Machine Intelligence 1 **Exercise Sheet 2**
WS 2019/20, Obermayer/Kashef due: 31.10.2018 at 23:55

Connectionist Neurons and Multi Layer Perceptrons

Please remember to upload exactly *one* ZIP file per group and name the file according to the respective group name: `yourgroupname.zip`
The ZIP file should contain a single Jupyter notebook source file as well as a single PDF file that is generated from the Jupyter notebook. Please do **not** include any folder structure, exercise PDF or data files.

Exercise T2.1: Terminology (tutorial)

(a) What does a connectionist neuron compute?

(b) Which effect do the *weights* and the *bias* have?

(c) Why is a nonlinear transfer function beneficial compared to a linear one?

(d) What is a feedforward multilayer perceptron (MLP)?

Exercise H2.1: Connectionist Neuron (homework, 6 points)

The dataset[1] `applesOranges.csv` contains 200 measurements (`x.1` and `x.2`) from two types of objects as indicated by the column `y`. In this exercise, you will use a connectionist neuron with a "binary" transfer function $f(h)$ to classify the objects, i.e., obtain the predicted class y for a data point $\underline{\mathbf{x}} \in \mathbb{R}^2$ by

$$y(\underline{\mathbf{x}}) := f(\underline{\mathbf{w}}^\top \underline{\mathbf{x}} - \theta)$$

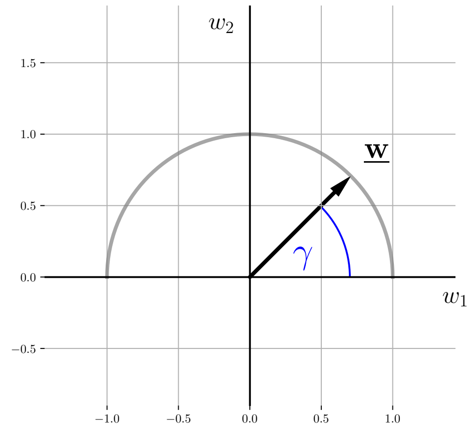
with

$$f(h) := \begin{cases} 1 & \text{for } h \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

where $h := \underline{\mathbf{w}}^\top \underline{\mathbf{x}} - \theta$ is the total input to the neuron.

(a) Plot the data in a scatter plot (x_2 vs. x_1). Mark the points with different colors to indicate the type of each object.

[1] This data file (and those required for future exercise sheets) is available on ISIS.



(b) Set the bias $\theta = 0$. Create a set of 19 weight vectors $\underline{\mathbf{w}} = (w_1, w_2)^\top$ pointing from the origin to the upper semi-circle with radius 1 (i.e. if γ denotes the angle between the weight vector and the x-axis, for each $\gamma = 0, 10, \ldots, 180$ (equally spaced) such that $||\underline{\mathbf{w}}||_2 = 1$, $w_1 \in [-1, 1], w_2 \in [0, 1]$).
For each of these weight vectors $\underline{\mathbf{w}}$,

(i) determine the classification performance ρ (% correct classifications) of the corresponding neuron and

(ii) plot a curve showing ρ as a function of γ.

(c) From the weight vectors generated above, pick the $\underline{\mathbf{w}}$ that yields the best performance. Now vary the bias $\theta \in [-3, 3]$ and pick the value of θ that gives the best performance.

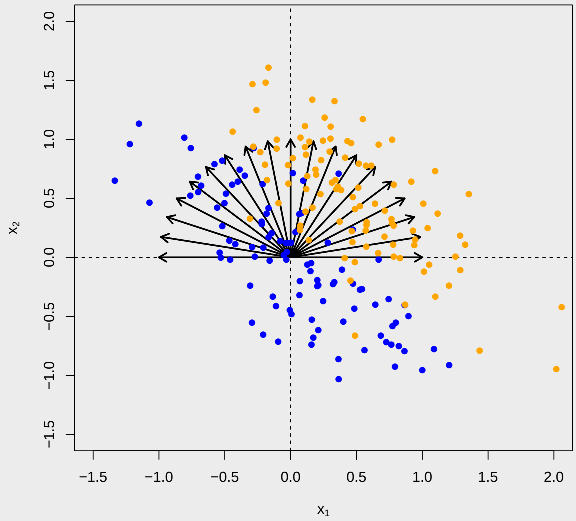
(d) Plot the data points and color them according to the predicted classification when using the $\underline{\mathbf{w}}$ and θ that led to the highest performance. Plot the weight vector $\underline{\mathbf{w}}$ in the same plot. How do you interpret your results?

