UNIVERSITY OF FREIBURG

Department of Computer Science

Dr. Joschka Bödecker, Dr. Frank Hutter, Dr. Michael Tangermann

Mockup Exam Machine Learning, Summer Term 2016

Question	Points
1 Short Answers	/20
2 Principal Component Analysis	/10
3 Backpropagation	/10
4 Algorithm-independent principles	/10
Total	/50

Short answer question	a	b	c	d	e	f	g	h	i	j
Points (out of 2 each)										

This mockup exam is a bit shorter than the real exam. The questions in the real exam can cover all topics discussed in the course, and you should by no means focus on the topics given here.

In the real exam there will be a maximum of **70 points** and you have **90 minutes** to answer the questions. You can write your answers either in **English or German**.

This examination is closed book: **you are not allowed to use any notes or calculators**. Please answer questions in the space provided, and if necessary continue your answers on the back of the same sheet. Please also put your name and matriculation number on every page, in case pages get separated.

Your name: _____

Your matriculation number: _____

Your signature: _____

Good luck!

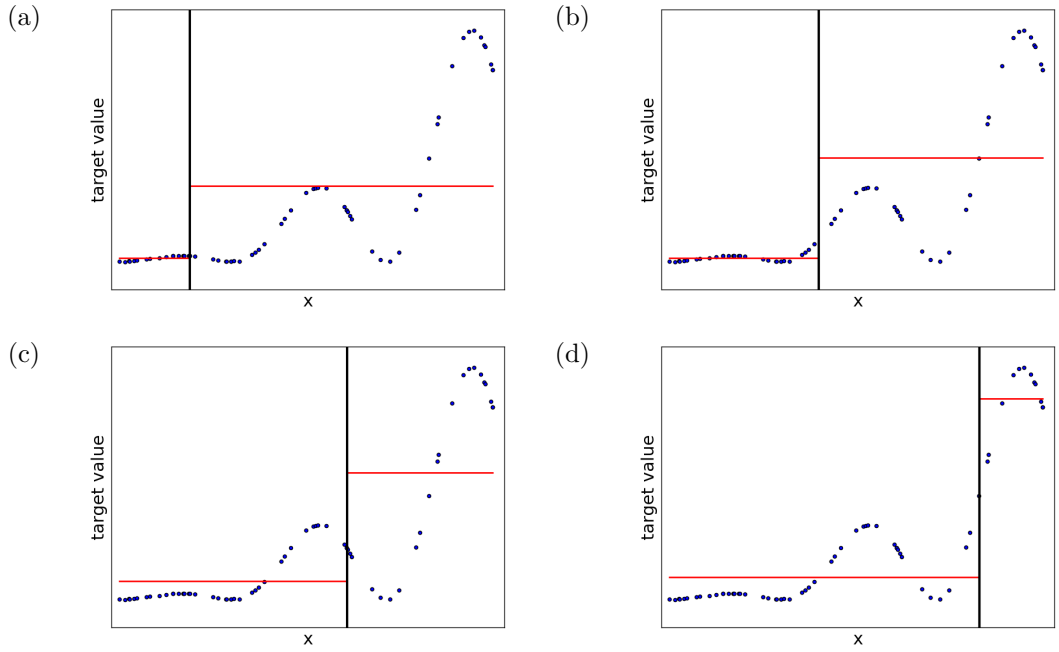
Part 1 – Short questions, short answers

(a) **(2P)** Name the three main types of machine learning tasks.
Solution: *supervised learning, unsupervised learning, reinforcement learning*

(b) **(2P)** Name two possible reasons to use linear regression in contrast to a non-linear fit.
Solution: *Smaller computational complexity, less overfitting due to fewer parameters, interpretability of results (e.g. R^2 value) and analytical solution (no iterative scheme necessary).*

(c) **(2P)** Consider the sample covariance matrix (covariance matrix estimated on data) in LDA. What do you commonly observe for the largest and smallest eigenvalues of the matrix in a high-dimensional space with few data points?
Solution: *Largest eigenvalues will be overestimated, smallest eigenvalues will be underestimated.*

(d) **(2P)** Which of the following four splits is the best with respect to a squared loss in this one-dimensional regression problem? Why?



