183.605
Machine Learning for Visual Computing
Assignment 1

Michael Reiter

16. Oktober 2018

1 Task1

1.1 MNIST Data Set and Feature Selection

We applied a perceptron model to classification of handwritten digits. MNIST is a standard data set with 28×28 pixel images of handwritten digits. We selected the digits 0 and 7 and 500 training images for each class for our classification task, as they are reasonably diffenrent. To chose the right feature we plotted all regionprops in a scatter plot matrix,see figure 1 and selected solodity and eccentricity.

1.1.1 Features without transform

We used once the batch training algorithm and once the online training algorithm to classify the data directly from these to features. The connfusion matrix evaluated on a separate test set gives us following perfomance:

Tabelle1

Tabelle2

The corresponding decision boundries are shown in figure ... . We see that the data is not linearly seperable

1.1.2 Classification with feature transform

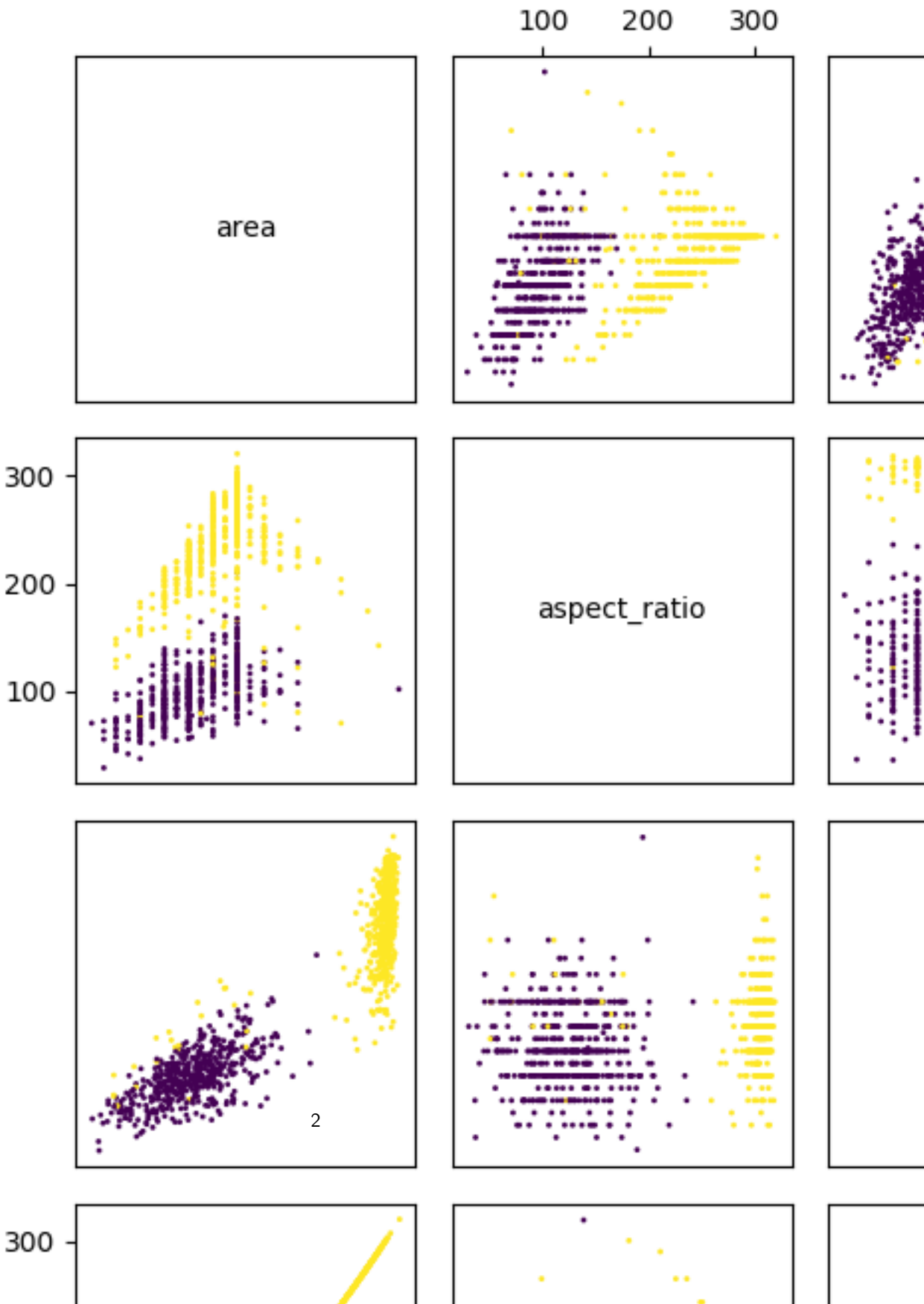
1.1.3 Classification directly with image data

2 Task2

2.1 LMS vs. closed Form

We followed the instructions and used the function below with $G = 6$ to determine our y values for the experimental setup.

$$f(x) = 2x^2 - Gx + 1 \tag{1}$$


100
200
300
area
300
200
100
aspect_ratio
300
2

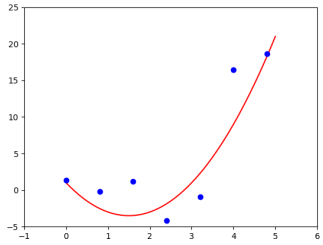


Figure 2: Setup and an example of t_i values with additional noise

2.1.1 What is the resulting weight Vector when using the LMS rule?

We implemented an online LMS learning rule for regression to determine useful weights to fit the curve. Figure **??** shows three runs with randomised training data. As we can see the fitted curve is moved towards the original curve with each iteration. γ or the learning factor controls how much impact new values have on the resulting weights for the next iteration.

