# EXAMINATION QUESTIONS

|  |  |  |
|---|---|---|
| Faculty: | **Science and Technology** |  |
| Examination in: | **DAT200**<br>*Course code* | **Applied Machine Learning**<br>*Course name* |
| Time for exams: | Monday, 28.05.2018<br>*Day and date* | 9:00 – 12:30 (3.5 hours)<br>*As from – to and duration of examinations (hours)* |
| Course responsible: | Kristian Hovde Liland and Oliver Tomic<br>*Name* |  |

**Permissible aids:**

**A1: no calculator, no other aids**

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

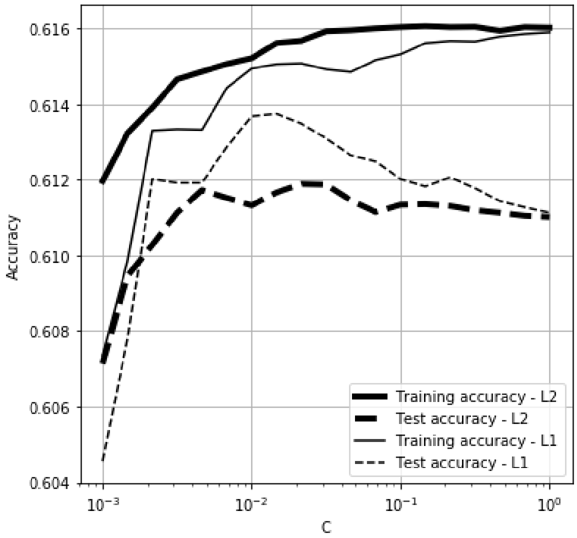
**If the examination consists of several parts, information must be given as to how much each part will count toward the grade**

Course responsible: Kristian Hovde Liland and Oliver Tomic

External examiner: Bjørn-Helge Mevik

# Exercise 1 (10 points in total)

When analysing a data set, two Logistic Regression models have been fitted. L2 and L1 norm regularization have been applied. Mean accuracy results for repeated training-test splits are reported in the following figure plotted against the inverse regularization parameter.



**a) (5 points)**
Which regularization would you choose for future predictions on new data? Approximately, which corresponding C-value would you choose?

**b) (5 points)**
What are the L2 and L1 regularizing? Which of these can lead to variable selection during regularization?

a) L1-regularization seems to give the best test accuracy, but the differences are really not very large. With a regularization parameter value C slightly above $10^{-2}$ the associated test accuracy is at the largest, and the difference to the training accuracy is at the smallest. Slightly stronger regularization (lower C-values) may provide more robust models, but too small ($< 10^{-3}$) C-values seems to be harmful to the level of accuracy. L2-regularization seems to overfit slightly more than the L1-regularization for C < 1.

b) Let $\boldsymbol{\beta} = (\beta_1, \beta_2, \dots, \beta_p)^t$ be the vector of regression coefficients in Logistic Regression.

- L2-regularization: Shrinking of the L2-norm of the regression coefficients $||\boldsymbol{\beta}||^2 = \beta_1^2 + \cdots + \beta_p^2$.
- L1-regularization: Shrinking of the L1-norm of the regression coefficients $||\boldsymbol{\beta}||^1 = |\beta_1| + \cdots + |\beta_p|$.

In the L1-case substantial shrinking may lead to sparse solutions (variable selection) as some of the regression coeffs may be set to 0.

# Exercise 2 (10 points in total)

A classification problem with three classes leads to the following one-versus-all confusion matrices.

| | | |
|---|---|---|
| True | 21 | 4 |
| | 2 | 13 |
| | Predicted | |

| | | |
|---|---|---|
| True | 29 | 6 |
| | 1 | 4 |
| | Predicted | |

| | | |
|---|---|---|
| True | 15 | 5 |
| | 6 | 14 |
| | Predicted | |

**a) (2 points)**
Sketch a 2 x 2 confusion matrix and fill it with TP, TN, FP and FN, assuming the first class is negative and the second class is positive.

**b) (4 points)**
Given the formulas for precision = TP/(TP+FP) and recall = TP/(FN+TP), how would you explain the interpretation of precision and recall based on the confusion matrix you sketched to someone not familiar with machine learning? (Only explain the type of performance that is measured.)

**c) (4 points)**
How would you compute precision and recall for the combined three-class classification? Show the elements of the calculations and explain your choice.

