction* of a high dimensional dataset.
- You can explain by an example what is meant by the term *curse of dimensionality* and the consequences of this fact.
- You can apply *Principal Component Analysis* (**PCA**) to high dimensional datasets and explain how PCA works. You know different versions of PCA like incremental PCA, vanilla PCA, randomized PCA and kernel PCA. You can explain the differences between these methods and know to apply them accordingly using `scikit learn`.
- You know the *kernel trick*, it's advantages and when it can be applied to a given technique. You can list at least four different kernels $k(x, x')$ that are frequently used in Machine Learning.
- You know the four axioms that define a *metrics* and can name four different metrices that are commonly used in Machine Learning.
- You know the manifold hypothesis and different manifold techniques like **MDS** (*Multidimensional scaling*), **LLE** (*local linear embedding*), **t-SNE** (*t-distributed stochastic neighbor embedding*) and **Isomap** (*Isometric mapping*) and can apply them to high dimensional datasets in `scikit learn`.
- You can successfully apply dimensionality reduction techniques for the *visualization* of high dimensional datasets, for *noise reduction* in datasets and for the *elimination of meaningless features used for classification*.

1. Applications of dimensionality reduction techniques [A, I]

a) What are the main *motivations* for reducing a dataset's dimensionality?

b) What are the main *applications* of dimensionality reduction techniques?

c) What are the main *drawbacks* of dimensionality reduction techniques?

2. Curse of Dimensionality [A,I]
What is the *curse of dimensionality*? Can you explain this with an example?

3. General questions [A,II]

a) Once a dataset's dimensionality has been reduced, is it possible to reverse the operation? If so, how? If not, why?

b) Can PCA be used to reduce the dimensionality of a highly nonlinear dataset? Which methods can alternatively be used?

c) How can you evaluate the performance of a dimensionality reduction algorithm?

d) Does it make sense to chain two different dimensionality reduction algorithms?

e) In which cases would you use incremental PCA, randomized PCA or kernel PCA?

4. PCA and scaling importance [A,II]
Feature scaling through standardization (or Z-score normalization) can be an important preprocessing step for many machine learning algorithms. Many algorithms such as SVM, K-nearest neighbors, and logistic regression and PCA require features to be normalized. In PCA we are interested in the components that maximize the variance. If one component (e.g. human height) varies less than another (e.g. weight) because of their respective scales (meters vs. kilos), PCA might determine that the direction of maximal variance more closely corresponds with the weight axis, if those features are not scaled.

The dataset used is the `Wine Dataset` available at UCI `https://archive.ics.uci.edu/ml/datasets/wine`. This dataset has continuous features that are *heterogeneous in scale* due to differing properties that they measure. It is the outcome of a chemical analysis wines grown in the same region in Italy but derived from *three different cultivars*. The analysis determined the quantities of 13 constituents (i.e alcohol content, and malic acid) found in each of the three types of wines.

a) Open the Jupyter notebook `Lab11_A4_PCA_ScalingImportance_Wine.jpynb`. We import the following modules:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine
from sklearn.pipeline import make_pipeline
```

b) Split the data into a training dataset (70%) and a test dat set (30%) using `train_test_split` from `sklearn.model_selection`.

c) Create a *pipeline* called `unscaled_clf` with `make_pipeline` using the following two operations:

1. PCA using only two components: `PCA(n_components=2)`
2. Naive Bayes Classifier (`GaussianNB`) for the different wine types (target).

d) Fit the model using the `.fit` method. Estimate the cultivar (target) using the `.predict` method on the test data and print the accuracy score on the test data using `metrics.accuracy_score`.

e) Change the PCA to 4 components, fit the model using the `.fit` method. Estimate cultivar (target) using the `.predict` method on the test data and print the accuracy score on the test data using `metrics.accuracy_score`.

f) Create a new pipeline called `scaled_clf` using the following operations:

1. `StandardScaler()`
2. PCA using only two components: `PCA(n_components=2)`
3. Naive Bayes Classifier (`GaussianNB`) for the different wine types (target).

g) Fit the model using the `.fit` method. Estimate the wine type (target) using the `.predict` method on the test data and print the accuracy score on the test data using `metrics.accuracy_score`.

h) Plot the transformed data as a scatter plot in the new coordinate system of the two principal components, once using the scaler and once without the scaler. Use differnt symbols and markers for each target (cultivar).

