Statistical Learning Exam 6 January 2016
14:00 – 17:00

- You are allowed to use the book, your private notes and printouts of the handwritten lecture notes from the course website.
- Every subquestion (1a, 1b, etc.) counts for the same number of points.
- You have to hand in the Appendix. Do not forget to add your name and student number!

1. **[Classification]** See the data set in Figure 1 in the Appendix.
   (a) How many mistakes does 1-nearest neighbour make when we train it on this data and then evaluate its total number of mistakes in classifying the same data?
   (b) How about 15-nearest neighbour?
   (c) What is the smallest number of mistakes on this training data that can be achieved by a linear classifier *with* an intercept without transforming the features? (That is, set $f_\beta(\boldsymbol{x}) := \beta_0 + \beta_1 x_1 + \beta_2 x_2$ and predict with $\text{sign}(f_\beta(\boldsymbol{x}))$.) Draw the corresponding decision boundary in Figure 1a.
   (d) What is the smallest number of mistakes that can be achieved *without* an intercept? (That is, $f_\beta(\boldsymbol{x}) := \beta_1 x_1 + \beta_2 x_2$.) Draw the corresponding decision boundary in Figure 1b.
   (e) The line in Figure 1c is given by $x_1^2 - 8x_1 + 4x_2 - 8 = 0$. How can we transform the features such that it is possible for a linear classifier on the transformed features, with an intercept, to make 0 mistakes on this data? Explain your answer.
   (f) Which kernel for a support vector machine (SVM) would correspond to this transformation?

2. **[Decision Trees and Regression]** Suppose we have some regression data set $\mathcal{T} = \begin{pmatrix} y_1 \\ \boldsymbol{x}_1 \end{pmatrix}, \ldots, \begin{pmatrix} y_N \\ \boldsymbol{x}_N \end{pmatrix}$ in which each feature vector $\boldsymbol{x}_i$ consists of two features.
   (a) Using the CART algorithm for regression trees described in Section 9.2.2 of the book, is it ever possible to end up with a tree that partitions the feature space as in Figure 2a in the Appendix? How about Figure 2b?
   (b) Suppose the second feature was originally measured in meters, but we transform it to be measured in centimeters. This has the effect of multiplying this feature by 100, while leaving the other feature unchanged. Let T_{before} be the regression tree produced by CART before transforming the second feature, and let T_{after} be the tree produced after transforming it. Then are T_{before} and T_{after} different in the sense that they produce a different regression estimate when we apply T_{before} to a new feature vector $\boldsymbol{x}$ and T_{after} to the transformed version of $\boldsymbol{x}$? Explain your answer.
   (c) If you answered "yes" on the previous question, then give an example of a regression method that does *not* produce a different regression function when we scale the second feature. If you answered "no", then give an example of a regression method that *does* produce a different regression function when we scale the second feature.
   (d) Suppose we grow a large regression tree T_0 on the data $\mathcal{T}$ using the CART algorithm without pruning. To complete the CART algorithm, we then want to prune the tree down to a subtree $T_\alpha \subseteq T_0$ that minimizes the cost complexity criterion (9.16) in the book. But this criterion

depends on a tuning parameter $\alpha \geq 0$, so we perform 5-fold cross validation by randomly partitioning $\mathcal{T}$ into 5 subsets of (approximately) equal size. For each α from a range of values $\alpha = 0.01, 0.02, \ldots, 10.00$, we minimize (9.16) over 4 of the subsets and we measure the prediction error of the resulting tree T_α on the remaining hold-out subset. We then average the prediction errors over the 5 possible choices of the hold-out subset, and select the α that predicts best on average. What is wrong with this way of doing cross-validation and what would be the correct way? (Assume that the chosen range of values for α is appropriate.)

3. **[Clustering]** Consider clustering with the K-means algorithm.

   (a) If we choose the number of clusters K to be between 1 and 10, which value for K do you expect to give the smallest sum of squared distances

   $$\sum_{k=1}^{K} \sum_{i:C(i)=k} \|\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k\|^2 \tag{1}$$

   for the data in Figure 3 after running K-means? Here $C(i)$ is the cluster assigned to data point $\boldsymbol{x}_i$, and $\bar{\boldsymbol{x}}_k$ is the mean of the k-th cluster. NB Don't forget that the book contains a mistake in the definition of K-means: K-means aims to minimize (1), not (14.31) from the book.