**a)** (Transposed or switched N and P gives partial correct when correcting the exam)

| | | |
|---|---|---|
| True | TN | FP |
| | FN | TP |
| | Predicted | |

**b)** The precision tells us how large proportion of the samples that are predicted to be positive are actually positive, i.e. how precise are the positive predictions.

The recall tells us how large proportion of the positive samples were predicted as positive, i.e. what is the rate of recall (by prediction) among the positive samples.

**c)** ‘micro’: sum over matrices, then compute metrics (all samples contribute equally), or ‘macro’: average over matrix-wise metrics (down-weight large classes, suitable for unbalanced data like the ones in this exercise.

Micro: PRE = (13+4+14) / (13+4+14+4+6+5) = 31/46 (= 0.674)
REC = (13+4+14) / (13+4+14+2+1+6) = 31/40 (= 0.775)

Macro: PRE = (13/(13+4) + 4/(4+5) + 14/(14+5)) / 3 (= 0.649)
REC = (13/(13+2) + 4/(4+1) + 14/(14+6)) / 3 (= 0.789)

No calculator was used on the exam.

## Exercise 3 (15 points in total)

The following algorithm is the AdaBoost in random order.

| | |
|---|---|
| A-2 | For $j$ in $m$ boosting rounds, do the following: |
| B-4 | Predict class labels: $\hat{y} = \text{predict}\left(C_j, \boldsymbol{X}\right)$. |
| C-1 | Set the weight vector **w** to uniform weights, where $\sum_i w_i = 1$. |
| D-6 | Compute coefficient: $\alpha_j = 0.5 \log \frac{1-\varepsilon}{\varepsilon}$. |
| E-9 | Compute the final prediction: $\hat{\boldsymbol{y}} = \left(\sum_{j=1}^{m} \left(\boldsymbol{\alpha}_j \times \text{predict}\left(C_j, \boldsymbol{X}\right)\right) > 0\right)$. |
| F-8 | Normalize weights to sum to 1: $\boldsymbol{w} := \boldsymbol{w} / \sum_i w_i$. |
| G-7 | Update weights: $\boldsymbol{w} := \boldsymbol{w} \times \exp\left(-\alpha_j \times \hat{\boldsymbol{y}} \times \boldsymbol{y}\right)$. |
| H-3 | Train a weighted weak learner: $C_j = \text{train}\left(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{w}\right)$. |
| I-5 | Compute weighted error rate: $\varepsilon = \boldsymbol{w} \cdot \left(\hat{\boldsymbol{y}} \neq \boldsymbol{y}\right)$. |

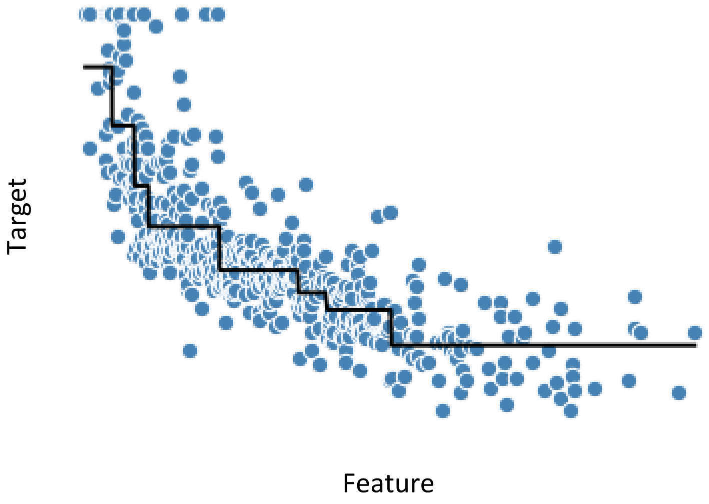
**a) (10 points)**
Reorder the lines correctly. (The correct sequence of line letters is enough).

**b) (5 points)**
Comment briefly on what each line does (short and concise, preferably one or two sentences per line).

a) C, A, H, B, I, D, G, F, E
b)
C: Initialize the weight without any prior knowledge.
A: Outer loop of *m* steps.
H: Train a classifier which is only slightly better than random guessing using the current weights (all data).
B: Classify the whole data set using the weak learner.
I: Let each sample's influence on the average error be adjusted by their weights.
D: $\alpha_j$ is the weight the current weak learner will be given in the final ensemble and controls the amount of change to the weights before the next iteration.
G: Down-weight correctly classified samples and up-weight wrongly classified samples, using the learner success weights from D.
F: Make sure the weights do not start to grow or shrink on average.
E: Combine all the weak learner predictions, weighting them by their learner success weights.