```
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=FIG_SIZE)

for l, c, m in
  zip(range(0, 3), ('blue', 'red', 'green'), ('^', 's', 'o')):
  ax1.scatter(Xtrain_PCA_unscaled[y_train == l, 0],
        Xtrain_PCA_unscaled[y_train == l, 1],
  color=c, label='class %s' % l, alpha=0.5, marker=m)

for l, c, m in
  zip(range(0, 3), ('blue', 'red', 'green'), ('^', 's', 'o')):
  ax2.scatter(Xtrain_PCA_scaled[y_train == l, 0],
        Xtrain_PCA_scaled[y_train == l, 1],
  color=c, label='class %s' % l, alpha=0.5, marker=m)
```

Note: ***Naive Bayes methods*** *are a set of supervised learning algorithms based on applying Bayes' theorem with the* naive *assumption of independence between every pair of features. In spite of their apparently over-simplified assumptions,* naive Bayes classifiers *have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters. Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the* curse of dimensionality*. However, although naive Bayes is known as a decent* classifier*, it is known to be a bad* estimator*, so the probability outputs from* ***`predict_proba`*** *are not to be taken too seriously.*

5. Stochastic Neighbour Embedding [A,II]

a) Use t-SNE to reduce the MNIST dataset down to two dimensions and plot the result using Matplotlib. You can use a scatterplot using 10 different colors to represent each image's target class.

b) Write colored digits at the location of each instance, or even plot scaled-down versions of the digit images themselves (if you plot all digits, the visualization will be too cluttered, so you should either draw a random sample or plot an instance only if no other instance has already been plotted at a close distance). You should get a nice visualization with well-separated clusters of digits.

c) Try using other dimensionality reduction algorithms such as PCA, LLE, or MDS and compare the resulting visualizations.

6. Feature Engineering using PCA [A,II]

a) Load the MNIST dataset and split it into a training set and a test set (take the first $60'000$ instances for training, and the remaining $10'000$ for testing).

b) Train a *Random Forest classifier* on the dataset and time how long it takes, then evaluate the resulting model on the test set.

c) Next, use PCA to reduce the dataset's dimensionality, with an explained variance ratio of 95%. Train a new Random Forest classifier on the reduced dataset and see how long it takes. How much faster was the training on your machine?

d) Next evaluate the classifier on the test set: how does it compare to the previous classifier?

7. Kernels and the Kernel Trick [A,II]

a) Explain what a kernel is.

b) List four commonly used kernels $k(x, x')$ in Machine Learning.

c) Show, that the RBS-kernel is symmetric and positive semi-definite.

OST
Ostschweizer
Fachhochschule

Lab 12

FTP MachLe MSE
FS 2022

Unsupervised Learning: Clustering

Machine Learning
WÜRC

After this unit, ...

Lernziele/Kompetenzen

- you know the *three* clustering algorithms: *k-means*, *dbscan* and a*agglomerative clustering* (using average, complete or Ward linkage).
- you are able to explain the working principle of *k-means*, *dbscan* and *agglomerative clustering*, their advantages and disadvantages and to apply them to data using `scikit-tlearn` in Python.
- you are able to plot the `inertia` and to determine the elbow point of this curve to find an optimum number of clusters as *hyperparameter*.
- you know the way how to *evaluate* a cluster algorithm using *metrics*, namely using `ARI` (*adjusted rand index*), `NMI` (*normalized mutual information*), `SC` (*silhouette score*) and *inertia*.
- you are able to correctly *scale* the data before clustering is applied especially (MinMax, StandardScaler, RobustScaler) or to meaningfully `transform` the data (eg. using `PCA`, `t-SNE` or `NMF`) before a clustering algorithm is applied.
- you know what a *Gaussian Mixture Model* `GMM` is and how the *expectation maximization algorithm* (`EM` algorithm) works. You are able to interpret the `kMeans` algorithm as a form of an `EM` algorithm with an E-step and and M-Step.
- your are able to apply clustering on the faces dataset (agglomerative, k-means and dbscan) to detect and *group* similar faces.
- your are able to apply a *hierarchical cluster analysis* on a voting dataset.

1. Clustering Algorithms [M,I]

This clustering algorithm initially assumes that each data instance represents a single cluster.

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) agglomerative clustering	○	○
b) t-SNE	○	○
c) k-means clustering	○	○
d) expectation maximization	○	○

2. Elbow Curve and `sklearn.cluster.KMeans` [A,I]

Using the following code lines, you can generate two-dimensional data clusters that can be used for testing clustering algorithms. In this exercise, you will learn how to apply k-means and how to determine the optimum number of clusters using the elbow criterium of the inertia plot.

