183.605
Machine Learning for Visual Computing
Assignment 1

Group 12:
Hanna Huber (0925230)
Lena Trautmann (1526567)
Elisabeth Wetzer (0726681)

November 22, 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).

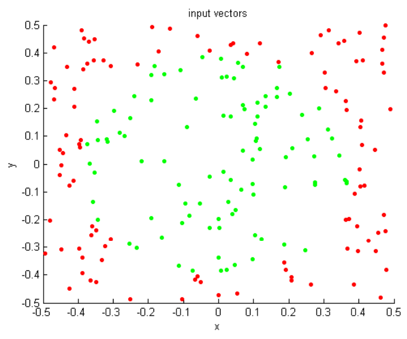


Figure 1: Plot of the input vectors with the target value visualized by colour.

Figure 1 shows the input vectors.

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

Figure 2 shows the transformed input vectors.

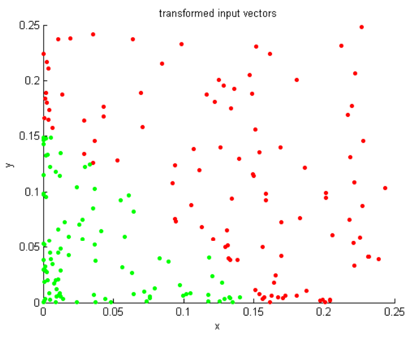


Figure 2: Plot of the transformed input vectors with the target value visualized by colour.

1.1.2 Perceptron training algorithm

Tasks:

- Implement both functions. Use homogeneous coordinates and the corresponding augmented weight vector $\mathbf{w} \in \mathbb{R}^3$.

The functions `percTrain` and `perc` are implemented in the files `percTrain.m` and `perc.m`, respectively.

- Plot the data and decision boundary in $\mathbb{R}^2$, both in the original data space (see e.g. Figure 3) and in the feature space of basis functions, each at three different stages of the training: after the first iteration, after approximately half of the required iterations, and after convergence.

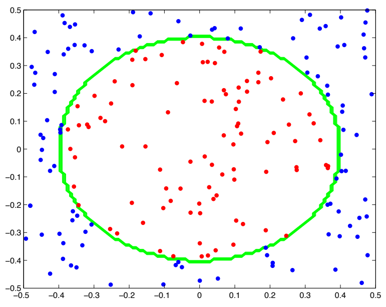
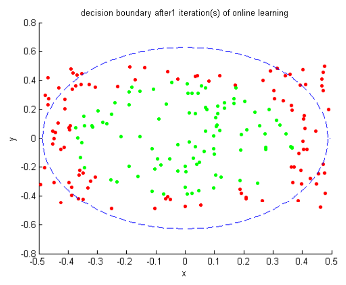


Figure 3: Plot of the decision boundary in the original data space found by the perceptron (green curve) together with labelled data points.

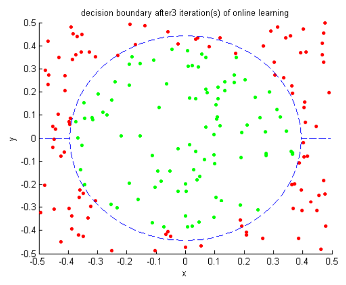
Figures 4, 5, 6, 7 show perceptron learning after the first iteration, half of the iterations needed and after the final iteration. They show that the algorithm converges much faster using online learning (see Figures 4 and 5). Figures 5 and 7 illustrate how the transformation into the feature space 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.

- Initialize $\mathbf{w} = \mathbf{0}$. What is the influence of the learning rate?

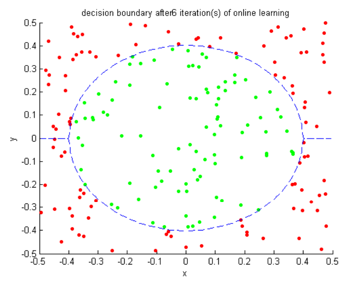
The weight vector is initialized as $\mathbf{w} = \mathbf{0}$. This way, the learning rate γ merely scales the weight vector $\mathbf{w}^{(\mathbf{j})} = \gamma \sum_{i \in M} x_i t_i$, where j denotes the current iteration and M the set of data points that have been misclassified and used to update **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.