(e) Find the best combination of $\underline{\mathbf{w}}$ and θ by exploring all combinations of γ and θ (within a reasonable range and precision). Compute and plot the performance of all combinations in a heatmap.

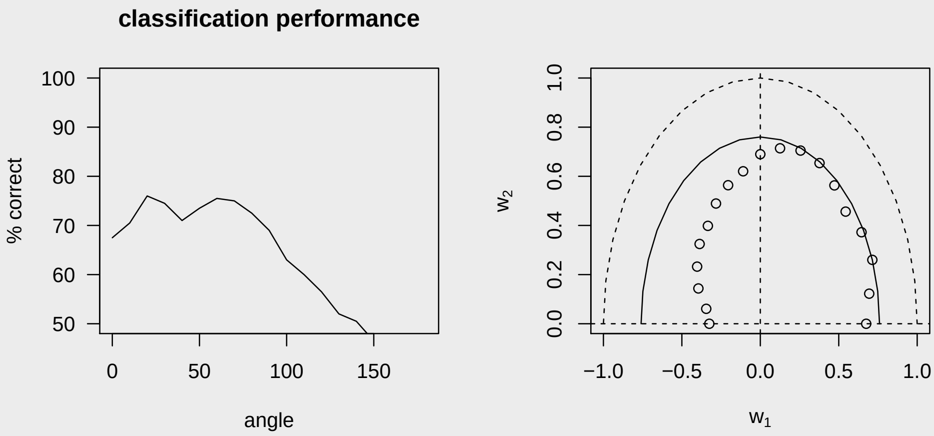
(f) Can the optimization procedure used in (e) be applied to any classification problem? Discuss potential problems and give an application example in which the above method must fail.

Solution

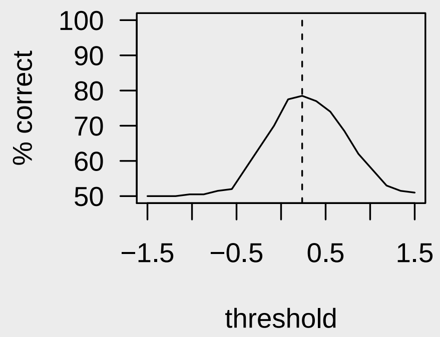
(a) The labelled data and the 19 independent weight vectors from (b).



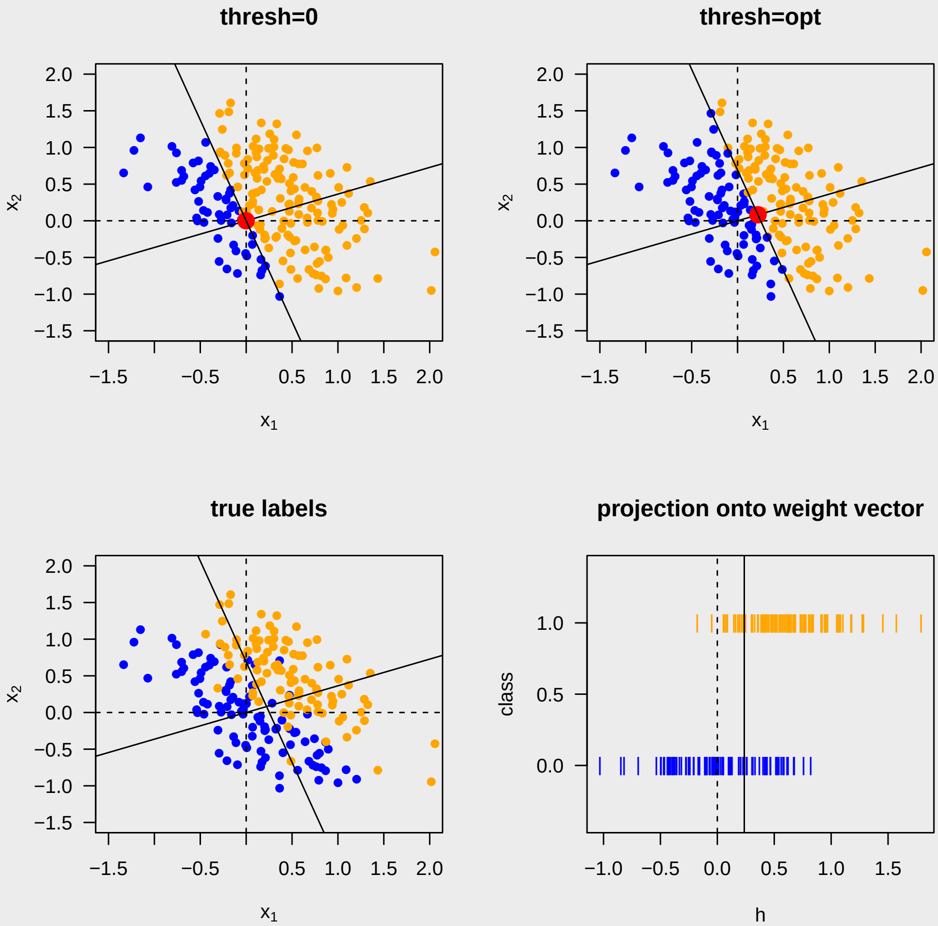
(b) The left plot shows the classification performance vs. the angle α, and the right plot shows that performance in a radial plot (circles).



