183.605
Machine Learning for Visual Computing
Assignment 1

Group 12:
Hanna Huber ()
Lena Trautmann (1526567)
Elisabeth Wetzer ()

November 17, 2016

- Upload a zip-file with the required programs. You can choose the programming language.
- Add a PDF document with answers to all of the questions of the assignment (particularly all required plots) and description and discussion of results.

1 Assignment 1

1.1 Part 1: Binary classification and the perceptron

1.1.1 Reading data

Tasks:

- Read the data using functions of your programming language resp. simulation software.

First, we wrote a new .csv-file with our .py-script to delete all extra blanks. Afterwards we could read in the data using `dlmread()`.

- Plot the input vectors in $\mathbb{R}^2$ and visualize corresponding target values (e.g. by using color).

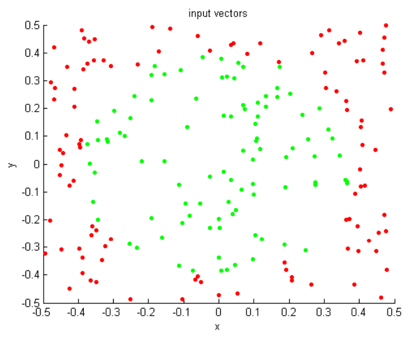


Figure 1: Plot of the input vectors with the target value visualized by colour.

Fig. 1.1.1 shows the input vectors.

- Use the feature transformation $(x_1, x_2) \to (x_1^2, x_2^2)$ and plot the data in the new feature space. The data should now be linearly separable.

Fig. 1.1.1 shows the transformed input vectors.

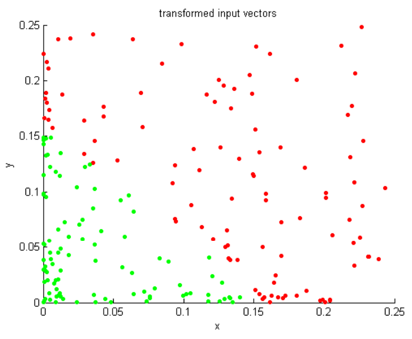


Figure 2: Plot of the transformed input vectors with the target value visualized by colour.

1.1.2 Perceptron training algorithm

The function

$$\texttt{y = perc(w,X).}$$

simulates a perceptron. The first argument is the weight vector $\mathbf{w}$ and the second argument is a matrix with input vectors in its columns $\mathbf{X}$. The output $\mathbf{y}$ is a binary vector with class labels 1 or -1.

The function

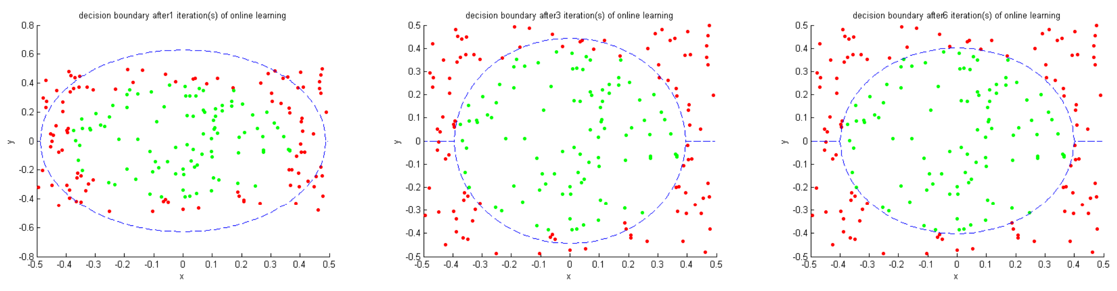
$$\texttt{w = percTrain(X,t,maxIts,online).}$$

returns a weight vector $\mathbf{w}$ corresponding to the decision boundary separating the input vectors in $\mathbf{X}$ according to their target values $\mathbf{t}$.

The argument `maxIts` determines an upper limit for iterations of the gradient based optimization procedure. If this upper limit is reached before a solution vector is found, the function returns the current $\mathbf{w}$, otherwise it returns the solution weight vector. `online` is *true* if the *online*-version of the optimization procedure is to be used or *false* for the *batch*-version.