   Suppose we get classification data with two classes, so now we are in a supervised learning setting with labeled data

   $$\begin{pmatrix} y_1 \\ \boldsymbol{x}_1 \end{pmatrix}, \ldots, \begin{pmatrix} y_N \\ \boldsymbol{x}_N \end{pmatrix}.$$

   We want to pre-process our data to add an extra feature as follows: First we cluster the feature vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ with K-means for $K = 2$, which gives us 2 cluster means $\bar{\boldsymbol{x}}_1$ and $\bar{\boldsymbol{x}}_2$. And then, for any feature vector $\boldsymbol{x}$, we add an additional feature which is the index of the cluster mean that is closest to $\boldsymbol{x}$. So, for example, if $\boldsymbol{x}$ is closer to $\bar{\boldsymbol{x}}_2$ than to $\bar{\boldsymbol{x}}_1$, then we extend $\boldsymbol{x}$ with an extra feature that has value 2.

   (b) Give a probability distribution for $(\boldsymbol{x}, y)$ for which you expect the extra feature to be useful for classification, and explain your answer. As part of your explanation, draw an approximate picture of the type of data generated by your probability distribution.

   (c) Give a probability distribution for $(\boldsymbol{x}, y)$ for which you do *not* expect the extra feature to be useful for classification, and explain your answer. As part of your explanation, draw an approximate picture of the type of data generated by your probability distribution.

4. **[General]**

   (a) Kaggle.com organizes online machine learning competitions on supervised learning tasks like classification, regression, etc. For each competition they split their data into three parts: a train set, which is released to the competitors; a test set which is not released and which is used to evaluate the submitted methods once the competition has finished; and finally a small *leader board* set, which is also not released. While the competition is running, competitors can make a preliminary submission and then their error on the leader board set is published on the leader board web page, which thereby acts as a high score list.

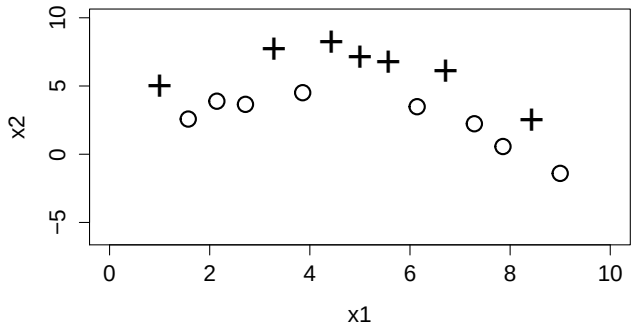
   When kaggle.com started their competitions, they often observed the following phenomenon: some competitors would make many preliminary submissions with different settings for a large number of hyperparameters[1]. They would then find hyperparameters for which they got a very small error on the leader board data set, and therefore appear to be much better than everybody else. But after the competition finished, these competitors performed much worse on the test data than on the leader board data, and ended up losing to other competitors that did not make so many preliminary submissions. Please explain what was happening.

[1] Hyperparameters are, for example, the parameter λ in ridge regression or the Lasso, or the value for K in K-nearest neighbour.

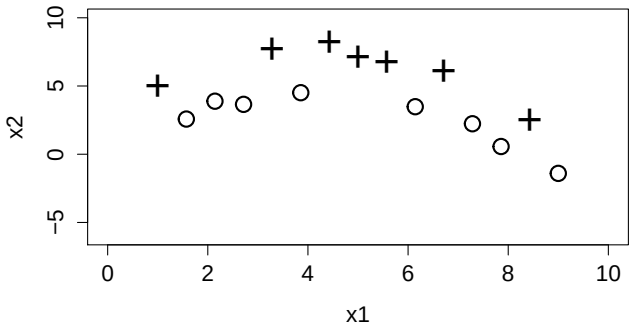
Appendix to the Statistical Learning Exam 6 January 2016

Name:

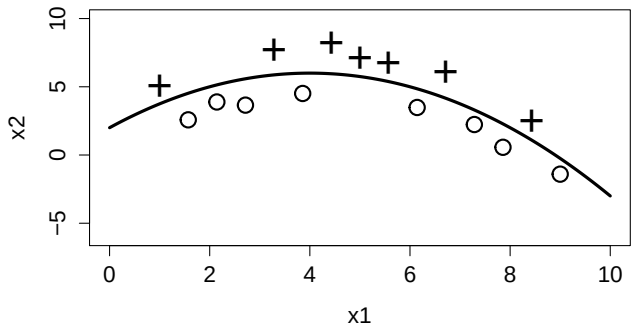
Student number:



