Aufgabe 1
Machine Learning for Visual Computing

Philipp Omenitsch, Thomas Pinetz and Andreas Mair

January 20, 2016

Abstract

Abgabe 2 für Machine Learning for Visual Computing über die Themen Linear Regression und Autoencoder.

Contents

1 Linear Regression

Implementing linear regression was the first part of the assignment.

1.1 Introduction

This assignment was seperated into three parts.

1. Data Generation
2. Implementation of the algorithm
3. Evaluation of the result

1.2 Data Generation

For the linear regression assignment we had specific description about how to generate the dataset. The dataset consists of 51 points on the quadratic function $2*x^2+G*x+1$ where G is the number of our group, which is 5. For this dataset we also had to generate a trainingdataset, which is 1/6 in size of the original dataset. Those points are taken from the original dataset and then shifted by a random number chosen by a normal distribution with parameter $\mu = 0$ and $\sigma = 0.6$.

1.3 Implementation of the algorithm

In this assignment we used the online learning rule of the gradient descent algorithm. Therefore, every trainingsexample changes the weights. In contrast by using the batch learning technique, we only update after the whole trainingset has been processed. Therefore the weight changes faster with the online learning rule than with the batch learning.

The algorithm works the following way:

1. Calculate δ for the current training example
2. Update weights according to δ and γ.

We are using Sum of Squared Error (SSE) as our cost function and therefore *delta* can be calculated using $delta = -(y - x't) * x$. Then the weight update is $w(t+1) = w(t) - \gamma * \delta)$.

1.4 Evaluation of the Results

I tested the performance of the gradient descent algorithm on the dataset described in Section 1.2. The result can be seen in Figure 2. Additionally I compared it to the result obtained by calculating the optimal weight in Figure 1. As can be seen the weights are nearly identically after converging.