(a)

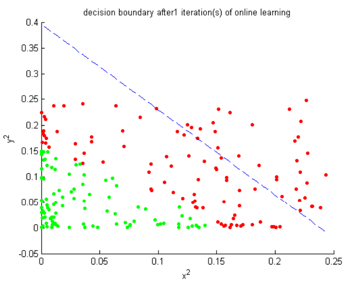


(b)

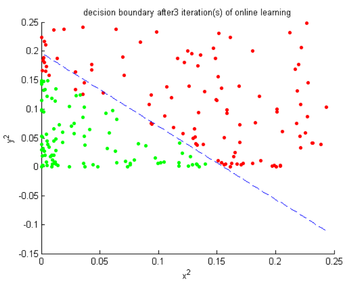


(c)

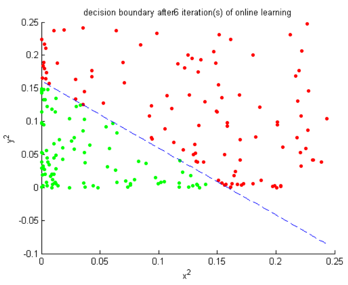
Figure 4: Perceptron decision boundary in the original data space at iterations #1, #3 and #6 of online learning.



(a)

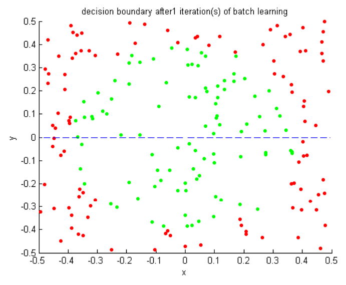


(b)

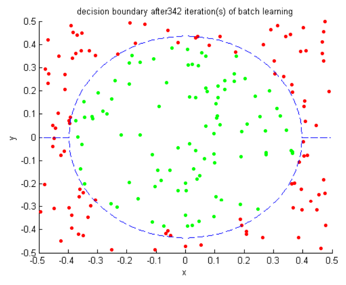


(c)

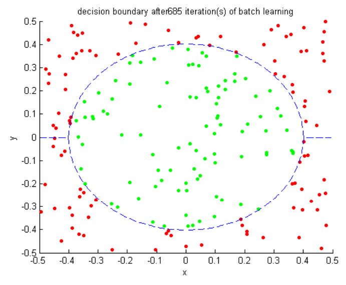
Figure 5: Perceptron decision boundary in the feature space of basis functions at iterations #1, #3 and #6 of online learning.



(a)

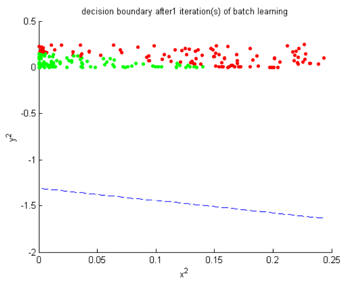


(b)

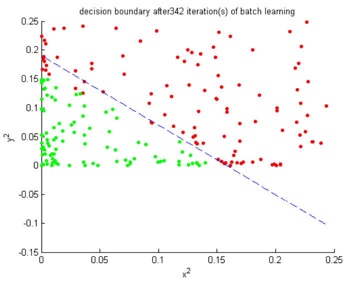


(c)

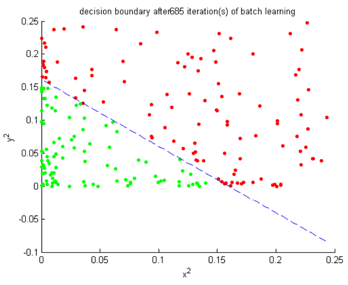
Figure 6: Perceptron decision boundary in the original data space at iterations #1, #342 and #685 of batch learning.



(a)



(b)



(c)

Figure 7: Perceptron decision boundary in the feature space of basis functions at iterations #1, #342 and #685 of batch learning.

1.2 Part 2: Linear basis function models for regression

1.2.1 Experimental setup

At the end of the setup we have:

Table 1: Weight vector derived for the LMS-rule (online and closed form)

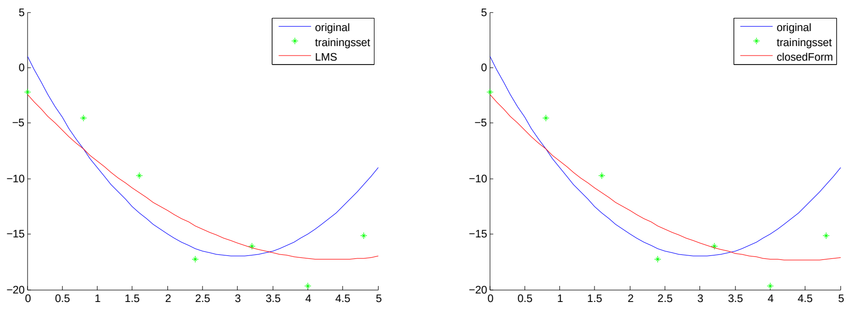
	ϕ_0	ϕ_1	ϕ_2
wOnline	-2.4164	-6.7862	0.7742
wClosed	-2.4193	-6.7593	0.7645

- xtrain and ttrain: the training data
- phi: the transformation function
- xtrain_phi: the transformation of xtrain

1.2.2 Optimization: LMS-learning rule vs. closed form

Tasks:

- What is the resulting weight vector when using the LMS-rule?