```
from sklearn.datasets import make_blobs

X, y = make_blobs(n_samples=250,
  n_features=8,
  centers=8,
  cluster_std=0.85,
  shuffle=True,
  random_state=0)
```

a) Generate a distribution of 8 clusters with 250 samples and plot them as a scatterplot. How many clusters do you recognize with your eye. Try to change the cluster standard deviation `cluster_std` until it will be hard for you to discriminate the 8 different clusters.

b) Import the method `KMeans` from `sklearn.cluster`. Instantiate a model `km` with 8 clusters (`n_clusters=8`). Set the maximum number of iterations to `max_iter=300` and `n_init=10`. Fit the model to the data and predict the cluster label using `km.fit_predict(X)`.
Hint: One way to deal with convergence problems is to choose larger values for ***`tol`****, which is a parameter that controls the tolerance with regard to the changes in the within-cluster sum-squared-error to declare convergence. Try a tolerance of 1e-04.*

c) Use the function `PlotClusters` to display the clustered data.

```
#used for cycling through all defined colors
from matplotlib import colors as mcolors
colors = dict(mcolors.BASE_COLORS, **mcolors.CSS4_COLORS)
ColorNames=list(colors.keys())
HSV=colors.values()

def PlotClusters(X,y, km):

  for ClusterNumber in range(km.n_clusters):
    plt.scatter(X[y_km == ClusterNumber, 0],
      X[y_km == ClusterNumber, 1],
      s=50, c=ColorNames[ClusterNumber+1],
      marker='s', edgecolor='black',
      label='cluster {0}'.format(ClusterNumber+1))
```