Tasks:

- The functions `percTrain` and `perc` are implemented in the files `percTrain.m` and `perc.m`, respectively.
- Figures 3, 4, 5, 6 show perceptron learning after the first iteration, half of the iterations needed and after the final iteration. They show that the algorithm converges much faster using online learning (see Figures 3 and 4). Figures 4 and 6 illustrate how the transformation into the feature space of basis functions makes the data linearly separable. The non-linear decision boundary in the original data space is obtained by applying the inverse transformation to the linear decision boundary in feature space.



(a) (b) (c)

Figure 3: Perceptron decision boundary in the original data space at iterations #1, #3 and #6 of online learning.

- The weight vector is initialized as $\mathbf{w} = \mathbf{0}$. This way, the learning rate γ merely scales the weight vector $\mathbf{w}^{(j)} = \gamma \sum_{i \in M} x_i t_i$, where j denotes the current iteration and M the set of data points that have been misclassified and used to update **w** up to this point. As $(\gamma_1 \sum_{i \in M} x_i t_i)^T (x_k t_k) \leq 0 \Leftrightarrow (\gamma_2 \sum_{i \in M} x_i t_i)^T (x_k t_k) \leq 0$ for any $\gamma_1, \gamma_2 > 0$, the classification of the kth data point and thus the learning behaviour of the perceptron are not influenced by the learning rate.

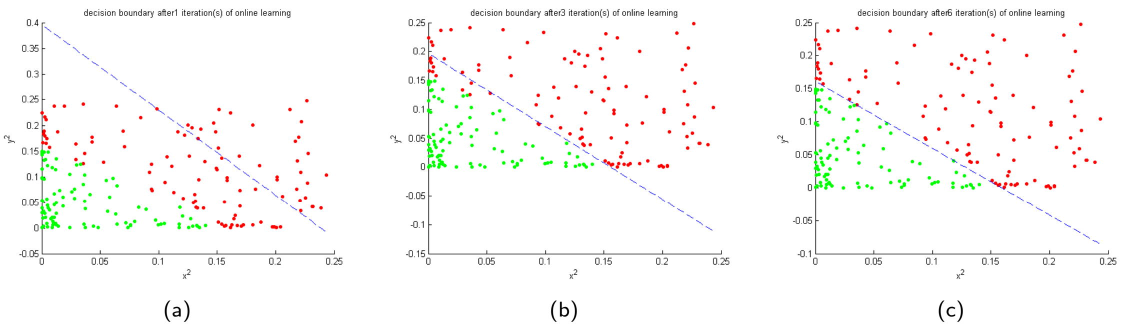


Figure 4: Perceptron decision boundary in the feature space of basis functions at iterations #1, #3 and #6 of online learning.

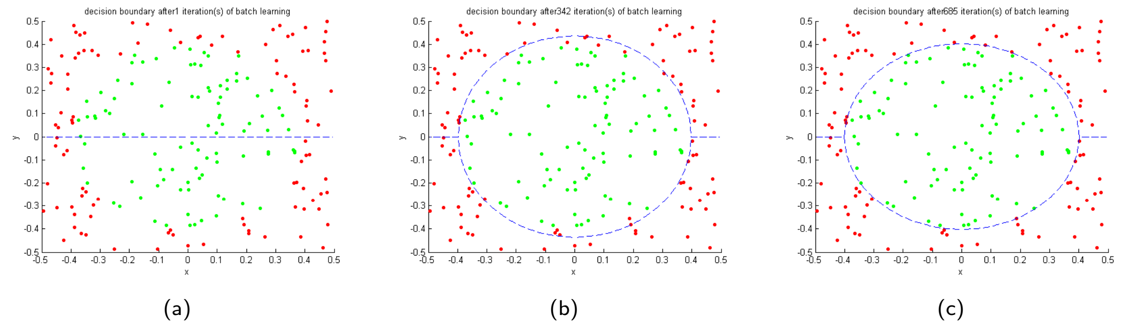


Figure 5: Perceptron decision boundary in the original data space at iterations #1, #342 and #685 of batch learning.

