EXAMINATION QUESTIONS

Faculty: **Science and Technology**

Examination in: **DAT200** (*Course code*) **Applied Machine Learning** (*Course name*)

Time for exams: Monday, 27.05.2019 (*Day and date*) 14:00 – 17:30 (3.5 hours) (*As from – to and duration of examinations (hours)*)

Course responsible: Oliver Tomic and Ulf Indahl (*Name*)

Permissible aids:

A1: no calculator, no other aids

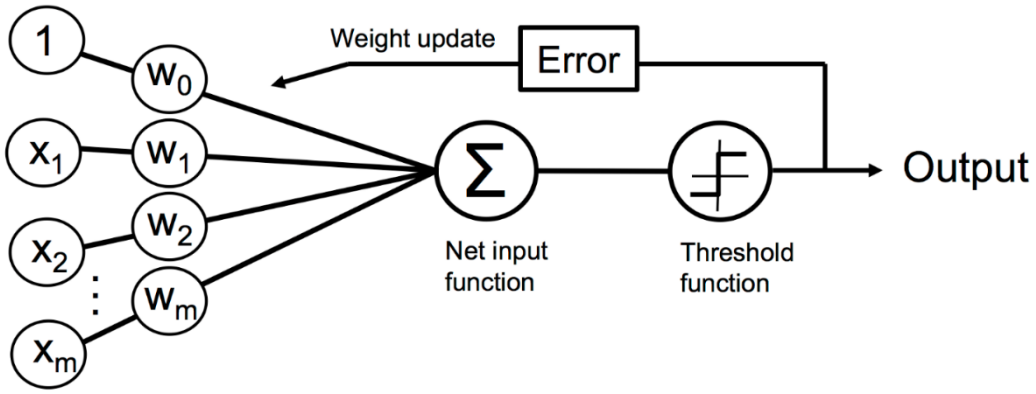
The exams papers includes: 9 (*Number of pages incl. attachment*)

If the examination consists of several parts, information must be given as to how much each part will count toward the grade

Course responsible: Oliver Tomic (9574 6167) and Ulf Indahl

External examiner: Tormod Næs

Exercise 1 (12 points in total)



Base your answers on the figure above.

a) (4 points)
Explain briefly how the Perceptron works when classifying a sample (no explanation on error computation or weight update needed here).
The input signals/features are weighted and summed together (including a bias), and a threshold is applied to decide which class the samples is predicted to belong to.

b) (4 points)
How are the ”net input function” and ”threshold function” formulated (explain and/or provide formulae)?
The “net input function” is the weighted sum of the inputs, i.e. the inner product between the features (and bias) and the weights. The “threshold function” sets a cut-off for the results from the “net input function”, usually 0 if a bias is included leading to positive or negative class.

c) (4 points)
How are the weights updated?
The weights are updated each time a sample has been predicted incorrectly during training. The weight update is computed by subtracting the predicted class from the actual class, which is then multiplied by the sample inputs and the learning rate.

$$\Delta w_j = \eta\left(y^{(i)} - \hat{y}^{(i)}\right)x_j^{(i)}$$

Exercise 2 (16 points in total)

a) (6 points)
What are unsupervised and supervised learning? When should you use them?
Unsupervised learning is the process of looking for systematic patterns in an unlabelled data set, e.g. patterns that maximize variation (like PCA). Supervised learning looks for patterns that are optimal for prediction of a response, i.e. labelled data. Unsupervised learning can be used for data compression or for explorative analyses. Supervised learning is used mostly for training models for prediction.

b) (6 points)
PCA, PCR and PLSR are related methods. In which way are they unsupervised/supervised?
PCA is unsupervised, searching for subspaces/components/factors that explain most of the variation in the data. PCR first applies unsupervised PCA, and then uses the first scores/components of this compression to do a supervised regression against a target vector/feature. PLSR maximizes the covariance between a set of features and a target vector/feature, i.e. fully supervised.

c) (4 points)
In machine learning, PCA is sometimes included in pipelines for regression or classification. What is the role of PCA in the pipeline? What does the hyperparameter for PCA control?
PCA compresses as set of features into a lower dimensional representation spanning most of the original variation and hopefully removing noise. The hyperparameter for PCA controls the number of components/features/dimensions to be extracted from the original data.

