Linnæus University
Department of Computer Science
Jonas Nordqvist
Jonas Lundberg
Rafael Messias Martins

Exam in Introduction to Machine Learning, 2DV516, 7.5 hp
June02, 2021, 10.00–15.00

The exam consists of 7 questions. The maximum number of points for the exam is 36.

1. In Figure 1 a neural network with three binary inputs (x_1, x_2, x_3), one hidden layer with two nodes and one output is given. All activation functions $f_h(x)$ in the hidden units and the output unit are step functions defined as

$$f_h(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise.} \end{cases}$$

The network should produce the output of the following function

$$y(x) = \begin{cases} 1, & \text{if } 1 \leq x_1 + x_2 + x_3 \leq 2 \\ 0, & \text{otherwise.} \end{cases} \quad . \tag{6p}$$

(a) Find weights and biases solving this problem.

(b) Produce the matrices $W^{(2)}$ and $W^{(3)}$ which carries all the weights for your answer in a).

(c) Is this a linear problem in the sense that a classifier producing a linear decision boundary could solve the problem? Motivate your answer!

Hint for (a): one hidden unit should activate if all inputs are active, and one should activate if none are active.

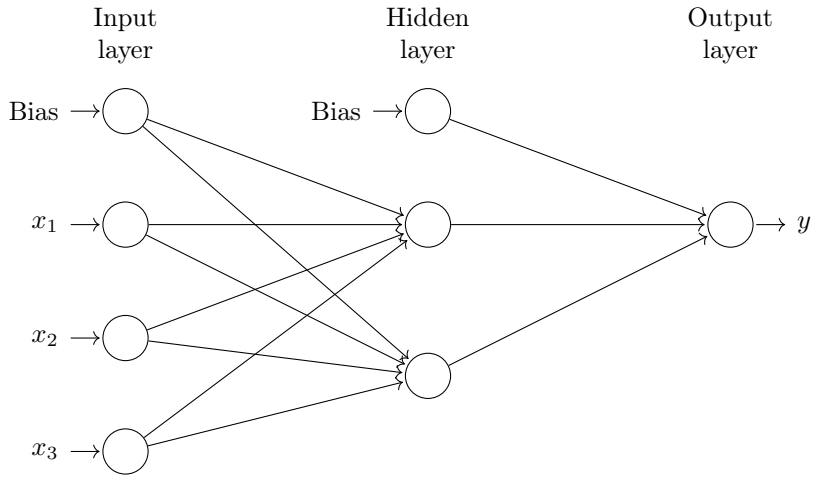


Figure 1: A neural network with one hidden layer consisting of two nodes.

2. Suppose that you've trained five models on the same training data. All the models achieve 95% accuracy. (4p)

(a) Is there any chance that you can combine these models to get better results? Motivate your answer.

(b) If you can improve the results, explain how and why this is achievable. If you cannot improve the results, explain why.

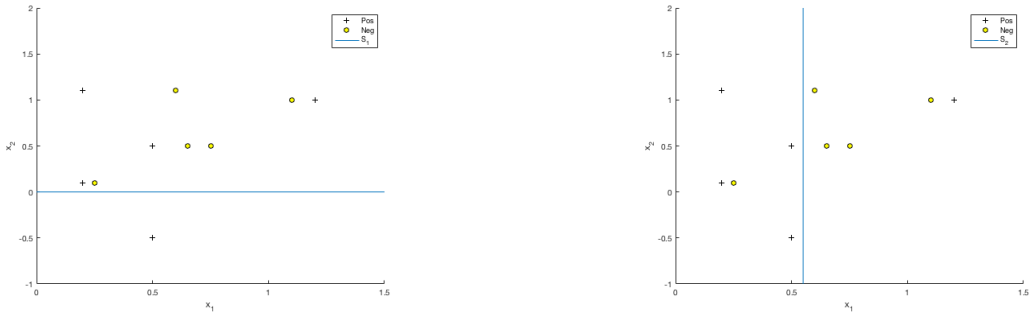
3. Recall that the purity gain of a binary split in a decision tree is computed as follows

$$\Delta = I(\text{parent}) - \sum_{j=1}^{2} \frac{N(v_j)}{N} I(v_j),$$

where I is an impurity measure, and $N(v_j)$ is the number of elements in the node v_j (*i.e.* the jth subregion after the split), and N is the number of elements in the parent node (*i.e.* the region in which we perform the split). Suppose that our impurity measure is the Gini index which is defined, for the mth region, by

$$G_m = 1 - \sum_{k=1}^{K} \widehat{p}_{mk}^2,$$

where $\widehat{p}_{mk}$ represents the proportion of training observations in the mth region that are from the kth class. Given the two-class data in Figure 2 compute the purity gain from both the splits S_1 and S_2. Based on your computations, which is the preferred split? (6p)



(a) Binary split S_1 (b) Binary split S_2

Figure 2: Two different splits in a decision tree

4. Consider a three-class classification task with only four training examples as seen in Figure 3. Suppose that a maximal margin classifier is used and trained on the mentioned training examples, together with either (i) one-vs-one binarization or (ii) one-vs-all binarization. (6p)

(a) In each of these cases (i) and (ii). How many classifiers needs to be trained?