(a) (b)

Figure 8: Optimization using the LMS-rule with online learning (a) and in closed form (b)

The values of the resulting weight vector is summarized in Table 1 and the corresponding curve plotted in Figure 8(a).

- How can you determine the optimal $\mathbf{w}^*$ in closed form? Compare $\mathbf{w}^*$ with the outcome of the LMS-rule training.

The closed form is calculated by updating the weight vector, determined by all N points, in one step. The resulting values are also summarized in Table 1 and the corresponding curve plotted in Figure 8(b).

- What is the influence of γ? Which value for γ represents a good tradeoff between number of iterations and convergence?

The learning rate γ controls the influence of the single data points. Using high learning rates during online learning, the regression reacts quickly and moves towards the current data point. This is disadvantageous if the current data point is an outlier. Consequently, if the learning rate is too high, the number of iterations until convergence increases as well. The same problem occurs, if the rate is too low and the regression reacts too slowly. For this assignment, we observed that a learning rate of $\gamma = 0.0001$ is a suitable choice.
Also, the difference between online and batch learning decreases with decreasing values of γ. This is due to the fact that the weight vector $\mathbf{w}$ only changes very little with each data point during online learning. As a result, the update factor for subsequent datapoints is also only slightly changed. This way, the difference between adding the update factors subseuqently or at the same time is less remarkable.

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 $\mathcal{N}(\mu = y_i, \sigma = 4)$, while the x_i remain unchanged.
Assuming that a deterministic function $y(\mathbf{x}, \mathbf{w})$ specifies the target variable t. Varying the only the t_i according

to $\mathcal{N}(\mu = y_i, \sigma = 4)$, we get $t = y(\mathbf{x}, \mathbf{w}) + \epsilon$, where ϵ is a random variable $\sim \mathcal{N}(\mu = 0, \sigma = 16)$.
We get

$$p(t|\mathbf{x}, \mathbf{w}, \frac{1}{16}) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), 4)$$

The sum of squares error function is definied as in [1]:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (t_n - \mathbf{w}^T \phi(\mathbf{x_n}))^2$$

To minimize the error function, the gradient is set to zero:

$$\sum_{n=1}^{N} t_n \phi(\mathbf{x}_n)^T - \mathbf{w}^T \left(\sum_{n=1}^{N} \phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^T \right) = 0$$

Hence the solution for $\mathbf{w}$ is obtained by:

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t}$$

where the entries of $\mathbf{X} \in \mathbb{R}^{N \times M}$ are given by $\phi_j(\mathbf{x}_n)$.
$(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$ is known to be the Moore-Penrose-inverse of $\mathbf{X}$ and can be calculated in Matlab by `pinv(X)`.

Tasks[1]:

- Select a fixed x', which is not an observation of the training set, but lies between two observations. In the following trials $x' = 2$ was chosen.
- Estimate the *mean squared error*

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

  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*$= \mathcal{E}(f_{\mathbf{w}^*}(x') - \mathcal{E} f_{\mathbf{w}^*}(x'))^2$.
- 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

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

  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).

The expected value of the mean squared error $\mathcal{E}(f(x') - f_{\mathbf{w}^*}(x'))^2$ for two thousand trials evaluated at $x' = 2$ is plotted against d, the number of dimensions of the linear basis function space in blue in figure 9. The same plot shows the bias $(f(x') - \mathcal{E} f_{\mathbf{w}^*}(x'))^2$ and the variance $\mathcal{E}(f_{\mathbf{w}^*}(x') - \mathcal{E} f_{\mathbf{w}^*}(x'))^2$ in red and yellow respectively. As described in [2] the expected squared error on a sample x can be decomposed by the bias and variance as follows:

$$\mathcal{E}(f(x') - f_{\mathbf{w}^*}(x'))^2 = (f(x') - \mathcal{E} f_{\mathbf{w}^*}(x'))^2 + \mathcal{E}(f_{\mathbf{w}^*}(x') - \mathcal{E} f_{\mathbf{w}^*}(x'))^2 + \epsilon$$

where ϵ is an irreducible error.