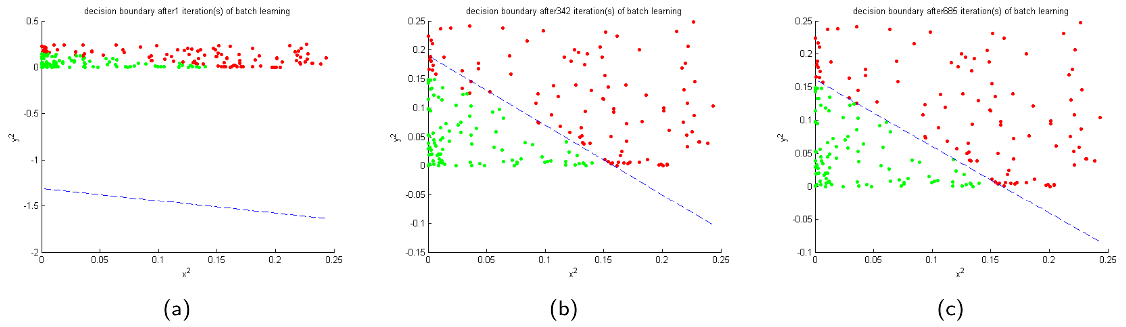


Figure 6: Perceptron decision boundary in the feature space of basis functions at iterations #1, #342 and #685 of batch learning.

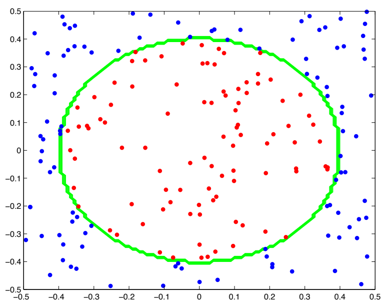


Figure 7: Plot of the decision boundary in the original data space found by the perceptron (green curve) together with labelled data points.

1.2 Part 2: Linear basis function models for regression

Aim of this exercise to deepen understanding of parameter optimization of error function while taking into account the relation of model complexity (in this case it corresponds to the number of basis functions used) and the expected error. Since this expected value (the mean squared error) is a theoretical quantity, it has to be estimated by the average of the error of predictions resulting from many training runs with different randomly generated target values.

1.2.1 Experimental setup

A row vector of scalar inputs $x \in [0, 5]$ sampling the interval in steps of 0.1 (resulting in 51 values) and a corresponding output vector $\mathbf{y}$ with values $y = f(x) = 2x^2 - Gx + 1$ is the basis of this experiment. The coefficient G is your group number. These 51 points are to be used for the visualization of the target function and the predictions of the fitted model. A training set is generated by subsampling the 51 values as follows: Every eighth value ($x_0 = 0$, $x_1 = 0.8$, $x_2 = 0.16$, ...) is assigned to the training set and the target values t_i are obtained by adding to the corresponding y_i a random value from the normal distribution $\mathcal{N}(\mu = 0, \sigma = 16)$. Thus, the training set contains $N = 9$ pairs of observations x_i, t_i.

We will employ a linear basis function model of the form $f_{\mathbf{w}}(x) = \mathbf{w}^T\mathbf{\Phi}(x)$, where

$$\mathbf{\Phi}(x) \to \begin{pmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^d \end{pmatrix}, \tag{1}$$

and $\mathbf{w} \in \mathbb{R}^{d+1}$. The model will be fitted to the training set by minimization of the training error

$$E(\mathbf{w}) = \sum_{i=1}^{N} (t_i - \mathbf{w}^T\mathbf{\Phi}(x_i))^2 \tag{2}$$

also known as the *residual sum of squares* (RSS). The optimal weight vector is given by $\mathbf{w}^* = \arg\min_{\mathbf{w}} E(\mathbf{w})$.