```
plt.scatter(km.cluster_centers_[:, 0],
km.cluster_centers_[:, 1],
s=250, marker='*',
c='red', edgecolor='black',
label='centroids')
plt.legend(scatterpoints=1)
plt.grid()
plt.tight_layout()
#plt.savefig('Clusters.png', dpi=300)
plt.show()
```

d) Vary the number of clusters `n_clusters=8` in your `KMeans` clustering algorithm from 4 to 8 and display each time the result using the function `PlotClusters`.

e) Vary in a `for` loop the number of clusters from `n_clusters=8` to `n_clusters=15` and cluster the data each time using the `km.fit_predict` method. Read out the inertia `km.inertia_` and store it in a list called `distortions` as function of the number of clusters using the `append` method. Display the inertia as function of the number of clusters and determine the optimum number of clusters from the elbow curve.

f) Without explicit defintion, a random seed is used to place the initial centroids, which can sometimes result in bad clusterings or slow convergence. Another strategy is to place the initial centroids far away from each other via the `k-means++` algorithm, which leads to better and more consistent results than the classic k-means. This can be selected in `sklearn.cluster.KMeans` by setting `init=k-means++`.

(D. Arthur and S. Vassilvitskii. k-means++: The Advantages of Careful Seeding. In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007). `http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf`

3. k-Means, Gaussian Mixture Models and the EM algorithm [A,II]

Open the Jupyter notebook `Lab12_A3_EM_KMeans_MixtureModels.jpynb` that was originally created by *Sebastian Raschka* (`https://sebastianraschka.com/books.html`) and that also contains code from Jake Vanderplas' Python Data Science Handbook (`https://github.com/jakevdp/PythonDataScienceHandbook`). Work through the code and answer the following questions.

a) What are the basic assumptions of the k-Means algorithm? How does it work? Study the implementation in cell [8] and play with the interacitve code from Jake Vanderplas.

b) What are the *limitations* of kMeans? What can be done to overcome these limitations?

c) What is *hard* and what is *soft* clustering?

d) Explain how the `expecation maximization algorithm` (`EM`) works and list a five data science applications where it can be used. How would you prove that the `EM` algorithm converges to the maximum likelihood estimate of the hypothesis made?

e) What is a *Gaussian mixture model* (`GMM`)? What are the advantages of soft clustering using a `GMM` compared to kMeans?

4. Image compression using `kMeans` [A,II]

Clustering can be used to reduce colors in an image. Similar colors will be assigned to the same cluster label or color palette. In In the following exercise, you will load an image as a $[w, h, 3]$ `numpy.array` of type `float64` , where w and h are the width and height in pixels respecively. The last dimension of the three dimensional array are three the RGB color channels. Using `kMeans`, we will reduce the color depth from 24 bits to 64 colors (6 bits) and to 16 colors (4 bits).

a) Start by reading in an image from the Python imaging library `PIL` (`https://en.wikipedia.org/wiki/Python_Imaging_Library`) in your Jupyter notebook.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.utils import shuffle
from PIL import Image

# First we read and flatten the image.
original_img = np.array(Image.open('tree.jpg'), dtype=np.float64) / 255
print(original_img.shape)
original_dimensions = tuple(original_img.shape)
width, height, depth = tuple(original_img.shape)
```

b) Flatten the image to a $[w \cdot h, 3]$-dimensional numpy.array and shuffle the pixels using `sklearn.utils.shuffle`.

c) Create an instance of the `kMeans` class called `estimator`. Use the fit method of `kMeans` to create sixty-four clusters `(n_clusters=64)` from a sample of one thousand randomly selected colors, e.g. the first 1000 colors of the shuffled pixels. The new color palette is given by the cluster centers that are accessible in `estimator.cluster_centers_`.

d) Assign the cluster labels to each pixel in the original image using the `.predict` method of your `kMeans` instance. Now, you know to which color in your reduced palette each pixel belongs to.

e) Loop over all pixels and assign the new color palette corresponding to the label of the pixel and create a new, reduced color picture. Plot the images using `plt.imshow`, compare the original image and the 64 color image. Try the same with 32 and 16 colors.

5. Detecting similar faces using `DBSCAN` [A, II]

The *labelled* faces dataset of sckit-learn contains gray scale images of 62 differnet famous personalites from politics. In this exercise, we assume that there are no target labels, i.e. the names of the persons are unknown. We want to find a method to cluster similar images. This can be done using a dimensionality reduction algorithm like `PCA` for feature generation and a subsequent clustering e.g. using `DBSCAN`.

a) Open the Jupyter notebook `DBSCAN_DetectSimilarFaces.jpynb` and have a look at the first few faces of the dataset. Not every person is represented equally frequent in this *unbalanced* dataset. For classification, we would have to take this into account. We extract the first 50 images of each person and put them into a flat array called `X_people`. The correspinding targets (y-values, names), are storeed in the `y_people` array.

b) Apply now a principal component analysis `X_pca=pca.fit_transform(X_people)` and extract the first 100 components of each image. reconstruct the first 10 entries of the dataset using the 100 components of the PCA transformed data by applying the `pca.inverse_transform` method and reshaping the image to the original size using `np.reshape`. What is the minimum number of components necessary such that you recognize the persons? Try it out.

c) Import `DBSCAN` class from `sklearn.cluster`, generate an instance called `dbscan` and apply it to the pca transformed data `X_pca` and extract the cluster labels using `labels = dbscan.fit_predict(X_pca)`. Use first the standard parameters for the method and check how many unique clusters the algorithm could find by analyzing the number of unique entries in the predicted cluster labels.