(b) If this was a four-class problem, how many classifiers would be trained in each of the cases (i) and (ii)?

(c) Are the same decision boundaries produced by (i) and (ii)? Motivate your answer with illustrations.

(d) In this particular example is either (i) or (ii) preferred over the other? Motivate your answer. (Note that one is not preferred over the other in an objective sense, so it is the argument itself which is important)

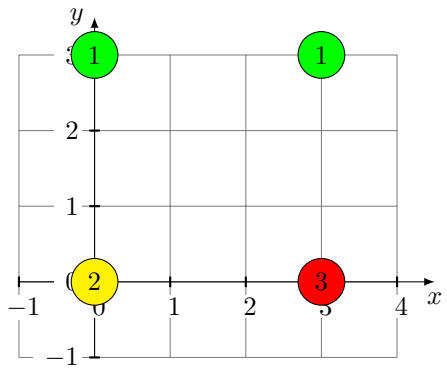


Figure 3: Four training examples from the classes with labels $1, 2$ and 3.

5. Suppose that we are dealing with data which we want model using ordinary linear regression $f(x) = \beta_1 + \beta_2 x$. The parameters β_1 and β_2 can be found by minimzing the mean-squared error

$$J(\beta_1, \beta_2) := \frac{1}{n} \sum_{j=1}^{n} (\beta_1 + \beta_2 x_j - y_j)^2,$$

using gradient descent (GD).

The GD-algorithm is presented in Algorithm 1 below. Suppose that the data consists of three points $x_1 = 1, x_2 = 2, x_3 = 3$ with corresponding labels $y_1 = 2$, $y_2 = 3.5$, $y_3 = 5.5$. Using

$$\gamma = 0.01, \quad \beta_1 = 0.5, \quad \beta_2 = 2$$

compute one update of the gradient descent. (4p)

The following computations may be useful for you

$$\frac{1}{3} \sum_{j=1}^{3} (0.5 + 2x_j - y_j)^2 = 3/4$$

$$\frac{1}{3} \sum_{j=1}^{3} (0.5 + 2x_j - y_j) = 5/6$$

$$\frac{1}{3} \sum_{j=1}^{3} x_j (0.5 + 2x_j - y_j) = 11/6.$$

Algorithm 1 Gradient descent

1: **for** *iteration* $= 1, 2, \ldots$ **do**
2: $\quad \text{tmp}_1 = \beta_1 - \gamma \frac{dJ(\beta)}{d\beta_1}$
3: $\quad \text{tmp}_2 = \beta_2 - \gamma \frac{dJ(\beta)}{d\beta_2}$
4: $\quad (\beta_1, \beta_2) = (\text{tmp}_1, \text{tmp}_2)$
5: **end for**

6. Figure 4 is an example of the results of applying two different dimensionality reduction methods (PCA and Sammon mapping) to the same dataset.

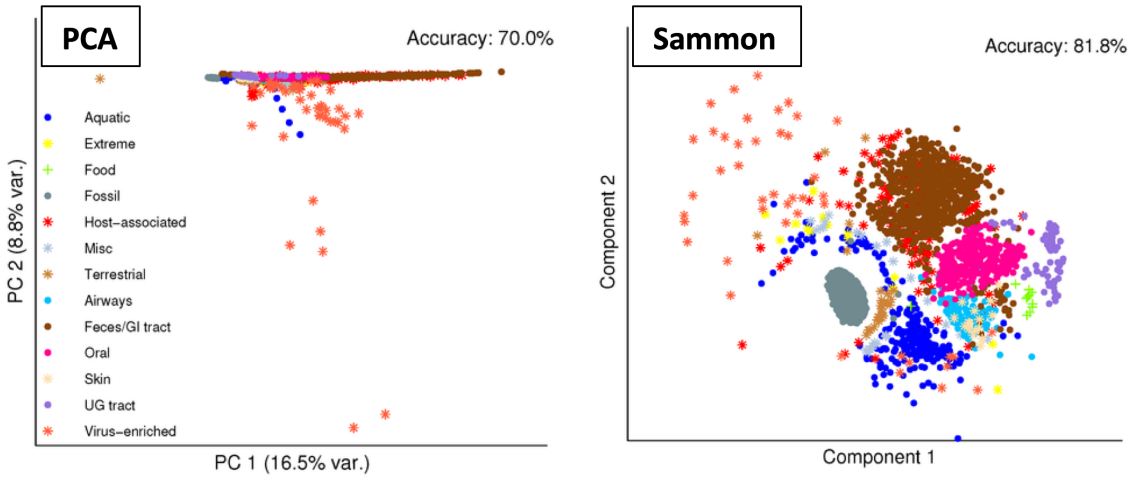


Figure 4: PCA vs. Sammon mapping. Source: `https://www.mdpi.com/1422-0067/15/7/12364`