Solution: *(d), since the sum of the squared differences between the observed data points and the model is lowest.*

(e) **(2P)** In cross-validation, how does the variance of the folds change as the number of folds approaches the number of data points?
Solution: *The variance of the folds' scores increases.*

(f) **(2P)** Which property of the activation function is necessary for applying backpropagation?
Solution: *The activation function must be (piece-wise) differentiable.*

(g) **(2P)** How will a regularization technique for neural networks affect bias and variance if it selectively removes weights of a trained network with minimal effect on training error?
Solution: *Bias will increase, variance will decrease.*

(h) **(2P)** What is the purpose of pooling layers in a ConvNet?
Solution: *Makes representations more manageable and achieves approximate shift invariance.*

(i) **(2P)** How is the final class of a test pattern determined by AdaBoost?
Solution: *Sign (1P) of weighted sum (1P) of classifications.*

(j) **(2P)** Consider the following statement: "The function $\mathbf{k} : \mathbb{R}^N$ x $\mathbb{R}^N \rightarrow \mathbb{R}$ with $\mathbf{k}(x, y) = \frac{x^T y}{||x|| \cdot ||y||}$ defines a positive definite kernel function." Is this true or false? Discuss why.
Solution: *True, see Assignment 11.*

Part 2 – Dimensionality reduction with PCA for classification (10P)

Your task is to write pseudocode for a classification problem. The script should output the estimated classification accuracy on unseen data. Provided are the matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$, containing the dataset (N number of samples and d number of features, standardized with zero mean and unit variance), and vector $\boldsymbol{y} \in \{0, 1\}^N$, containing the corresponding labels. Several functions are available for its use in the pseudo-script:

`computePCA`
Input:
$\boldsymbol{C_x} \in \mathbb{R}^{d \times d}$, covariance matrix of the original data.
Output:
$\boldsymbol{\lambda} \in \mathbb{R}^d$, vector containing the eigenvalues of $\boldsymbol{W}$, sorted in descending order
$\boldsymbol{W} \in \mathbb{R}^{d \times d}$, matrix containing the corresponding eigenvectors of $\boldsymbol{C_x}$.

`trainClassifier`
Input:
$\boldsymbol{X} \in \mathbb{R}^{N \times d}$, matrix containing the dataset to train the classifier (training set).
$\boldsymbol{y} \in \{0, 1\}^N$, vector containing the corresponding labels for each row of $\boldsymbol{X}$.
Output:
Model: Trained classifier.

`applyClassifier`
Input:
Model, classifier model as returned by `trainClassifier`.
$\boldsymbol{X} \in \mathbb{R}^{N \times d}$, dataset on which the labels will be predicted (test set).
Output:
$\boldsymbol{y} \in \{0, 1\}^N$, predicted labels.

(a) **(9P)** On the next page, write pseudo-code solving the scenario described above. Include a step of dimensionality reduction using PCA. Choose the abstraction level of your pseudo-code such that your script contains a maximum number of 15 lines.

Solution: Here, we give actual code, but any type of high-level pseudocode would be fine.

```
accuracy = 1
for idx_sample in 0:X.shape[0]: # Leave one out cross-validation
    # Split train and test data
    X_tr, y_tr = X, y;
    X_tr[idx_sample,:] = []
    Y_tr[idx_sample] = []
    X_te, y_te = X[idx_sample,:], y[idx_sample]

    W, lambda = computePCA(dot(X_tr.T,X_tr))

    # cumulative variance crit.
    ncomp = find(cumsum(lambda)/sum(lambda) >= 0.9)[0]

    model = trainClassifier(dot(X_tr,W[:,0:ncomp]), y_tr)

    y_predict = applyClassifier(model,dot(X_te,W[:,0:ncomp]))

    If y_te != y_predict: accuracy = accuracy - 1/X.shape[0]
```

Scoring:

- **3P** Cross-Validation including accuracy assessment.
- **3P** for the right implementation of PCA projection.
- **3P** for PCA component selection.

(b) **(1P)** Explain your strategy to select the number of PCA components to perform the dimensionality reduction step.
Solution: *The cumulative variance criterion is used to select the number of components: The number of components is set to preserve 0.9 of the original variance. Other strategies include the selection of the eigenvalues that are significantly large, in contrast to the smallest of the set.*

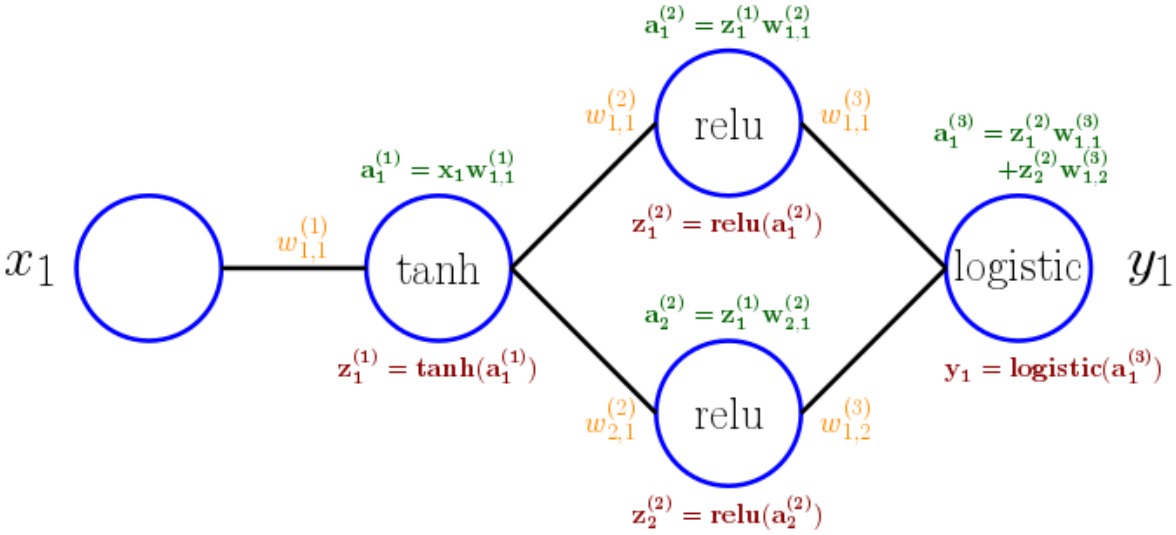
Part 3 – Backpropagation (10P)