## Exercise 4 (12 points in total)

A decision tree has been used to fit the jagged curve going through the points in the following plot.



**a) (6 points)**
Describe briefly how the curve is made. Focus on the decision process of splitting the feature. What is optimised in the steps of this process?

**b) (6 points)**
Explain how Random Forests generalizes this decision tree. What makes Random Forests a more likely candidate for robust and general predictions in this type of problem?

(Please limit descriptions to less than a page in total Exercise 4.)

**a)** The feature is sequentially split into two parts (vertical jumps), breaking the line in to 2^#repeats horizontal lines, each placed at the mean feature value for its area.
Placement of the split in each iteration is adjusted to minimize the sum of squared error between the horizontal lines on the left and right side and their corresponding samples.
**b)** In Random Forests, many decision trees are made from bootstrapped object-feature sets, i.e. random samples of objects and random samples of features. Each tree is a decision tree, often having features split until all branches are unique objects. The final model is either a majority vote on a subset of well-performing trees (classification) or a weighted sum of all trees (regression).
Though the trees making up a Random Forest may be overfitted, taking the average over many differently overfitted trees often smooths over these effects and balances the model between high flexibility and robustness. One would expect a Random Forest prediction on the plotted data to have more steps, possibly with small local overfit, but in general following the curvature of the data better.

## Exercise 5 (8 points in total)

After a clustering algorithm has been applied to six data points, the following two clusters are found.

| | Feature 1 | Feature2 |
|---|---|---|
| Obj. 1 | 1 | 1 |
| Obj. 2 | 1 | 2 |
| Obj. 3 | 2 | 2 |

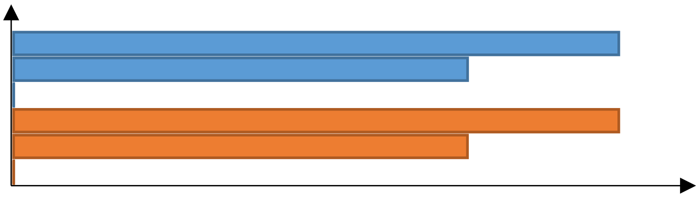
| | Feature 1 | Feature2 |
|---|---|---|
| Obj. 4 | 2 | 3 |
| Obj. 5 | 3 | 3 |
| Obj. 6 | 3 | 4 |

Manhattan/Cityblock/L1/"sum absolute" distance matrix:

| Objects | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 1 | 0 | 1 | 2 | 3 | 4 |
| 3 | 2 | 1 | 0 | 1 | 2 | 3 |
| 4 | 3 | 2 | 1 | 0 | 1 | 2 |
| 5 | 4 | 3 | 2 | 1 | 0 | 1 |
| 6 | 5 | 4 | 3 | 2 | 1 | 0 |

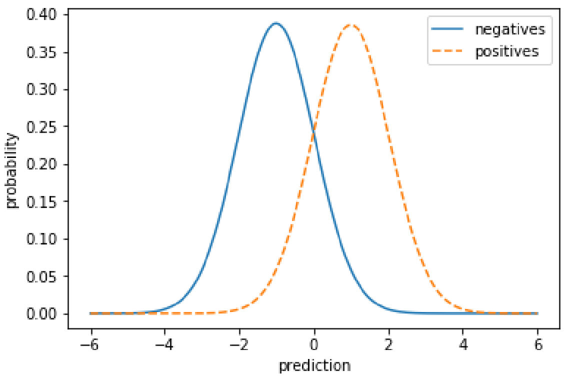
Compute the silhouette values of all observations and sketch the resulting silhouette plot.

