Assignment 3

Computational Intelligence, SS2020

Team Members		
Last name	First name	Matriculation Number
Blöcher	Christian	01573246
Bürgener	Max	01531577

1 Regression with Neural Networks

1.1 Simple Regression with Neural Networks

1.1.1 Learned Function

1.1.2 Variability of the performance of deep neural networks

1.1.3 Varying the number of hidden neurons

1.1.4 Change of MSE during the course of training

1.2 Regularized Neural Networks: Weight Decay

2 Classification with Neural Networks: Fashion MNIST

- The confusion matrix

	T-Shirt/top	Trousers	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
T-Shirt/top	841	3	12	25	4	1	108	0	6	0
Trousers	7	960	2	21	5	0	4	0	1	0
Pullover	19	0	835	15	52	0	78	0	1	0
Dress	21	4	14	904	23	0	31	0	3	0
Coat	2	1	104	30	797	0	65	0	1	0
Sandal	1	0	0	1	0	955	0	23	2	18
Shirt	104	1	75	26	45	1	741	0	7	0
Sneaker	0	0	0	0	0	15	0	958	0	27
Bag	5	0	5	6	2	3	4	4	970	1
Ankle boot	1	0	0	0	0	9	0	3	0	960

Table 1: Confusion matrix for Fashion MNIST Classification

The rows of our confusion matrix correspond to the true classes and the columns to the predicted classes. The global accuracy for all classes is fairly good with $89,2\%$. Apparently it was difficult for the code to distinguish t-shirts/tops from shirts and coats from pullovers. Also remarkable is that all top clothes were often mistakenly recognized as shirts.

- Are particular regions of the images get more weights than others?

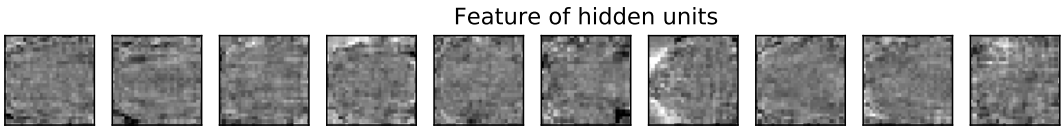


Figure 1: Weights per pixel for 10 samples

In figure 2 are the weights per features visualized. As for the features these are referring to one pixel of a 28 x 28 pixel conversion from each picture. The weights are scaled from important regions (white) to regions with less information (black). It makes sense and can be seen in the pictures that that the object edges are important since all picture have the black background in common and the object outlines vary a lot.

- Boxplot of classifications with different inital weight vector values

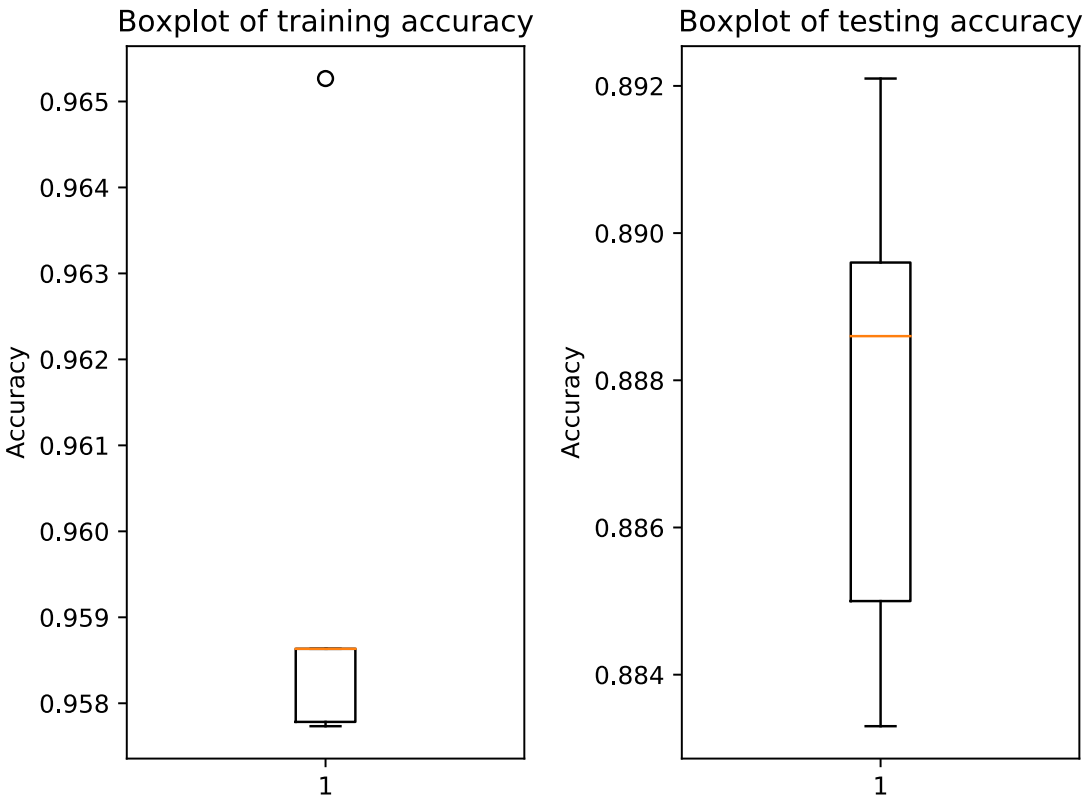
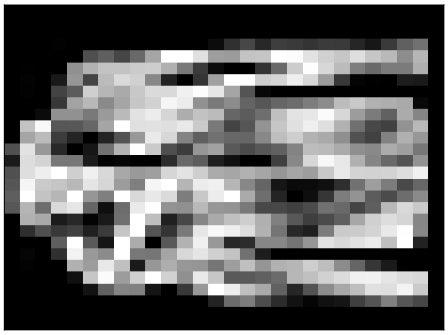