(a) **(2P)** Bring the following steps of the Backpropagation algorithm into the right order:

A - calculate the error between network output and target value
B - for each weight: calculate the partial derivatives and add up their values
C - for each neuron: calculate the activation value
D - apply the input vector of a training example to the input neurons of the MLP

Solution: *D - C - A - B*

Next, you will compute symbolic expressions (no actual numbers!) of the partial derivatives w.r.t. the weights of the third layer and w.r.t. the outputs of the second layer in the following MLP:



Assume that the cross-entropy (CE) loss is used, but simply make use of

$$\frac{\partial CE}{\partial y_1}\frac{\partial y_1}{a_1^{(3)}} = \mathrm{logistic}(a_1^{(3)}) - t_1$$

in your symbolic expression.

(b) **(4P)** Partial derivatives w.r.t. the weights of the third layer:
Solution:

$$\frac{\partial CE}{\partial w_{1,1}^{(3)}} = \frac{\partial CE}{\partial y_1}\frac{\partial y_1}{\partial a_1^{(3)}}\frac{\partial a_1^{(3)}}{\partial w_{1,1}^{(3)}} = \left(logistic(a_1^{(3)}) - t_1\right) z_1^{(2)} \quad (1)$$

$$\frac{\partial CE}{\partial w_{1,2}^{(3)}} = \frac{\partial CE}{\partial y_1}\frac{\partial y_1}{\partial a_1^{(3)}}\frac{\partial a_1^{(3)}}{\partial w_{1,2}^{(3)}} = \left(logistic(a_1^{(3)}) - t_1\right) z_2^{(2)} \quad (2)$$

(c) **(4P)** Partial derivatives w.r.t. the outputs of the second layer:
Solution:

$$\frac{\partial CE}{\partial z_1^{(2)}} = \frac{\partial CE}{\partial y_1}\frac{\partial y_1}{\partial a_1^{(3)}}\frac{\partial a_1^{(3)}}{\partial z_1^{(2)}} = \left(logistic(a_1^{(3)}) - t_1\right) w_{1,1}^{(3)} \quad (3)$$

$$\frac{\partial CE}{\partial z_2^{(2)}} = \frac{\partial CE}{\partial y_1}\frac{\partial y_1}{\partial a_1^{(3)}}\frac{\partial a_1^{(3)}}{\partial z_2^{(2)}} = \left(logistic(a_1^{(3)}) - t_1\right) w_{1,2}^{(3)} \quad (4)$$

Part 4 – Algorithm Independent Principles (10P)

Your friend works at a company that develops weather prediction systems. There is one data point per day, and predictions shall be made for the next days. Now your friend wants to try out different types of models and evaluate their performance using cross-validation.

1. **(2P)** Are the individual data points in your friend's data independently distributed? Why or why not?
   Solution: *No. E.g., learning about tomorrow's weather changes our belief about the weather two days from now.*

2. **(2P)** What problems would occur when using standard k-fold cross-validation for this data?
   Solution: *Overfitting. No predictive capability to future data points.*

3. **(6P)** Explain which changes are necessary to the following regular cross validation code (words are enough, you don't need to correct the code) to suit your friend's time series data.

```
num_points = len(X)
num_folds = 10
# Shuffle data to make sure there are no sorting artifacts
indices = np.shuffle(range(num_points))
X = X[indices]
y = y[indices]

length_fold = floor(num_points / num_folds)
fold_indicators = [floor(i / length_fold) for i in range(num_points)]
for fold in range(num_folds):
    X_train = X[fold_indicator != fold]
    y_train = X[fold_indicator != fold]
    X_valid = X[fold_indicator == fold]
    y_valid = X[fold_indicator == fold]

    model.fit(X_train, y_train)
    scores.append(model.score(X_valid, y_valid)
```

   Solution:

   (a) Don't shuffle the data.

   (b) Possibly: Divide the data in blocks which are of the same length as the prediction horizon.

   (c) Only train on folds which have a lower index than the current one; change equal comparison to lower than comparison.