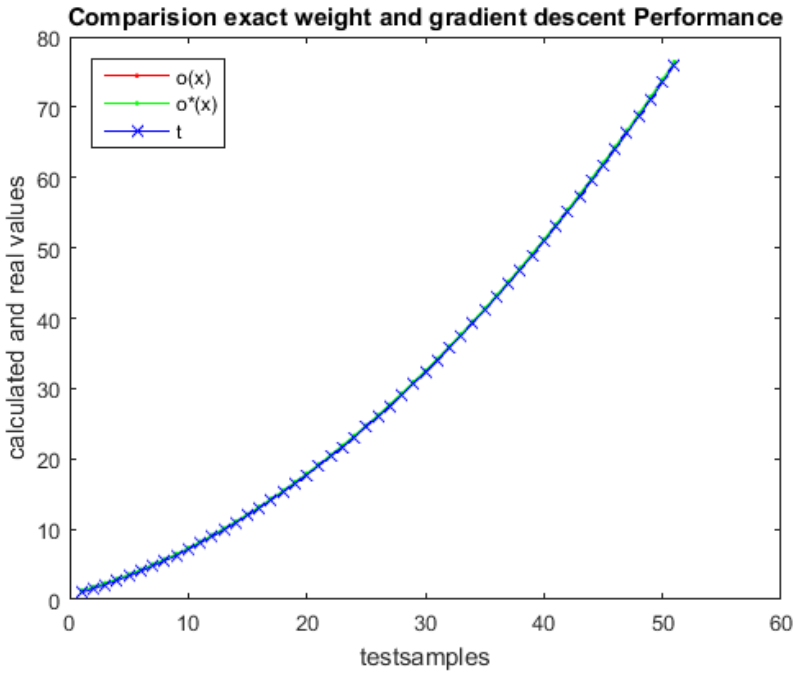


Figure 1: Comparision by gradient descent to the optimal weight

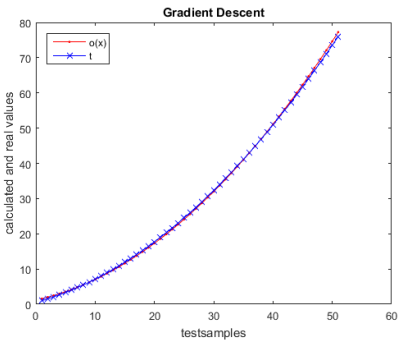


Figure 2: Performance by gradient descent on the real dataset

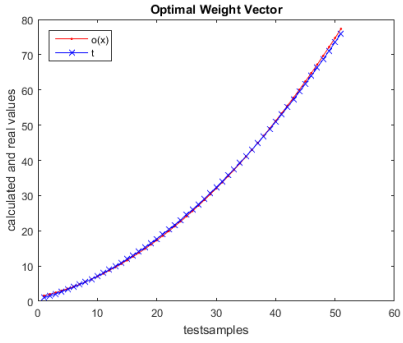


Figure 3: Performance by the optimal weight on the real dataset

1.5 Questions

1.5.1 Question 1: What weight vector was calculated using gradient descent?

We got the weight vector $w = [0.4154, 5.2012, 2.0025]$ for the sample problem after 1000 iterations of gradient descent with the learning rate $\gamma = 0.001$.

1.5.2 Question 2: How can you calculate the optimal weight directly? How much different is it to the result of gradient descent?

We can use linear algebra to reformulate the gradient descent problem. After solving the formula we get $w = (XX^T)^{-1}Xt^T$. Therefore, if XX^T is invertible we can use this formula to directly compute the optimal weight vector.

Our test data is randomly generated and the variance to the real data is also at random, therefore the weight vector had small differences in distances to each other. With this procedure we got t the optimal weight vector $w^* = [0.4040, 5.2352, 1.9910]$, whereby the gradient descent weight vector was $w = [0.4154, 5.2012, 2.0025]$. As can be seen in Figure 2 and Figure 3 the result is very similar. This result was obtained with the same testdata, we had used for question 1.

1.5.3 Question 3: Which tradeoff exist in choosing γ between trainingtime and convergence? Is there a γ where gradient descent diverges?

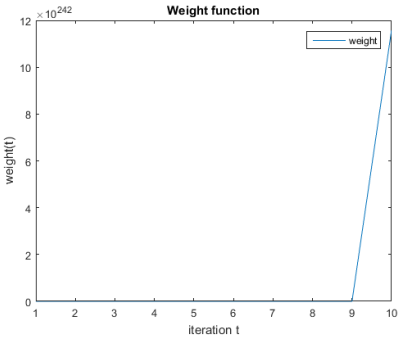


Figure 4: Convergence with γ at 1

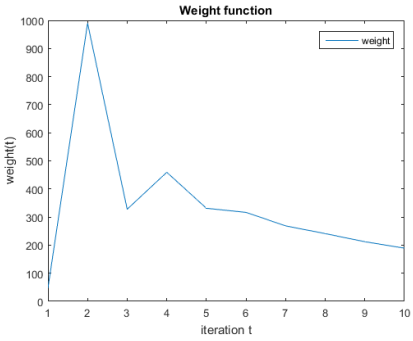
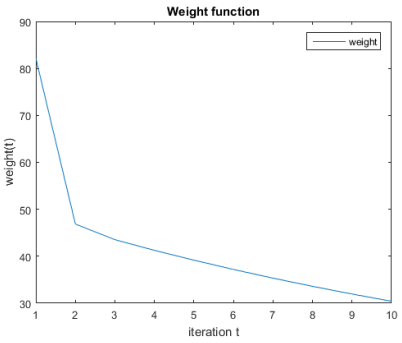


Figure 5: Convergence with γ at 0.005

As can be seen in Figure 4, 4, 5 and 7 chosing γ is a tradeoff between trainingtime and convergence. Also convergence depends on the random initialization of the training data. However there is a way to make the learning rate smaller in time by setting $\gamma = c/t$, where c is the initial learning rate and t is the current Iteration. For the given training data $\gamma = 1$ diverges as can be seen in Figure 4.



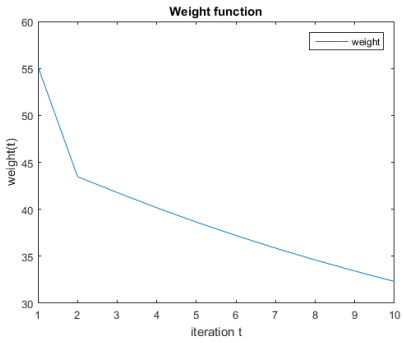


Figure 6: Convergence with γ at 0.005 Figure 7: Convergence with γ at 0.001