AARHUS UNIVERSITET, SCIENCE, COMPUTER SCIENCE

Machine Learning - Handin 1

Peter Burgaard - 201209175
Marie Louisa T. Berthelsen - 201303610
Nanna Engell Rønde Andersen - 201205671

September 26, 2016

INTRODUCTION

In this report we will discuss our implementation of a logistic regression algorithm, which is used for optical character recognition of handwritten numbers 0 through 9. We will outline our implementation choices, and briefly discuss some reflective questions about the algorithm.

IMPLEMENTATION OF THE ALGORITHM

In your report, you should shortly explain your algorithms and the choices you have made. You should include a plot of some of the digits that your classifier makes mistakes on – and discuss why that may be.

We implemented a batch gradient descent function, which updates our weight vector w by.

$$w := w - \eta \cdot \frac{gradient}{n} \tag{0.1}$$

where η is a constant which states how "big" a step we take down the convex in our descent, and n denotes the number of times we update our w vector. Our gradient vector is calcuated as

$$gradient := -X^{\mathsf{T}}(Y - \sigma(X \times w)) \tag{0.2}$$

To make sure our gradient descent stops when it has reached the (approximated) buttom or after n steps with length η, on the function, we check make sure our cost function always is descending. We defined our cost function as

$$cost := \frac{NLL}{n} = \frac{-\sum_{i=1}^{n} y_i \ln(\sigma(w^\top x_i)) + (1-y_i)\ln(1-\sigma(w^\top x_i))}{n}$$

This also makes sure, our η isn't so big, that we start ascending on the function.

When running the main function we train the dataset from auTrain.npz and get w by calling batch_grad_descent or mini_batch_grad_descent on the datapoints - the matrix X, and the target values - the vector y. The difference between the function is that in the mini-batch we divide our dataset into smaller batches containing points from the dataset chosen randomly. In both functions we call log_cost to find the gradient and value of the cost. In log_cost we calculate sigma_w_X. If sigma_w_X is to close to 0 or equals 1 we get problems because we then will take log(sigma_w_X) or log(1 - sigma_w_X) which equals infinity. Therefore if that's the case for sigma_w_X we then subtract or add a very small number so this won't happen.

After finding w we then preceed to test it on the test-data from auTest.npz. First we calculate $\sigma(w^\top x)$ which for each point gives the properbility for wether the point is in class 1, or not. If the probability is > 0,5 then the point is set to be in class 1. Afterwards we then compare with the target values from the given test-data-set.

PERFORMANCE

To show the performance of your gradient descent implementation include a plot of the cost function as a function of the number of steps taken. You should compare the running time and the convergence/output quality of mini-batch and full-batch gradient descent.

You should provide a figure that shows the parameter vectors for the case of classifying 2s versus 7s, and for the 10 all vs. one classifiers. Furthermore, your report should include results for the pairwise computations (at least two vs seven) as well as the results for the full classifier on the AU data set.

THEORERICAL QUESTIONS

- Sanity Check: If we randomly permute the pixels in each image and train the classifier, the classifier will be just as good, given we permutate all later given images the same way as the training data was permutated.

- Linear Separable: If the data is linearly separable, what happens to weights when we implement logistic regression with gradient descent? That is, how do the weights that minimize the negative log likelihood look like?

  Assume that we have full precision (that is, ignore floating point errors). We can run gradient descent on the data set for as long as we want (suppose God helps you). Now what will happen with the weights in the limit?

  Do they converge to some fixed number (fluctuate around it) or do they keep increasing in magnitude (absolute value)? Give a short explanation for your answer. What happens if we add regularization?

  If the data is linearly separable the gradient will converge to infinity, because the difference between the two classes becomes bigger and bigger. That is why it's a good idea to add a regularization parameter, since it will penalize the weight vector from getting too big.
  The weigth will converge to infinity or minus infinity depending on wether it is in the class in question or not.

- Bonus Question: Convexity of negative log likelihood. Show that the negative log likelihood function for logistic regression is convex. Is it still convex if we add regularization?