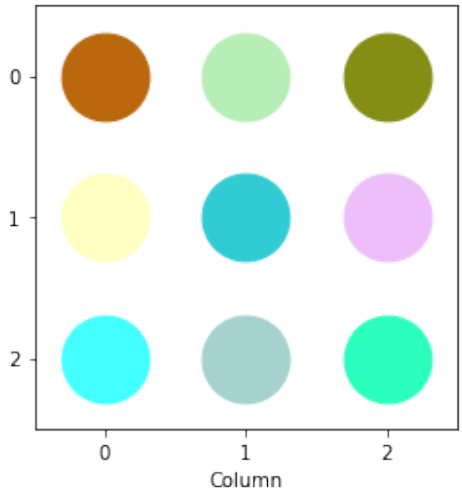
The questions below are regarding the methods in general, not the figure specifically (you should use the figure as an example only).

(a) What are the main differences between these two methods, which cause them to yield so different results? (Tip: think about the optimization process involved in both cases) (2p)

(b) What is the unique characteristic of Sammon mapping that might make it more attractive than PCA in some occasions? (2p)

(c) Why would an analyst choose a complex non-linear DR (such as Sammon or t-SNE) over a simple linear one (such as PCA)? (1p)

7. Figure 5a contains a hidden face of a dice, obfuscated by adding noise to its original colors (and also to the original white background). The original, hidden face is similar to one of the examples in Figure 5c, but with different colors. The exact RGB values of each circle of the obfuscated image are given in Figure 5b.



(a) The obfuscated dice face

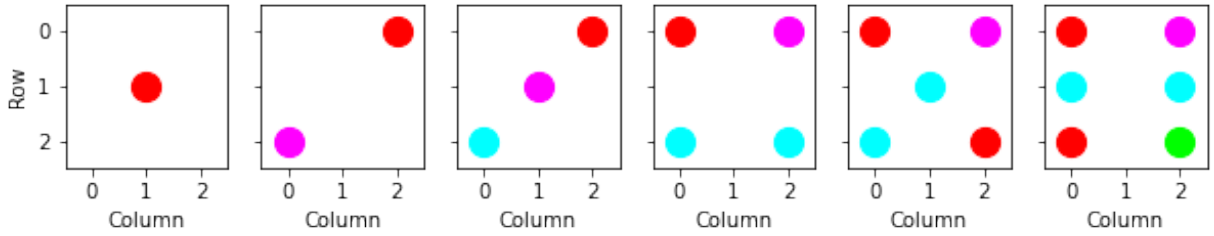
```
// R       G       B

[[0.7303, 0.4031, 0.0515],     // Row 0, Col 0
 [0.7172, 0.9339, 0.7101],     // Row 0, Col 1
 [0.5179, 0.5511, 0.0794]],    // Row 0, Col 2

[[1.    , 1.    , 0.7694],     // Row 1, Col 0
 [0.1922, 0.8018, 0.8286],     // Row 1, Col 1
 [0.931 , 0.7433, 0.9749]],    // Row 1, Col 2

[[0.2654, 1.    , 1.    ],     // Row 2, Col 0
 [0.6475, 0.819 , 0.8098],     // Row 2, Col 1
 [0.1723, 1.    , 0.7405]]]    // Row 2, Col 2
```

(b) The RGB values of the obfuscated image



(c) Examples of possible solutions (with different colors)

Figure 5: Exercise 7: Find the hidden dice.

Your task for this question is to find the hidden face using Agglomerative Clustering (with *single* linkage), then report on the process you followed. Consider each circle as a 3D vector of RGB components. You will find the solution by clustering the circles into an appropriate number of clusters (which you do not know from the start, so you need to explore all possibilities). Note that the white background also counts as one cluster (i.e. some of the circles from the obfuscated image are hiding the white background, and thus they will be joined in a single cluster when the right solution is found).

(a) Draw the final obtained face, with an indication of which pixels belong to which cluster. The exact colors of each cluster (or any color at all) are not important. (3p)

(b) Draw a dendogram of the clusters of pixels. The "height" of the clusters do not need to be exact. (2p)

This exercise was designed to have one (and only one) solution, but if you find more than one possible solutions (or none), report one of them, and you will be scored fairly based (mainly) on the dendrogram.

Good Luck!