d) Change the parameter `eps` of the `dbscan` using `dbscan(min_samples=3, eps=5)`. Change the value of `eps` in the range from 5 to 10 in steps of 0.5 using a `for` loop and check for each value of `eps` how many clusters could be determined.

e) Select the value of *eps* where the numbers of clusters found is maximum and plot the members of the clusters found using the follwing python code.

```
dbscan = DBSCAN(min_samples=3, eps= ...)
labels = dbscan.fit_predict(X_pca)

for cluster in range(max(labels) + 1):
mask = labels == cluster
n_images =  np.sum(mask)
fig, axes = plt.subplots(1, n_images, figsize=(n_images * 1.5, 4),
subplot_kw={'xticks': (), 'yticks': ()})
for image, label, ax in zip(X_people[mask], y_people[mask], axes):

ax.imshow(image.reshape(image_shape), vmin=0, vmax=1)
ax.set_title(people.target_names[label].split()[-1])
```

OST
Ostschweizer
Fachhochschule

Checkup

FTP MachLe MSE
FS 2022

Exercise Collection

Machine Learning
WÜRC

1 Short questions (K1)

1. Short Questions [A,II]

Answer the following questions with a `true` or `false` and give a short explanation for each question.

a) A classifier trained on less training data is less likely to overfit?

b) Given m i.i.d. data points, the training error converges to the true error as $m \to \infty$.

c) The maximum likelihood model parameters α can be learned using linear regression for the model: $y_i = \alpha_1 x_1 x_2^3 + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ iid noise.

d) The maximum likelihood model parameters (α) can be learned using linear regression for the model: $y_i = x_1^{\alpha_1} \cdot e^{\alpha_2} + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ iid noise.

e) Why does the kernel trick allow us to solve SVMs with high dimensional feature spaces, without significantly increasing the running time?

f) You are a reviewer for the International Mega-Conference on Algorithms for Radical Learning of Outrageous Stuff, and you read papers with the following experimental setups.

My algorithm is better than yours. Look at the training error rates!

Would you accept or reject each paper? Provide a one sentence justification. (This conference has short reviews.)

2. Kernel Trick [A,II]

a) What is the *kernel trick* and how is it useful?

2 Multiple Choice Questions (K1)

3. Mean and Variance [M,I]

Suppose x is a random variable which is distributed uniformly between 0 and 2. What are the *mean* and *variance* of x?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) $\bar{x} = 1$ and $\mathrm{VAR}(x) = \frac{1}{3}$.	○	○
b) $\bar{x} = 0$ and $\mathrm{VAR}(x) = 1$.	○	○
c) $\bar{x} = 1$ and $\mathrm{VAR}(x) = \frac{\sqrt{3}}{3}$.	○	○
d) $\bar{x} = 1$ and $\mathrm{VAR}(x) = \frac{2}{3}$.	○	○

4. Gaussian Distribution [M,I]

Suppose X and Y are two independent Gaussian random variables. Which of the following variable is Gaussian random variable?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) $4X + 3Y$.	○	○
b) X^2.	○	○
c) $X \cdot Y$.	○	○
d) $$Z = \begin{cases} X & \text{with probability } \frac{1}{2} \\ Y & \text{with probability } \frac{1}{2} \end{cases}$$.	○	○

5. k-fold crossvalidation [M,I]

The numerical complexity of k-fold crossvalidation is ....

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) linear in k.	○	○
b) quadratic in k.	○	○
c) cubic in k.	○	○
d) exponential in k.	○	○

6. Unsupervised Learning [M,I]
Thinking about unsupervised learning, which ones of the following statements are true (multiple choice) ?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) Differently from partitional graph based algorithms and density estimation algorithms the K-means and the EM algorithm need the number of clusters as an input parameter.	○	○
b) With hierarchical clustering algorithms based on graph theory we obtain a partition of a dataset in K different classes.	○	○
c) The K-Means algorithm obtains a global optimal solution for the partition of a dataset by minimizing the square distance between examples and their nearest centroid.	○	○
d) The EM algorithm assumes that the model of the data comes from a mixture of K n- dimensional probability distributions.	○	○

