Machine Learning

Summer Semester 2019, Homework 3

Prof. Dr. J. Peters, H. Abdulsamad, S. Stark, D.Koert

TECHNISCHE UNIVERSITÄT DARMSTADT

Total points: 67 + 10 bonus
Due date: Friday, 5 Juli 2019 (17:00)
You need to hand in the pdf in moodle **and** a printed version to the postbox from the IAS secretary office (S2 02 | E315)

Group ID : 146
Name, Surname, ID Number **Peter Nickl, 1941346** **Steffen Schäfer, 2635897**

Problem 3.1 Linear Regression [28 Points + 5 Bonus]

In this exercise, you will use the dataset `linRegData.txt`, containing 150 points in the format `<input variable, output variable>`. The input is generated by a sinusoid function, while the output is the joint trajectory of a compliant robotic arm. The first 20 data points are the training set and the remainder are the testing set.

a) **Polynomial Features [10 Points]**

Write the equation of the model and fit it with polynomial features. Using the Root Mean Square Error (RMSE) as a metric for the evaluation, select the complexity of the model (up to a 21st degree polynomial) by evaluating its performance on the testing data. Which is the best RMSE you achieve and what is the model complexity? Does it change if we evaluate our model on the training data? Comment your findings and plot the RMSE for each case (use two lines, one for evaluation on training data, one for evaluation on testing data). For the estimation of the optimal parameters use a ridge coefficient of $\lambda = 10^{-6}$.
Using what you think is the best learned model from the previous point, show in a single plot the ground truth (full dataset) and the model prediction over it. Attach snippets of your code showing how you generate polynomial features and how you fit the model.

b) **Gaussian Features [4 Points]**

Now use Gaussian features. Each feature is a Gaussian distribution were the means are distributed linearly in $x \in [0, 2]$ and the variance is set to $\sigma^2 = 0.02$. The features have to be normalized, i.e., they have to sum to one at every x. Using 10 features generate a plot with the activation of each feature over time (i.e., plot the matrix Φ). Attach a snippet of your code showing how to compute Gaussian features.

c) **Gaussian Features, Continued [4 Points]**

Repeat the process of fitting the model using the Gaussian features from the previous question. Compare the RMSE on the testing data using $15 \ldots 40$ basis functions and plot the RMSE. Which number of basis functions has the best performance and what is the best RMSE? Use a ridge coefficient of $\lambda = 10^{-6}$.

d) **Bayesian Linear Regression [10 Points]**

Using Bayesian linear regression, plot the mean and the standard deviation of the predictive distribution learned using the first $\{10, 12, 16, 20, 50, 150\}$ data points (one plot per case; plot it in the interval $x \in [0, 2]$). Discuss how the model uncertainty changes with the amount of data points and the problem of overfitting with Bayesian linear regression. Use the best performing polynomial features that you found in 3.1a, a ridge coefficient of $\lambda = 10^{-6}$, and assume Gaussian noise with $\sigma^2 = 0.0025$.

e) **Cross Validation [5 Bonus Points]**

So far, we have split our dataset in two sets: training data and testing data. Cross-validation is a more sophisticated approach for model selection. Discuss it and its variants, pointing out their pro and cons.

Cross validation is used for a number of different techniques in machine learning. In general it means comparing a Trainingset with some kind of Validation or Testset. Using this definition the easiest crossvalidation-method is the holdout method. The holdout method splits the data into three sets: Training-, Validation and Testset. The machine learning Algorithm is trained using the training set and evaluated on the validation set, for tweaking the parameters1. This approach prevents knowledge "leaking" from the Testset into the model. Therefore the metrics used on the testing set will measure the ability of the model to generalize.

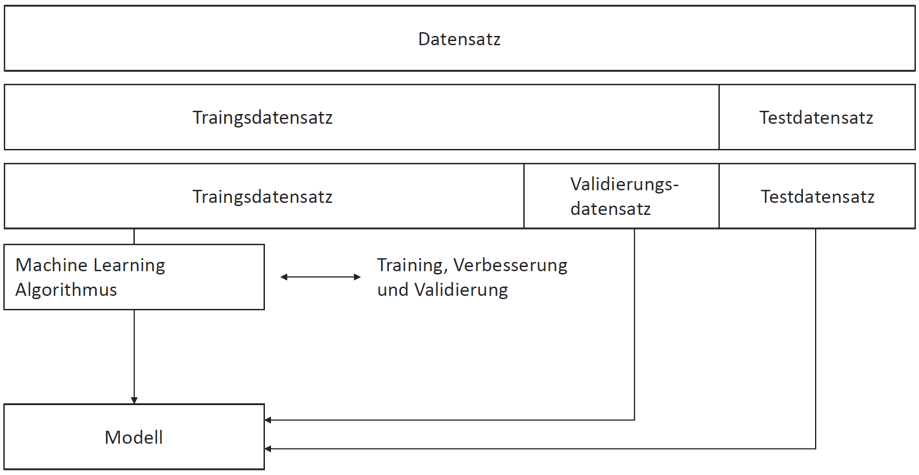

Figure 1: Holdout Cross-Validation schematic

While simple, this approach comes with several limitations. Mainly in reducing the number of training samples for learning the model drastically, but also for introducing a high amount of variance in the evaluation, since the results can depend on a particular random choice of train and validation sets.