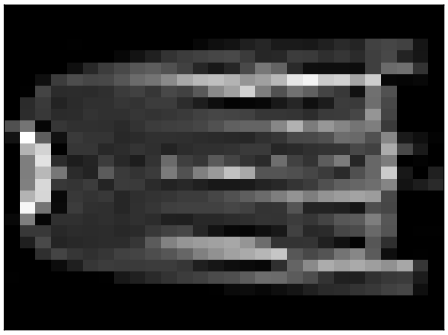
Figure 2: Accuracy of classifications using different inital **w**

After choosing the optimal amount of hidden layers the boxplots visualize the accuracy of classifications which started with different weight vectors **w**. The orange coloured line shows the median of our calculated numbers. The lower border of the box is enclosing the median of all numbers which are lower than our global median. The upper border is enclosing all numbers till the median of all higher numbers. The last section is showing the global minimum and maximum. Any outliers are marked with a spot.

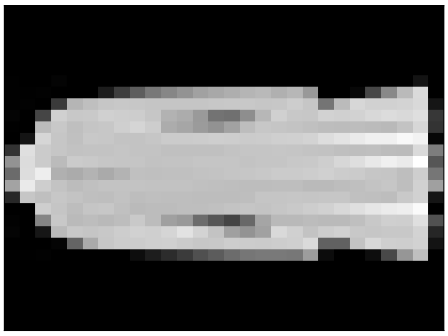
- Missclassified items



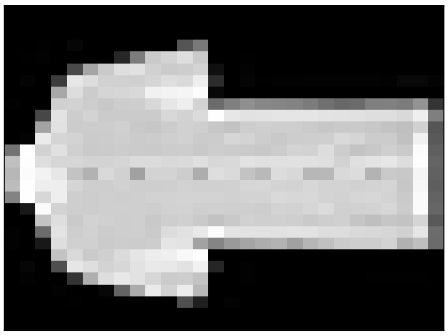
(a) Missclassified image 1



(b) Missclassified image 2



(c) Missclassified image 3



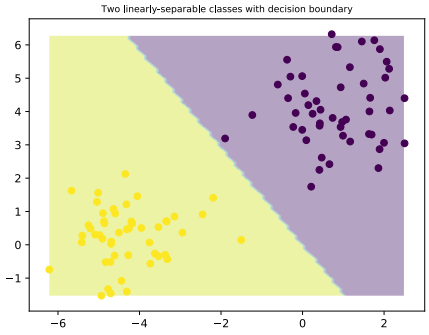
(d) Missclassified image 4

Figure 3: Missclassified items from our test set

3 Bonus: Implementation of a Perceptron

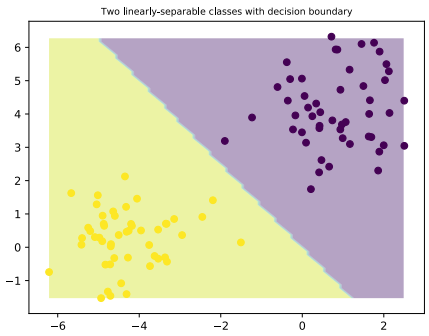
- Linear separable dataset:

(a) self implementation

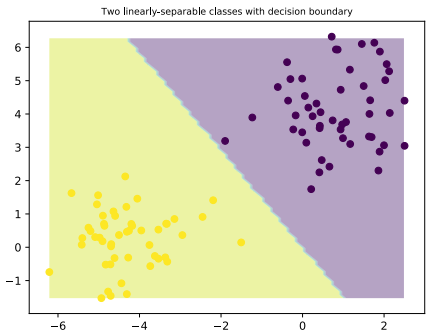


(b) scikit-learn's implementation

Figure 4: Classification using eta = $0,01$ and max. iterations of 2



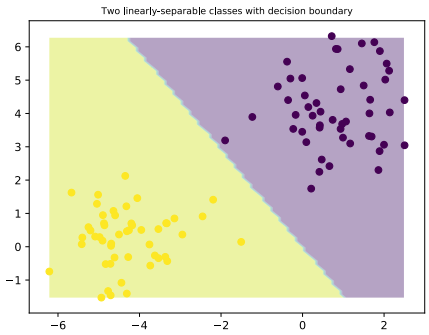
(a) self implementation



(b) scikit-learn's implementation

Figure 5: Classification using eta = $0,01$ and max. iterations of 5

(a) self implementation



(b) scikit-learn's implementation

Figure 6: Classification using eta = $0,001$ and max. iterations of 5

The classification of the dataset was similar for both implementations. Only using small eta's $< 0,001$ our code began to misclassify the whole dataset.

- How many training iterations does it take for the perceptron to learn to classify all training samples perfectly?

  Our own implementation needs in average 2 iterations to classify the linear dataset perfectly. This make sense since we're using random initial weights for just two different states.

- The misclassification rate for both datasets

Training MSE:	0,475
Test MSE:	0,5

Training MSE:	0,475
Test MSE:	0,5

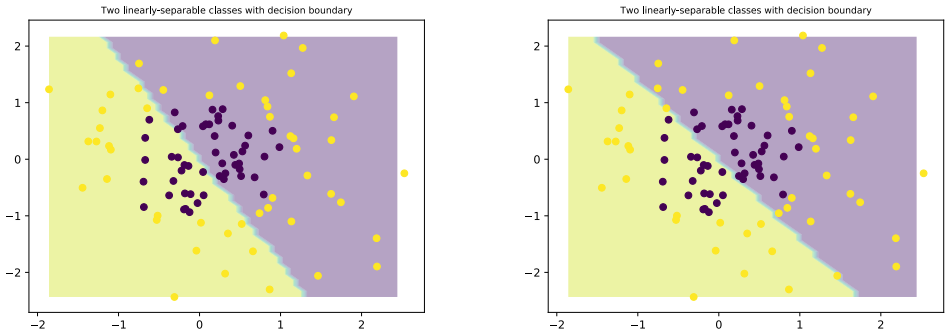
Table 2: MSEs using eta $= 0,01$ and 5 iterations self impl.(l.) skl.impl.(r.)

Training MSE:	0,525
Test MSE:	0,55

Training MSE:	0,475
Test MSE:	0,5

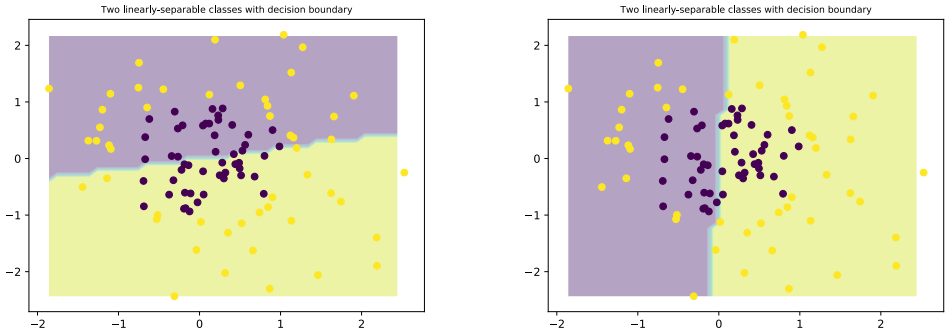
Table 3: MSEs using eta $= 0,1$ and 100 iterations self impl.(l.) skl.impl.(r.)

- Non linear seperable dataset:



(a) self implementation (b) scikit-learn's implementation

Figure 7: Classification using eta $= 0,01$ and max. iterations of 5



(a) self implementation (b) scikit-learn's implementation

Figure 8: Classification using eta $= 0,1$ and max. iterations of 100

Both implementations can not classify the non linear dataset.

- How would you learn a non-linear decision boundary using a single perceptron?

  We should use a non linear activation function i.e. the sigmoid function with logistic regression as we used it in our homework 2. Using a suiting degree and learning rate the classification could be much better.