7. k-NN algorithm, accuracy and computational cost [M,I]
Given a k-NN classifier which ones of the following statements are true (multiple choice) ?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) The more examples are used for classifying an example, the higher is the accuracy we obtain.	○	○
b) The more attributes we use to describe the examples the more diffcult it is to obtain high accuracy.	○	○
c) The most costly part of this method is to learn the model.	○	○
d) We can use k-NN for classification and regression.	○	○

8. Overfitting [M,I]
How do you ensure you're not overfitting with a model ?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) Keep the model simpler: reduce variance by taking into account fewer variables and parameters, thereby removing some of the noise in the training data.	○	○
b) Use cross-validation techniques such as k-fold cross-validation.	○	○
c) Use regularization techniques such as LASSO that penalize certain model parameters if they're likely to cause overfitting.	○	○
d) Generating an ensemble of learners and using a softmax or voting output.	○	○

9. Validation and Learning Curves [M,I]

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) A *validation* curve shows the training error as function of the training data size.	○	○
b) If the *learning* curve shows a large gap between training and test error, the learner most probably suffers from a high *bias* problem.	○	○
c) *Hyperparameter tuning* shoud be done using a *cross-validated grid search* within the inner loop.	○	○
d) In a Bayesian approach, the hyperparameters can be defined by a *prior distribution*. Using Bayes theorem to calculate the posterior, we get a point estimate of the hyperparameters.	○	○

10. Estimation Error [M,I]

Thinking about the estimation error with the training set for a learning method. Which of the following statements are true? (multiple answer)

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) The estimation error of the decision tree built for the training set without pruning on the this trainng data is zero.	○	○
b) The estimation error for the training set for the built 3-KNN classifier with this data is zero.	○	○
c) The estimation error for the training set in the decision tree constructed with this data after the pruning procedure is zero.	○	○
d) The estimation error for the training set for the built 1-KNN classifier with this is zero.	○	○

11. Dimensionality Reduction, Wrappers and Filters [M,I]

Thinking about dimensionality reduction and attribute selection, which ones of the following statements are true (multiple choice) ?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) *Wrapper* methods usually provide the best performing feature set for a particular type of model. However, they are computationally intensive since for each subset a new model needs to be trained.	○	○
b) *Filter* methods are independent to the type of predictive model. Common measures are distance metrics, correlation, mutual information, and consistency metrics.	○	○
c) *Wrappers* are feature selection methods that, given a classifer as a performance criteria, search in the space of subset of features (feature combinations) for the minimal one that obtains the higher accuracy.	○	○
d) *Filters* are unsupervised feature selection methods because they use evaluation criteria from the intrinsic connections between features to score a feature subset.	○	○

12. Bias-Variance [M,I]
Adding more basis functions in a linear model ...
(Pick the most probable option.)

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) ... decreases model bias.	○	○
b) ... decreases estimation bias.	○	○
c) ... decreases variance.	○	○
d) ... doesn't affect bias and variance.	○	○

13. Debugging machine learing algorithms [M,I]
Suppose you are using some classifier algorithm on a given training set. The training error is acceptable. However, it makes unacceptably large errors in its predictions on unseen data. What should be tried next?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) Get more training samples.	○	○
b) Try larger sets of features, e.g. by generating polynomial features.	○	○
c) Decrease the regularization parameter λ if regularization is used.	○	○
d) Create a meta learner, e.g. a voting or bagging classifier out of several different classifiers or build an ensemble of classifiers.	○	○

14. Regularization for regression [M,I]

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) LASSO regularization uses the L_2 norm of the vector of parameters to be estimated.	○	○
b) LASSO regularization can be used for feature selection, because it forces a spase output where some parameters are set to zero.	○	○
c) Regularization for for any learner can be done by adding a penalty to the cost (loss) function that penalizes too complex models.	○	○
d) The parameter etimates for ridge regression can be calculated directy using an analytical solution.	○	○