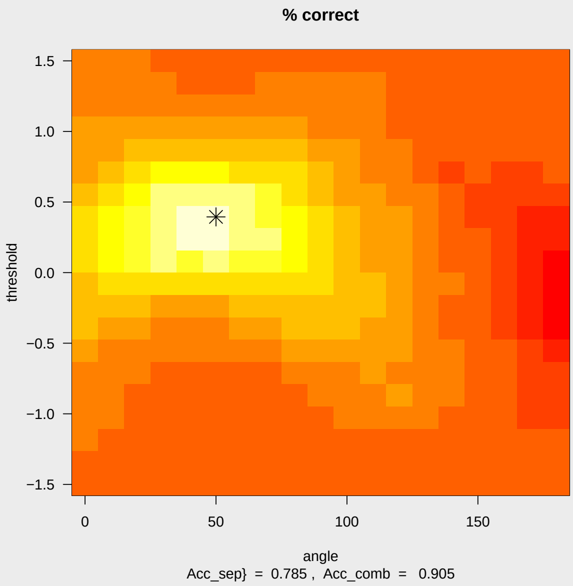
(c) The performance at the best angle from (b) for varying thresholds θ.



(d) The decision boundaries for thresholds $\theta = 0$ (upper left plot), and the optimal θ from (c) (upper right plot). The latter is plotted against the original labels in the lower left. The lower right plots the projection of the data points onto the weight vector. The data is not linearly separable.



(e) The heat-map of correct classification for range of angles (x-axis) and thresholds (y-axis):



(f) Grid-search fails in high dimensional input spaces, as the number of measured combinations grows exponential in the inputs dimensionality (e.g. pixel images → millions of weights).

Exercise H2.2: Multilayer Perceptrons (MLP) (homework, 4 points)

For an MLP with input $x \in \mathbb{R}$ and 1 hidden layer and 1 output node. The input-output function can be computed as

$$y(x) = \sum_{i=1}^{N_{\text{hid}}} w_{1i}^{21} f(w_{i1}^{10} x - b_i))$$

with output weights w_{1i}^{21} and parameters w_{i1}^{10} and b_i for the i-th hidden unit. The output node has no bias.

(a) Create 50 independent MLPs with $N_{\text{hid}} = 10$ hidden units by sampling for each MLP a set of random parameters $\{w_{1i}^{21}, w_{i1}^{10}, b_i\}, i = 1, ..., 10$.
- Use $f(\cdot) := \tanh(\cdot)$ as the transfer function.
- Use normally distributed $w_{1i}^{21} \sim \mathcal{N}(0, 1)$
- Use normally distributed $w_{i1}^{10} \sim \mathcal{N}(0, 2)$ and uniformly distributed $b_i \sim \mathcal{U}(-2, 2)$.