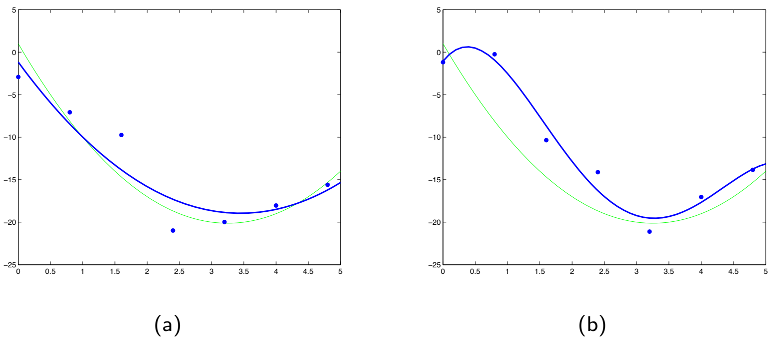


Figure 8: An example of a true target function (thin green curve) from which the training data was generated, training set (without feature transformation) with $N = 9$ (blue dots) and prediction of the fitted model $\mathbf{w}^T\mathbf{\Phi}(x)$ (blue curve). The basis functions are $\mathbf{\Phi}(x) \to (1, x, x^2, x^3, ..., x^d)^T$. (a) $d = 2$ (b) $d = 4$.

1.2.2 Optimization: LMS-learning rule vs. closed form

Use a linear unit (*online* LMS-learning rule) for regression on transformed input data. In a first step use a linear basis function model with $d = 2$ (in Matlab you can calculate the power elementwise: e.g. `[x x x].∧ [0 1 2]`). Hint: Visualize y and its prediction during the training or observe the chance of the weight vector to determine useful values for γ.

Tasks:

- What is the resulting weight vector when using the LMS-rule?
- How can you determine the optimal $\mathbf{w}^*$ in closed form? Compare $\mathbf{w}^*$ with the outcome of the LMS-rule training.
- What is the influence of γ? Which value for γ represents a good tradeoff between number of iterations and convergence?

1.2.3 Model-complexity and model-selection

Determine $\mathbf{w}^*$ in closed form for 2000 different training sets, in which only the t_i are varyied according to $\mathcal{N}(\mu = y_i, \sigma = 16)$, while the x_i remain unchanged.

Tasks[1]:

- Select a fixed x', which is not an observation of the training set, but lies between two observations (e.g. $x' = 2$)
- Estimate the *mean squared error*

  $$\textit{mse} = \mathcal{E}(f(x') - f_{\mathbf{w}^*}(x'))^2, \tag{3}$$

  i.e., the mean of the squared residuals of the models prediction $f_{\mathbf{w}^*}(x')$ from the true function value $f(x')$ for all $0 \leq d \leq 8$ ($d = 0$ corresponds to a constant function) using at least 2000 trials.
- Estimate by the same way the quantities *bias*$^2 = (f(x') - \mathcal{E} f_{\mathbf{w}^*}(x'))^2$ and *var*$= (f_{\mathbf{w}^*}(x') - \mathcal{E} f_{\mathbf{w}^*}(x'))^2$.
- Plot *mse*, *bias*^{2} and *var* against d together in one plot. What is the relation of the quantities?
- (optional) Generate the above plots only for $d = 8$, but minimize instead of $E(\mathbf{w})$ the regularized error function

  $$E_\lambda(\mathbf{w}) = \sum_{i=1}^{N} (t_i - \mathbf{w}^T \mathbf{\Phi}(x_i))^2 + \lambda \|\mathbf{w}\|^2, \tag{4}$$

  i.e. $\mathbf{w}^* = \arg\min^*_{\mathbf{w}} E_\lambda(\mathbf{w})$. Plot the quantities against λ instead of d. Hint: The minimum can be obtained in closed form (see lecture slides).

[1] In all tasks $\mathcal{E}$ refers to the expected value with respect to the random variable $\mathbf{w}^*$, i.e. $\mathcal{E} \equiv \mathcal{E}_{\mathbf{w}^*}$