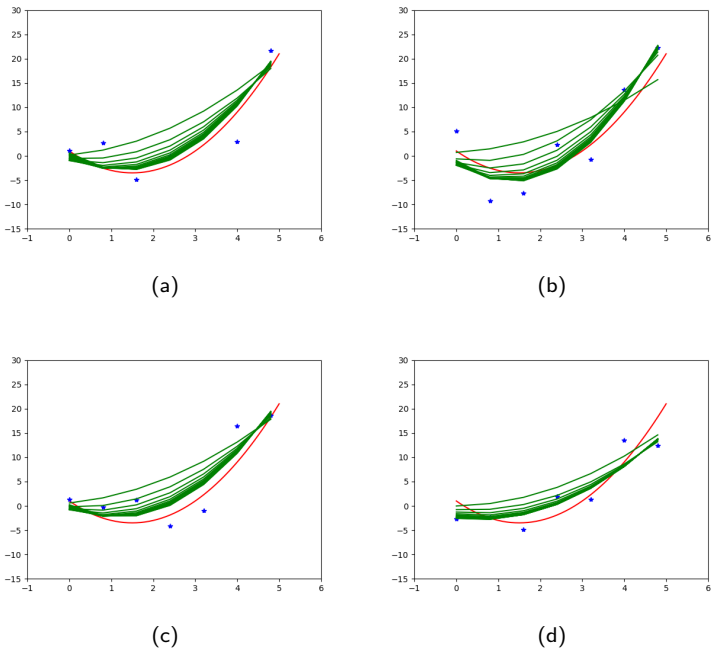
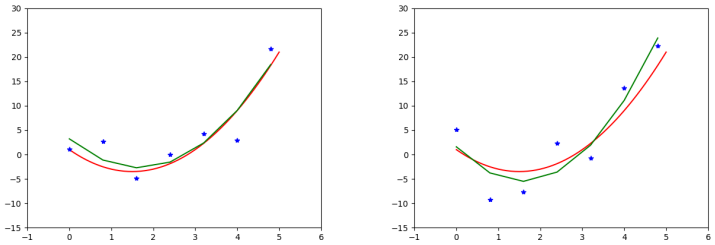


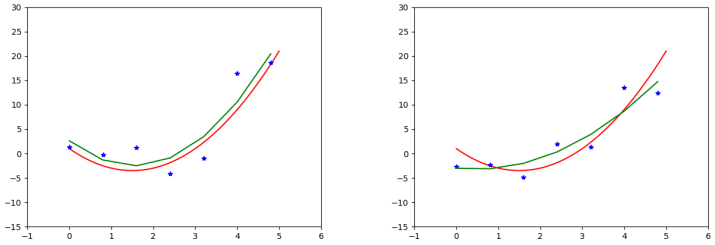
Figure 3: Visualization of the learning process of **LMS**

2.1.2 How can you determine the optimal $w*$ in closed form? Compare $w*$ with the outcome of the LMS-rule training

We used the pseudo inverse approach to minify the sum of squares error. As displayed the in the Tables, 1 and 2 we can see the resulting weights and the calculated errors. We can see that the error rate for our four examples with CF are all better without much surprise.



(a) (b)



(c) (d)

Figure 4: the same training data as in figure, however this time solved in closed form (**CF**)

Figure	**LMS**	**CF**
a	97.09004286828198	71.03243449830224
b	112.43543529868002	97.82006516766448
c	109.4839763954856	84.46023189949588
d	61.164485036889054	47.10797373501289

Table 1: error comparisons betweenLMS and CF

Figure	LMS	CF
a	2.79641366 -4.06370085 1.6057828	6.10503808 -6.93083294 2.04038233
b	0.90857717 -6.42409915 2.36363687	1.57854501 -8.9662967 2.8358482
c	0.25694124 -4.13859362 1.69469808	2.59983553 -6.64808401 2.1588638
d	-2.58665078 -0.97237072 0.9121616	-3.01226058 -0.92406774 0.96261322

Table 2: resulting weightsLMS and CF

2.1.3 Influence of γ

As mentioned earlier γ has to influence if the algorithm can converge. In case $gamma$ is too big, new values get too much weight, thus 'overriding' older ones. See Figure 5

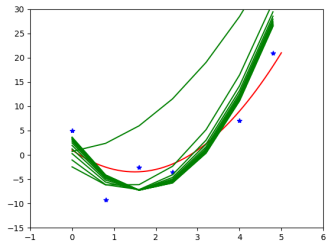


Figure 5: γ has been chosen too high

2.2 Image Data