15. Regularization [M,I]
Which of the following statements about regularization and the regularization parameter λ are correct ?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) Using too large a value of λ can cause your hypothesis to underfit the data.	○	○
b) Using too large a value of λ can cause your hypothesis to overfit the data.	○	○
c) Using a very large value of λ cannot hurt the performance of your hypothesis.	○	○
d) None of the above.	○	○

16. Getting struck in local minima [M,I]
How can you prevent K-means algorithm from getting stuck in bad local optima ?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) Set the same seed value for each run.	○	○
b) Use multiple random initializations.	○	○
c) Taking bootstrap samples of the data and run it several times.	○	○
d) Using K-means++.	○	○

17. Text Features [M,I]
Which of the following techniques can be used for *normalization* in text mining ?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) Stemming.	○	○
b) Lemmatization.	○	○
c) Stop Word Removal.	○	○
d) Bag of Words	○	○

18. k-NN classifier [M,I]
Which of the following statements is true for k-NN classifiers ?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) The classification accuracy is better with larger values of k.	○	○
b) The decision boundary is smoother with smaller values of k.	○	○
c) The decision boundary is linear.	○	○
d) k-NN does not require an explicit training step.	○	○

19. Feature generation for audio signals [M,II]

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) MFCC (Mel Frequency Cepstral Coefficients) are generated using filters in the spectral domain with overlapping linearly spaced frequency bands.	○	○
b) The *spectal spread* is the second central moment of the frequency spectrum of a signal.	○	○
c) A time-frequency Fourier transform with overlapping timeframes of approximately 330 ms duration is a good choice for feature generation for speech recognition.	○	○
d) The Fourier or cosine transformed spectrum is called *Cepstrum*. The Cepstrum is calculated in order to decorrelate the features in the MFCC representation.	○	○

3 Exercises (K3)

20. Kernel Method [A, II]

Suppose we have six training points from two classes. Note that we have four points from class 1: $(0.2, 0.4)$, $(0.4, 0.8)$, $(0.4, 0.2)$, $(0.8, 0.4)$ and two points from class 2: $(0.4, 0.4)$, $(0.8, 0.8)$. Unfortunately, the points cannot be separated by a linear classifier. The kernel trick is to find a mapping of x to some feature vector $\phi(x)$ such that there is a function K called kernel which satisfies $K(x, x') = \phi(x)^T \phi(x')$. And we expect the points of $\phi(x)$ to be linearly separable in the feature space. Here, we consider the following normalized kernel:

$$K(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^T \mathbf{x}'}{\|\mathbf{x}\| \, \|\mathbf{x}'\|} \tag{1}$$

a) What is the feature vector $\phi(\mathbf{x})$ corresponding to this kernel? Draw $\phi(\mathbf{x})$ for each training point $\mathbf{x}$ in a figure, and specify from which point it is mapped.

b) You now see that the feature vectors are linearly separable in the feature space. The maximum-margin decision boundary in the feature space will be a line in $\mathbb{R}^2$, which can be written as $w_1 x + w_2 y + c = 0$. What are the values of the coefficients w_1 and w_2? (`Hint`: you don't need to compute them, you can determine them graphically.)

c) Circle the points corresponding to support vectors in your figure.

d) Draw the decision boundary in the original input space resulting from the normalized linear kernel in your figure. Briefly explain your answer.

21. Manual calculation of the M step for a GMM [A, II]

Consider clustering 1D data with a mixture of 2 Gaussians using the EM algorithm. You are given the three 1-D data points $x = [1, 10, 20]$. Suppose the output of the E-step is the following matrix:

$$\gamma_{ik} = \begin{bmatrix} 1 & 0 \\ \frac{2}{5} & \frac{3}{5} \\ 0 & 1 \end{bmatrix} \tag{2}$$

where entry γ_{ik} is the probability of obervation x_i belonging to cluster k. You just have to compute the M-step. You may state the equations for maximum likelihood estimates of these quantities (which you should know) without proof; you just have to apply the equations to this data set. You may leave your answer in fractional form. Show your work.

a) Write down the likelihood function you are trying to optimize.

b) After performing the M-step for the mixing weights π_1, π_2, what are the new values $\pi_1^{(t+1)}$ and $\pi_2^{(t+1)}$?

c) After performing the M-step for the means μ_1 and μ_2, what are the new values $\mu_1^{(t+1)}$ and $\mu_2^{(t+1)}$?