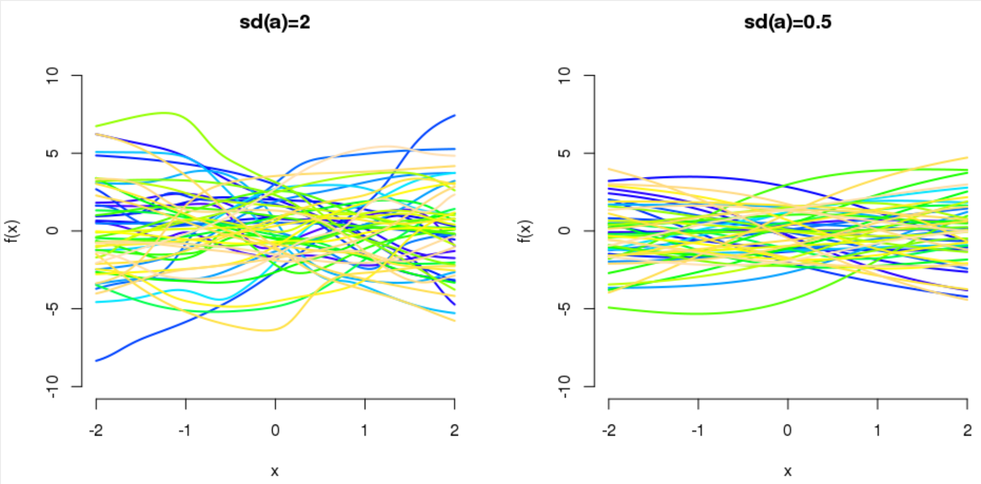
(b) Plot the input-output functions (i.e. the response $y(x)$) of these 50 MLPs for $x \in [-2, 2]$.

(c) Repeat this procedure using instead $w_{i1}^{10} \sim \mathcal{N}(0, 0.5)$. What difference can you observe?

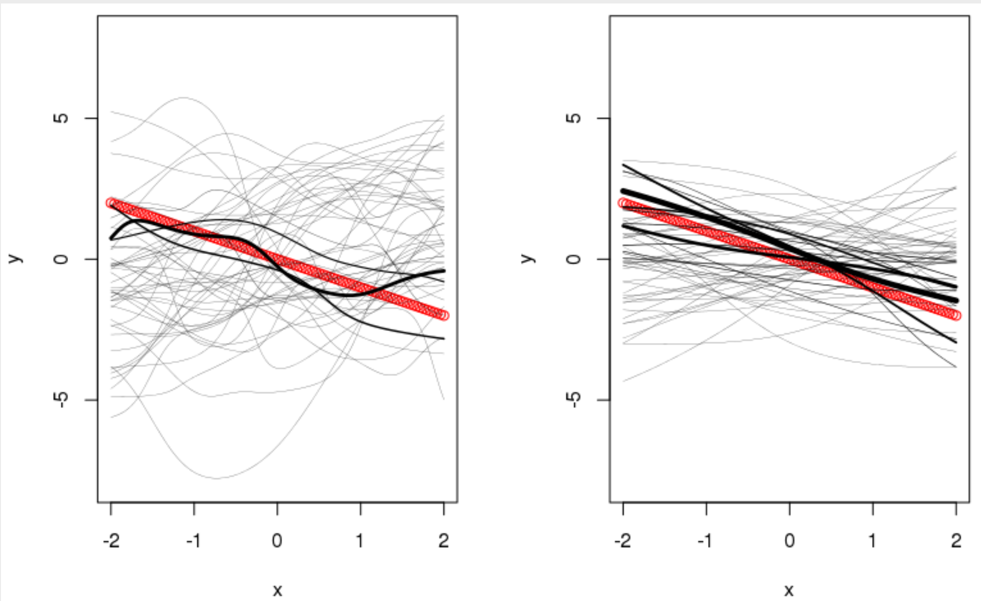
(d) Compute the mean squared error between each of these 2x50 (50 from each of the above two initialization procedures) input-output functions and the function $g(x) = -x$. For each of the two procedures, which MLP approximates g best? Plot $y(x)$ for these two MLPs

Solution

(b) is plotted on the left side and (c) on the right side. The difference is the smoothness of the drawn functions (smaller weights $\implies$ linear regime of sigmoidal function).



(d) The MLPs from (b) and (c) are plotted, each with a different line thickness. The thicker the line the less MSE is scored by that model on the training data (i.e. the thickest line has the least MSE, the faintest line has highest MSE). g is plotted in red. The thickest two lines are the two MLP that approximate g the best.



Total 10 points.