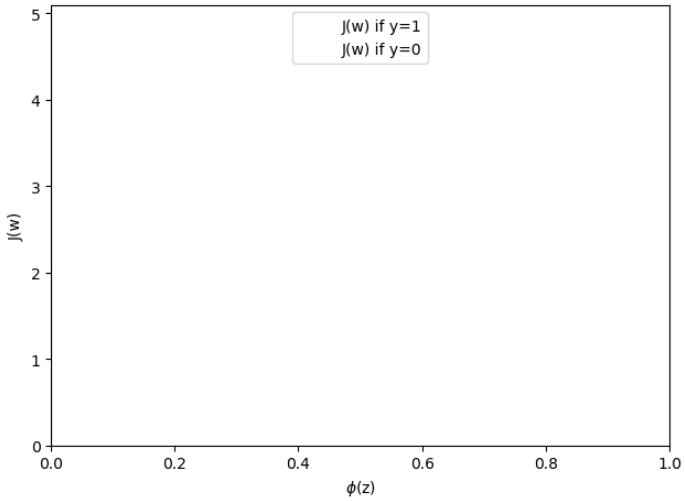
Exercise 3 (7 points in total)

Below the cost function of the logistic regression algorithm for a single sample is given.

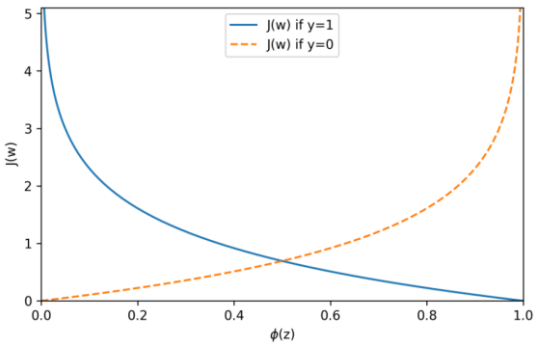
$$J(\boldsymbol{w}) = -y \log\big(\phi(z)\big) - (1-y)\log(1-\phi(z))$$

a) (4 points)
What does the logistic regression cost function above simplify to when a sample belongs to either class 0 or 1?
* For class 1 the cost function simplifies to $J(w) = -\log(\phi(z))$.
* For class 0 the cost function simplifies to $J(w) = -\log(1-\phi(z))$.

b) (3 points)
Sketch the logistic regression cost function in a figure as shown below. Indicate in the plot which part of the cost function represents class 0 and class 1 respectively. *Do not sketch it on this sheet, but on the sheets where you provide your answers!*



Solution



Exercise 4 (10 points in total)

a) (5 points)
Regularization/penalization is often used in machine learning models. Why would you regularize a model? Explain briefly.
Regularization is applied to reduce overfitting by penalizing the size of the weight in a model. Without regularization, many models tend to produce unstable and large weights that more or less cancel out and become too adapted to the training data. Regularization will reduce this behaviour and give more robust, generalising models.

b) (5 points)
Which are the two most common types of regularization in machine learning and how/what do they shrink?
L1 regularization (absolute shrinkage, LASSO) and L2 regularization (norm shrinkage, Ridge) are the two most commonly used. They add a penalty term to the cost function consisting of the sum of the absolute weights or the sum of squared weights, i.e. forcing shrinkage of the weights, sometimes leading to variable selection in the L1 case.