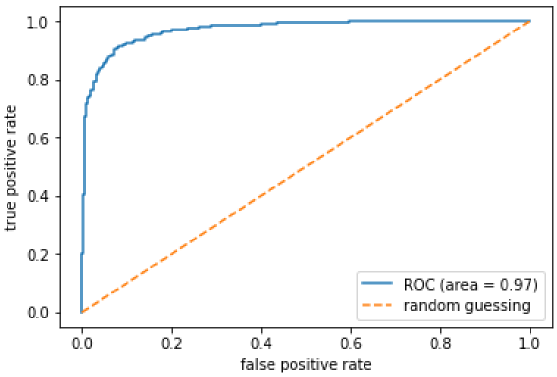
Obj. 1: (4 - 3/2) / 4 = 5/8<br>
Obj. 2: (3 - 1) / 3 = 2/3<br>
Obj. 3: (2 - 3/2) / 2 = 1/4<br>
Obj. 4: (2 - 3/2)/2 = 1/4<br>
Obj. 5: (3 - 1)/3 = 2/3<br>
Obj. 6: (4 - 3/2)/4 = 5/8



## Exercise 6 (15 points in total)

Assume that a classification has been performed where the class decision is made for predictions < 0 (class A) or ≥ 0 (class B). The corresponding class probabilities are shown in the left hand plot and the Receiver Operating Characteristic curve in the right hand plot.





**a) (10 points)**
Explain the basics of the Receiver Operating Characteristic. What happens as you trace the ROC curve from lower left to upper right? (You can use the prediction densities to aid your interpretation.) What does the ROC look like for a perfect classifier? What does it look like for a classifier that is worse than random guessing?

**b) (5 points)**
Imagine a ROC curve only slightly higher than random guessing. What kind of technique could be used to try to leverage such a low performing model?

**a)** ROCs show how the TPR and FPR evolve as one moves the decision border from beyond the positive class to beyond the negative class. The optimal ROC is vertical on the left, kisses the top left corner, and horizontal on the top. An ROC below the diagonal is worse than random guessing.
Starting from the right in the prediction figure, all samples are predicted negative, thus FPR=0 and TPR=0. Moving to the left the TPR increases as positive samples get a positive classification. Around prediction=2, the FPR starts increasing as negative samples are predicted as positive. At prediction=-2, all positive samples are predicted positive, so TPR=1, while FPR increases all the way to the left endpoint of the plot.

**b)** This would be considered a weak learner, meaning that a boosting strategy could help it to perform better.

**See the following link that illustrates the dynamics of the ROC:**
http://www.navan.name/roc/

# Exercise 7 (15 points in total)

We want to compare three basic classifiers: the Perceptron, Adaline and Logistic Regression, but the teacher has been sloppy, mixing his cost functions and activation functions.

$$\boxed{\phi\,(z) = z}$$

$$\boxed{\phi\,(z) = \begin{cases} 1 \; if \; z \geq 0 \\ -1 \; otherwise \end{cases}}$$

$$\boxed{J(\boldsymbol{w}) = \sum_{i=1}^{n}\left[-y^{(i)}\log\left(\phi\left(z^{(i)}\right)\right) - (1-y^{(i)})\log\left(1-\phi\left(z^{(i)}\right)\right)\right]}$$

$$\boxed{J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - \phi(z^{(i)})\right)^2}$$

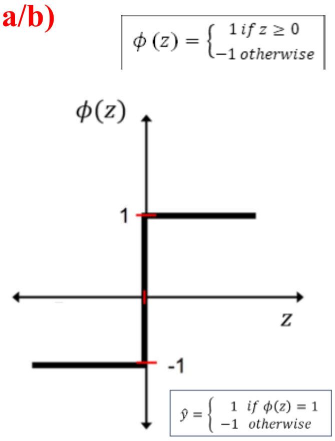
$$\boxed{\phi(z) = \frac{1}{1+e^{-z}}}$$

**a) (5 points)**
Help the teacher associate the correct cost functions and activation functions to the three classifiers.

**b) (5 points)**
Sketch the activation functions and report their threshold functions (the regions leading to different $\hat{y}$ values).

**c) (5 points)**
Sketch a hypothetical cost function curve and explain briefly how gradient decent searches for the minimum

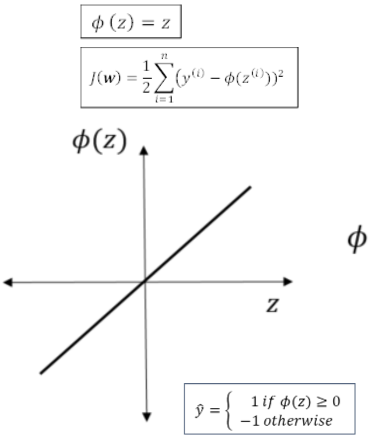
**a/b)**



$$\phi\,(z) = \begin{cases} 1 \; if \; z \geq 0 \\ -1 \; otherwise \end{cases}$$

$$\hat{y} = \begin{cases} 1 \;\; if \;\; \phi(z) = 1 \\ -1 \;\; otherwise \end{cases}$$