-

22. Box Distribution [A,II]

A *box*-distribution $\text{box}(L, H)$ of a scalar random variable x is a PDF with two parameters, L and H (L for low and H for high) in which:

$$p(x) = \begin{cases} 0 & \text{if } (x < L) \\ \frac{1}{H-L} & \text{if } (L \leq x \leq H) \\ 0 & \text{if } (x > H) \end{cases} \tag{3}$$

We will use the notation $X \sim \text{box}(L, H)$ to mean that X is a random variable drawn from a *box*-distribution with parameters L and H.

a) Sketch the PDF of $\text{box}(L, H)$ by hand.

b) If $X \sim \text{box}(L, H)$, what is $\mathbb{E}[X]$?

c) If $X \sim \text{box}(L, H)$, what is $\text{VAR}[X]$?

d) Suppose you have data $x_1, x_2, \ldots, x_N$ i.i.d $\sim \text{box}(L, H)$ and suppose L and H are unknown. What are their MLE values (maximum likelihood estimates)? Explain. (Note this is a case where a careful few sentences explaining your answer may be better than an attempt at a proof by classic differentiation of a log-likelihood.)

23. Naive Bayes [A,II]

You are stranded on a deserted island. Mushrooms of various types grow widely all over the island, but no other food is anywhere to be found. some of the mushrooms have been determined as poisonous and others are not (determined by your former companion's trial and error) You are the only remaining on the island. You have the following data to consider.

Sample	`IsHeavy`	`IsSmelly`	`IsDotted`	`IsLammelar`	`IsPoisonous`
A	0	0	0	0	0
B	0	0	1	0	0
C	1	1	0	1	0
D	1	0	0	1	1
E	0	1	1	0	1
F	0	0	1	1	1
G	0	0	0	1	1
H	1	1	0	0	1
U	1	1	1	1	?
V	0	1	0	1	?
W	1	1	0	0	?

You know whether or not mushrooms A though H are poisonous, but you don't know about U through W. For the first couple of questions, consider only mushrooms A through H.

a) What is the entropy of `IsPoisonous`?

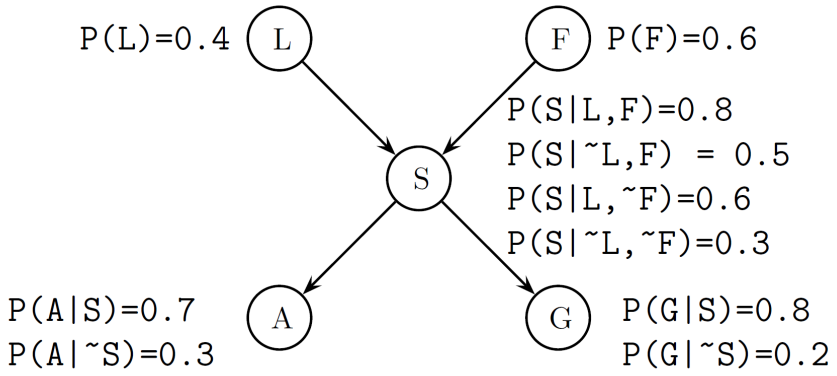
b) What prediction would a naive Bayes classifier assign for sample U?

c) What prediction would a naive Bayes classifier assign for sample W?

d) Which attribute should you choose as the root of a decision tree? Hint: You can figure this out by looking at the data without explicitely computing the information gain or the decrease in gini impurity of all four attributes.

24. Inference and Bayesian Networks [A, II]

A student of the Machine Learning class notices that people driving SUVs (S) consume large amounts of gas (G) and are involved in more accidents than the national average (A). He also noticed that there are two types of people that drive SUVs: people from Pennsylvania (L) and people with large families (F). After collecting some statistics, he arrives at the following *Bayesian network*.



a) What is $P(S)$?

b) What is $P(S \mid A)$?

-