To solve this problem k-fold crossvalidation (often just cross-validation, cv) is used. A test set still needs to be hold out for final validation, but we dont need a seperate validation set anymore. Instead we split the training set in k smaller sets. The model is now trained on k-1 sets and validated on the remaining part. This process is now repeated for each of the "k-folds" 1.

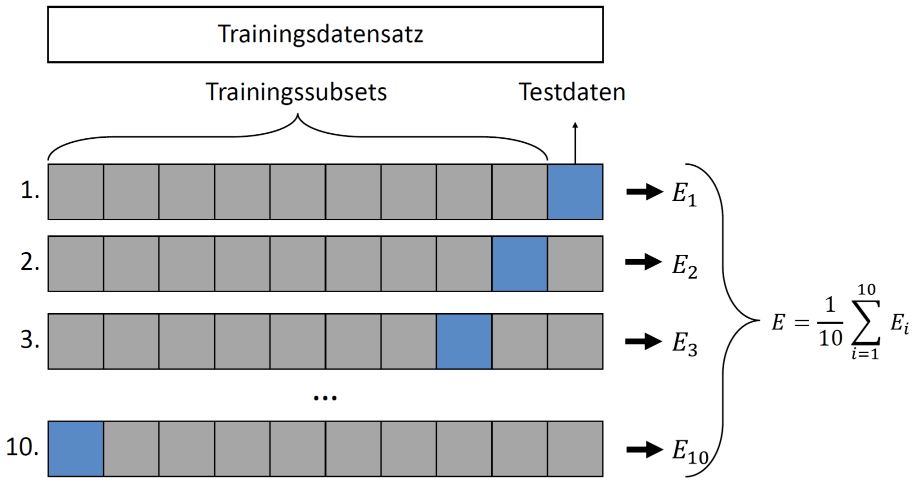

Figure 2: k-Fold cross validation splits the data into k-sets.

The measured perfromance is the average of all individuall runs. This approach can be quite expensive to compute, but uses less data and generally leads to less variance in the outcome.

To futher reduce the variance one can use the stratified k-fold cross-validation method. This method can be used, whenever all data is seen as equally important by the estimator, but may not be in reality. Instead of random shuffling the datasets, the stratified approach tries to create "stratified" folds, meaning that all folds have about the same percentage of samples of each target class, as the complete set. Only works on classification problems.

K-fold cross-validation reaches its limiations when one wants to Hyperparameterize their models. Since no validation set is used, any change of parameters done after seeing the test metrics will lead to overfitting. To bypass this, nested cross-validation is used. In nested cross-validation another loop is added to the process, to be able to hyperparameterize algorithms. Now the inner cross-validation loop is used for training and validating models, while the outer loop is used for changing hyperparameters (for example by using grid search) and to evaluate different model architectures. For example, when using nested cross-validation with neural networks, the inner loop is used to train the network by changing its weights and bias vectors and validate its generalizing performance. Meanwhile the outer loop will be used for changing its hyperparameters e.g. the number of hidden layers and Neurons. The result of the outer loop leads to the best overall model. The excact process is shown in the picture 3.