Perceptron

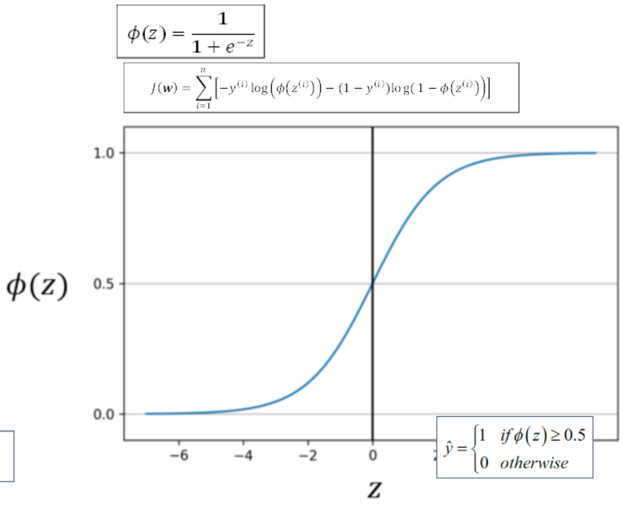


$$\phi\,(z) = z$$

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - \phi(z^{(i)})\right)^2$$

$$\hat{y} = \begin{cases} 1 \; if \; \phi(z) \geq 0 \\ -1 \; otherwise \end{cases}$$

Adaline



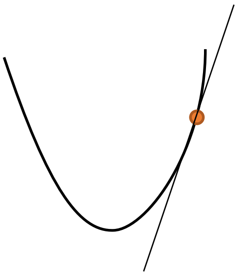
$$\phi(z) = \frac{1}{1+e^{-z}}$$

$$J(\boldsymbol{w}) = \sum_{i=1}^{n}\left[-y^{(i)}\log\left(\phi\left(z^{(i)}\right)\right) - (1-y^{(i)})\log\left(1-\phi\left(z^{(i)}\right)\right)\right]$$

$$\hat{y} = \begin{cases} 1 \;\; if\,\phi(z) \geq 0.5 \\ 0 \;\; otherwise \end{cases}$$

Logistic regression

c) With GD one iteratively moves the parameter estimates a step in the (opposite) direction of the gradient of the cost function to gradually search for an optimum solution with regard to minimizing the cost.



## Exercise 8 (15 points in total)

Using the code below, describe briefly the role of each step of the analysis being performed and precisely what the choice of parameters in each line means (1 point for step 1, 2 points for each of the other steps).

```
# Step 1:
# Import all relevant libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

# Step 2:
# Read standard CSV-data using Pandas
df = pd.read_csv('https://espionage.no/top_secret/surveilance'
                 '/foreign_affairs/spy_list.data')

# Step 3:
# Subset df, OneHot encoding (non-collinear) of ‘spy_network’,
concatenate horizontally
X = pd.concat([df.iloc[:, 2:10].values, \
               pd.get_dummies(df['spy_network'],\
               drop_first=True)], axis=1)
y = df.iloc[:, 0].values

# Step 4:
# Split train test 70:30, stratify on the classes in y (equal
proportions in segments) with fixed random state
X_train, X_test, y_train, y_test = \
    train_test_split(X, y,
                     test_size=0.30,
                     stratify=y,
                     random_state=1)

# Step 5:
# Pipe with centering and standardisation plus SVM (fixed random
state)
pipe_svc = make_pipeline(StandardScaler(),
                         SVC(random_state=1))

# Step 6:
# Set up optimisation grid for C and RBF’s gamma.
param_range_C = \
    [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0]
param_range_g = [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0]
param_grid = [{'svc__C': param_range_C,
               'svc__gamma': param_range_g,
               'svc__kernel': ['rbf']}]

# Step 7:
# Tune hyperparameters using cross-validated gridsearch with 10
segments, single core processing
gs = GridSearchCV(estimator=pipe_svc,
                  param_grid=param_grid,
                  cv=10,
                  n_jobs=1)
```

```
gs = gs.fit(X_train, y_train)

#Step 8:
# Predict test set using best_estimator and compute accuracy
result = gs.best_estimator_.score(X_test, y_test)
```