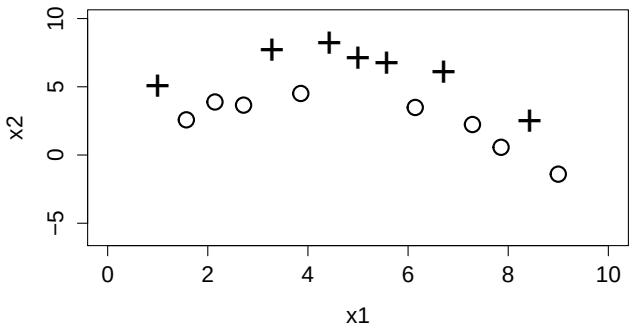
(a) Smallest nr. of mistakes with intercept



(b) Smallest nr. of mistakes without intercept

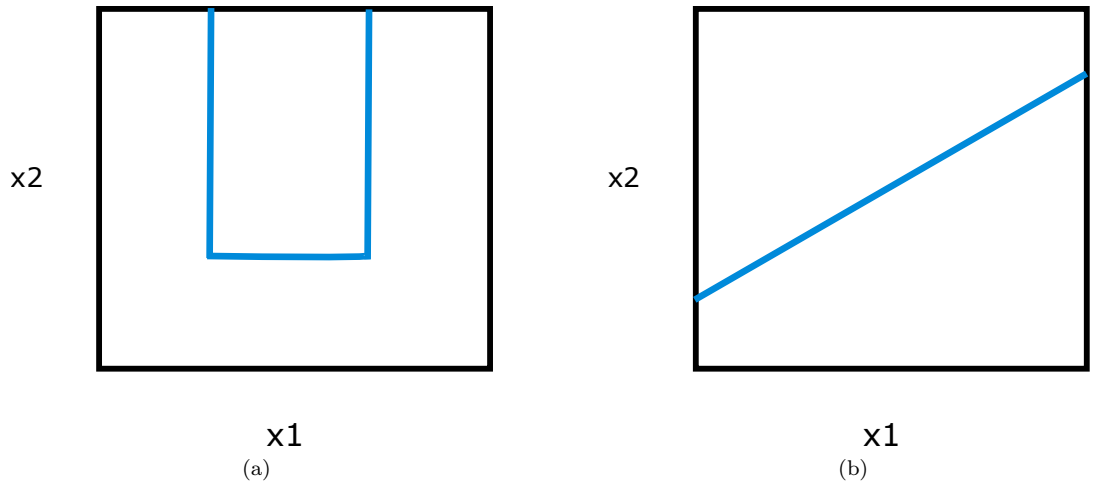


(c) A line that perfectly separates the two classes



(d)

Figure 1: Four times the same classification data set for binary classification with two features. For concreteness, let us say that the '+'-symbols have label +1 and the circles have label −1. The exact location of each symbol is its center. (You can use Figure 1d as a backup in case you make a drawing mistake in one of the other figures.)



(a) (b)

Figure 2: A feature space with two features, partitioned in two different ways into regions

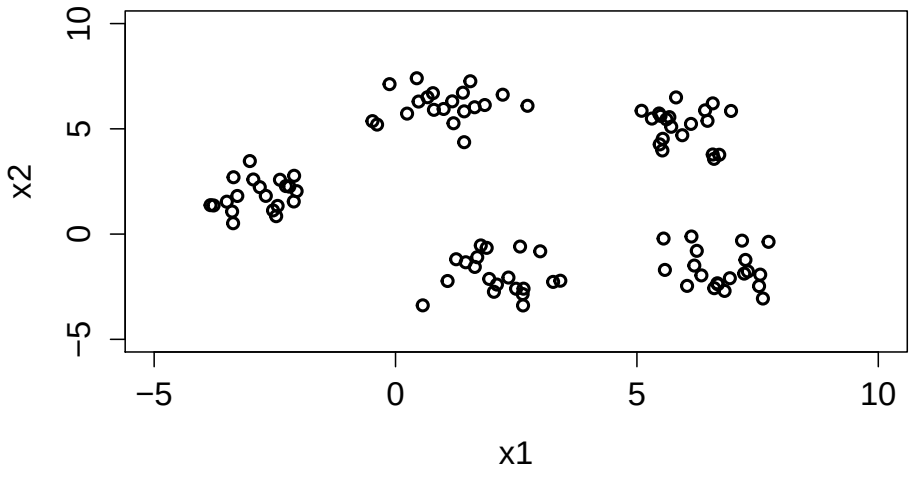


Figure 3: Unsupervised data with two features without any labels