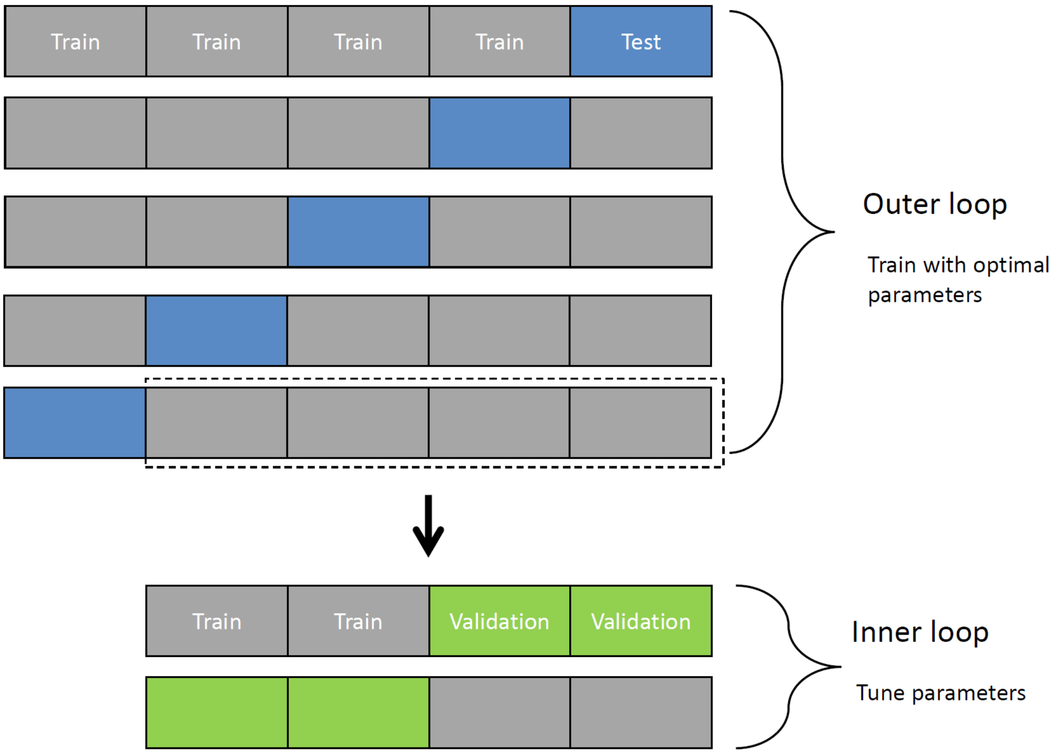

Figure 3: Schematic of nested cross validation. The inner loop is used to train and validate a simple mode, while the outer loop is used to compare different model architecures.

Problem 3.2 Linear Classification [16 Points]

In this exercise, you will use the dataset `ldaData.txt`, containing 137 feature points $\boldsymbol{x}$. The first 50 points belong to class C_1, the second 43 to class C_2, the last 44 to class C_3.

a) **Discriminative and Generative Models [4 Points]**

Explain the difference between discriminative and generative models and give an example for each case. Which model category is generally easier to learn and why?

Generative models *first use the underlying data to deduce properties of the underlying probability distribution in form of the class-doncitional densties $p(x|C_k)$ for each class C_k. Seperatly they infer the prior class probabilities $p(C_k)$. After that, they use Bayes'theorem in the form*

$$\frac{p(x|C_k)\,p(C_k)}{p(x)} \tag{2}$$

to fint the posteriori class probabilities $p(C_k|x)$. Now decision theory is used to determine class membership.

Since in theory we can use these distribution to create new data, this is called a gernative model. Also note that generative models also work when modeling the joint distribution $p(x, C_k)$ and normalizing.

Discriminative models *determine the posterior class probabilities $p(C_k|x)$ directly. After that decision theory is used to each new input to a class.*

Because generative models need the joint distribution $p(x, C_k)$ for calculating the posterior distribution, instead of deriving it directly, they're much harder to train. This is especially true if x is highly dimensional, since then we need a very big dataset to model class-doncitional densities correctly.

b) **Linear Discriminant Analysis [12 Points]**

Use Linear Discriminant Analysis to classify the points in the dataset, i.e., assume Gaussian distributions in each class with equal covariances and use the posterior distributions for assigning classes. Attach two plots with the data points using a different color for each class: one plot with the original dataset, one with the samples classified according to your LDA classifier. Attach a snippet of your code and discuss the results. How many samples are misclassified? (You are allowed to use built-in functions for computing the mean and the covariance.)

Problem 3.3 Principal Component Analysis [23 Points + 5 Bonus]

In this exercise, you will use the dataset `iris.txt`. It contains data from three kind of Iris flowers ('Setosa', 'Versicolour' and 'Virginica') with 4 attributes: sepal length, sepal width, petal length, and petal width. Each row contains a sample while the last attribute is the label (0 means that the sample comes from a 'Setosa' plant, 1 from a 'Versicolour' and 2 from 'Virginica'). (You are allowed to use built-in functions for computing the mean, the covariance, eigenvalues and eigenvectors.)

a) **Data Normalization [3 Points]**

Normalizing the data is a common practice in machine learning. Normalize the provided dataset such that it has zero mean and unit variance per dimension. Why is normalizing important? Attach a snippet of your code.

b) **Principal Component Analysis [8 Points]**

Apply PCA on your normalized dataset and generate a plot showing the proportion (percentage) of the cumulative variance explained. How many components do you need in order to explain at least 95% of the dataset variance? Attach a snippet of your code.

c) **Low Dimensional Space [6 Points]**

Using as many components as needed to explain 95% of the dataset variance, generate a scatter plot of the lower-dimensional projection of the data. Use different colors or symbols for data points from different classes. What do you observe? Attach a snippet of your code.

d) **Projection to the Original Space [6 Points]**

Reconstruct the original dataset by using different number of principal components. Using the normalized root mean square error (NRMSE) as a metric, fill the table below (error per input versus the amount of principal components used).

N. of components	x_1	x_2	x_3	x_4
1				
2				
3				
4				

Attach a snippet of your code. (Remember that in the first step you normalized the data.)

e) **Kernel PCA [5 Bonus Points]**

Throughout this class we have seen that PCA is an easy and efficient way to reduce the dimensionality of some data. However, it is able to detect only linear dependences among data points. A more sophisticated extension to PCA, *Kernel PCA*, is able to overcome this limitation. This question asks you to deepen this topic by conducting some research by yourself: explain what Kernel PCA is, how it works and what are its main limitations. Be as concise (but clear) as possible.