Exercise 5 (13 points in total)

a) (3 points)
Explain the concept of majority voting in classification.
Majority voting means that the final choice of predicted class will be decided by an ensemble of predictions, e.g. from different models, where the most common prediction for a given sample is selected.

b) (5 points)
How is majority voting applied in bagging? Why may bagging be more accurate than using a single classification model?
In bagging many subsets of at data set are used to train different models, and the final prediction is found by a majority vote over the individual predictions. Sample subsets may learn local patterns better and reduce the influence of outliers.

c) (5 points)
How is majority voting applied in K-nearest neighbours (KNN) classification? Why is KNN usually too resource demanding when the training data set is very large?
Majority voting is the foundation of KNN, where the predicted class is the one which is most common among the K neighbours. KNN needs to retain the full training data set for predictions of new samples instead of learning a simpler representation, e.g. a coefficient/weight vector of a decision boundary. Both storage of the original data and searching for neighbours becomes infeasible if there are too many samples.

Exercise 6 (7 points in total)

One-vs-Rest (OvR) (also called One-vs-All (OvA)) is an often used strategy when working with classifiers.

a) (5 points)
Explain briefly the concept of OvR / OvA.
* OvR or OvA is a method that extends binary classifiers (such as perceptron, adaline or logistic regression) to handle multiclass problems.
* Using OvA, we train one classifier per class, where the particular class is treated as the positive class and the samples from all other classes are considered negative classes

b) (2 points)
Explain briefly how new samples are classified with OvR / OvA applied to the Perceptron algorithm.
* For the new sample, compute net input value z for each of the *n* trained classifiers (one classifier per class)
* Choose the class that is associated with the largest net input value z among the *n* trained classifiers

Exercise 7 (15 points in total)

Feature importance is an important tool for determination of which variables are important to a model. Assume you have a regression model and you would like to know the importance of each feature of the model. The feature importance methods available to you are dropout/drop column feature importance and feature importance permutation.

a) (6 points)
Explain how feature importance is computed using dropout/drop column feature importance.
Using the dropout method, the importance of a feature is computed by: (I) removing first feature f1 from the training and test set; (II) train a new model using the same hyperparameters (as on the full model) on the training set; (III) make predictions from the test set and compute the performance; (IV) compare that drop-out performance against the baseline performance (original test set where all features were used). The importance is computed in the following way: importance of feature f1 = performance (baseline) – performance (drop-out f1). The higher the resulting value, the more important this feature is in the model and vice versa. This procedure needs to be repeated for each feature in the model.

b) (5 points)
Explain how feature importance is computed using feature importance permutation.
Using the feature importance permutation method, the importance of a feature is computed by: (I) permuting the rows of feature f1 in the test set; (II) make predictions from that modified test set; (III) compare that permutation performance against the baseline performance (with the original test set). The importance is computed in the following way: importance of feature f1 = performance (baseline) – performance (permutation f1). The higher the resulting value, the more important this feature is in the model and vice versa. This procedure needs to be repeated for each feature in the model.

c) (4 points)
Given a situation where you have only limited hardware resources (it would take a long time to re-train the model), which of the methods mentioned in sub-exercise a)/b) would you use? Explain why you would make that choice.
In this situation, the right choice would be feature importance permutation since re-training would not be necessary. Compared to a long training period, permutation of column data is inexpensive, as is prediction using the permuted data.

Exercise 8 (20 points in total)

You work as a data scientist at hospital A. Your department leader wants you to analyse data of 500 patients that were treated for myocardial infarction (heart attack) at hospital A. You are supposed to build a model (based on 15 clinical features) that can be used to predict whether a patient will be dead (class 1) or alive (class 0) after the treatment. Note that about 85% of the patients survive the treatment. Your department leader asks you to train a K-Nearest Neigbours model using a scikit-learn pipeline. Here is what you are supposed to implement in a script using pandas and scikit-learn:

1. Use the data provided to you in a comma-separated CSV-file named "infarction.csv". The first column contains integer class labels, the remaining 15 columns are clinical features of type float. The data has no header.
2. Use pandas to load the data.
3. Reduce the dimension of the data from 15 features to three features using PCA.
4. Train a KNN classifier with those three extracted features using grid search (5-fold cross validation). Search across the following KNN parameters:
    a. 1, 2, 3, 4 and 5 neighbours (`'kneighborsclassifier__n_neighbors'`)
    b. Values 1 and 2 for p (the parameter of the distance metric 'Minkowski', `'kneighborsclassifier__p'`)
    c. Make sure to use all processor cores on your computer
5. Use 20% of the data to test the model
6. After completed grid search, print the best score and the best parameters of the KNN

Use the list of arbitrary commands provided in **scikit-learn / pandas code** on the last page for support. Make sure that you insert your choices of parameters in those places highlighted with '*YOUR_INPUT*' or '*YOUR_ESTIMATOR*'. NOTE: you don't need to use all commands on this list to make your script work. Regular Python code is not provided.