This relationship is seen in figure 9. In higher dimensions than $d = 2$ the bias becomes zero and the expected mean squared error is equal to the variance estimate. Furthermore table 3 lists the model qualities in dependence of d. The last column shows that the mean squared error can be decomposed into the given bias and variance up to an accuracy of 10^{-13}, which relates to the irreducible error ϵ. The plots in figure 9 are conclusive with the literature: dimensionality reduction can decrease variance, but simplicity of the model introduces more bias, while adding features to the model decreases the bias, but in return increases the variance. The mean squared error shows a minimum at $d = 2$. The spike in the plot can be explained by

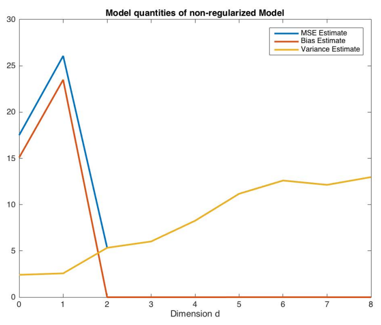


Figure 9: The expected value of the mean squared error, the bias and the variance against d for a fixed $x' = 2$ evaluated for 2000 trials, where d denotes the dimension of the basis functions model

Table 2: Estimated MSE, Variance and Bias of non-regularized Model

d	MSE	Bias	Variance	MSE-(Bias+Variance)
0	17.51	15.09	2.42	1.0e-13 *-0.96
1	26.07	23.49	2.57	1.0e-13 * 0.60
2	5.34	1.10	5.34	1.0e-13 * 0.07
3	6.02	6.63	6.02	1.0e-13 * 0.10
4	8.26	0.00	8.26	1.0e-13 * 0.23
5	11.18	0.00	11.17	1.0e-13 *-0.05
6	12.60	0.00	12.60	1.0e-13 * 0.16
7	12.14	0.00	12.14	1.0e-13 *-0.28
8	12.99	0.00	12.98	1.0e-13 *-0.21

the fact that the quantities were calculated for one explicit x and would average out by taking the average of multiple evaluation points of the trials.

In order to control overfitting a regularization term is often added to the error function which shall be minimized to find an optimal solution. Instead of minimizing $E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(t_n - \mathbf{w}^T\phi(\mathbf{x_n}))^2$, the modified error function $\tilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(t_n - \mathbf{w}^T\phi(\mathbf{x_n}))^2 + \frac{1}{2}\lambda\mathbf{w}^T\mathbf{w}$ is to be minimized. There are other options to choose a regularizer than $\lambda||\mathbf{w}||$ but it is a simple choice to demonstrate the trade off between bias and variance. This specific regularizer is called weight decay and does not effect the circumstance of the error function being quadratic and hence convex. Setting the gradient of the error function in respect to $\mathbf{w}$ to zero gives:

$$\mathbf{w} = (\lambda\mathbf{I} + \mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T t$$

where $\mathbf{I}$ denotes the identity matrix. As before the expected mean squared error, the bias and the variance are plotted, but only for dimension $d = 8$, which introduced a large variance in the non-regularized model. Figure 10 shows the quantities against λ, where λ is chosen to be $\exp(i-4)$, where i is the index on the x-axis in the figure. For small λ the variance dominates the mean squared error, because the model complexity is high. For greater λ, hence greater regularization, the variance is restrained, but in exchange to an increase in bias. The results of the tests are shown in table 3. The decomposition of the mean squared error into the bias and variance terms leaves an error as small as 10^{-12}.

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

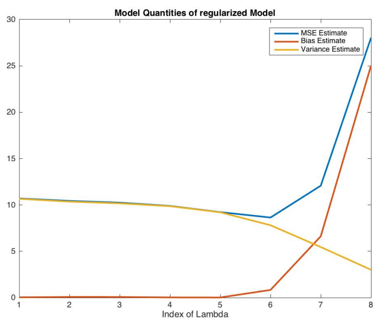


Figure 10: The expected value of the mean squared error, the bias and the variance against the index of λ for a fixed $x' = 2$ evaluated for 2000 trials, where λ denotes the weight decay parameter and is considered $\exp(\text{index} - 4)$

Table 3: Estimated MSE, Variance and Bias of regularized Model

λ	MSE	Bias	Variance	MSE-(Bias+Variance)
$\exp(-3)$	10.70	0.04	10.66	1.0e-12 *-0.01
$\exp(-2)$	10.44	0.08	10.36	1.0e-12 *-0.04
$\exp(-1)$	10.25	0.07	10.17	1.0e-12 * 0.02
1	9.90	0.03	9.87	1.0e-12 * 0.01
$\exp(1)$	9.23	0.02	9.21	1.0e-12 * 0.01
$\exp(2)$	8.64	0.83	7.81	1.0e-12 * 0.01
$\exp(3)$	12.08	6.62	5.46	1.0e-12 *-0.15
$\exp(4)$	28.04	25.03	3.01	1.0e-12 *-0.16

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Friedman J., Hastie T., and Tibshirani R. *The Elements of Statistical Learning*. Springer Series in Statistics, 2009.