Solution:

```
#================================================================
# Import modules
#================================================================
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import make_pipeline
from sklearn.decomposition import PCA

import pandas as pd


#================================================================
# Load data
#================================================================
df = pd.read_csv('infarction.csv', sep=',', header=None)
X, y = df.iloc[:, 1:].values, df.iloc[:, 0].values
```

```
#=================================================================
# Preprocess data
#=================================================================
X_train, X_test, y_train, y_test = \
    train_test_split(X, y,
                     test_size=0.20,
                     stratify=y,
                     random_state=1)


#=================================================================
# Generate pipeline and train model
#=================================================================
# Generate pipeline
pipe_knn = make_pipeline(StandardScaler(),
                         PCA(n_components=3),
                         KNeighborsClassifier(n_jobs=-1))

# Define range of values for parameters in grid search
n_range = [1, 2, 3, 4, 5]
p_range = [1, 2]

param_grid = [{'kneighborsclassifier__n_neighbors': n_range,
               'kneighborsclassifier__p': p_range}]

# Do grid search
gs = GridSearchCV(estimator=pipe_knn,
                  param_grid=param_grid,
                  scoring='f1',
                  cv=5)

# Print out best metric score and the best parameters
gs = gs.fit(X_train, y_train)
print(gs.best_score_)
print(gs.best_params_)

# Not part of solution, just additional info
test_score = gs.score(X_test, y_test)
```

Python / scikit-learn / pandas code

```
X_train, X_test, y_train, y_test = train_test_split(YOUR_INPUT, YOUR_INPUT, test_size=YOUR_INPUT,
      stratify=YOUR_INPUT, random_state=YOUR_INPUT)
from sklearn.ensemble import RandomForestClassifier
YOUR_ESTIMATOR.best_params_
from sklearn.metrics import accuracy_score
from sklearn.decomposition import PCA
from sklearn.metrics import precision_score
np.hstack((YOUR_INPUT, YOUR_INPUT))
'kneighborsclassifier__p'
PCA(n_components=YOUR_INPUT)
from sklearn.neighbors import KNeighborsClassifier
YOUR_ESTIMATOR.fit(YOUR_INPUT, YOUR_INPUT)
from sklearn import datasets
GridSearchCV(estimator=YOUR_INPUT, param_grid=YOUR_INPUT, scoring=YOUR_INPUT, cv=YOUR_INPUT)
from sklearn.model_selection import train_test_split
np.vstack((YOUR_INPUT, YOUR_INPUT))
from sklearn.svm import SVC
pandas.read_csv(YOUR_INPUT, sep= YOUR_INPUT, header=YOUR_INPUT)
from sklearn.metrics import accuracy_score
X, y = someName.iloc[YOUR_INPUT, YOUR_INPUT].values, someName.iloc[YOUR_INPUT, YOUR_INPUT].values
from sklearn.preprocessing import StandardScaler
YOUR_ESTIMATOR.best_score_
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import recall_score
from sklearn.pipeline import make_pipeline
YOUR_ESTIMATOR.transform(YOUR_INPUT)
from sklearn.metrics import f1_score
ax.axhline(y=YOUR_INPUT, linewidth=YOUR_INPUT, color=YOUR_INPUT, linestyle=YOUR_INPUT)
from sklearn.datasets import make_moons
accuracy_score(YOUR_INPUT, YOUR_INPUT)
'kneighborsclassifier__n_neighbors'
import pandas as pd
from sklearn.metrics import roc_auc_score
```