FYS-STK3155/4155 Project 2

Michael Bitney and Magnus Grøndalen

November 8, 2019

Abstract

The material for this project can be found in our GitHub-repository at:
https://github.com/michaesb/machine_learn_2.git

Introduction

Machine learning is today a growing field, with numerous breakthrough that makes a desirable method of handling big data in numerous industries. For example credit card data is big data sets, which previously done by employees manually. A bank needs to take a risk every time they make a loan and too many bad loans is what happened in economic crisis in 2008. This shows the importance of classifying the data properly and we look at how machine learning can tackle this problem.
Here we will look at Credit Card data and first look at it for Logistic Regression and then look how accurate this model. We then train a Neural Network with the same data and look at the difference in performance. We will look at how the Neural Network performs on a Regression on on the Franke function and compare to Linear Regression. Our goal is see how well these perform on a given data sets and see what is the best for the given case.
We have structured this paper in Theory, where we explain the theory behind our Data sets and Methods, Results for where we explain our parameter values and display the scores for the different methods. We then discuss the results in Discussion and then come with our Conclusion.

—-Just drafts—-

1. Write about machine learning in general
2. Write about Logistic Regression
   (a) Theory
      i. Cost function
      ii. Accuracy
      iii. Design matrix
      iv. Gradient descent
   (b) Results
   (c) Compare with Scikit-Learns logistic regression
3. Write about Neural Network implementation
   (a) Basic Theory - Neurons - Layers - Weights - Bias
   (b) Cost function
      i. Cross-Entropy
      ii. Mean Squared Error
   (c) Activation function
      i. Sigmoid
      ii. ReLU
      iii. Softmax

(d) Regularization

(e) Feed Forward

(f) Gradient Descent

(g) Backpropagation

—-End of drafts—-

Theory

Regression Methods

A regression method is a method using a statistical process to determine the relationship between one or multiple input parameters. Logistic Regression is a case where the output is binary, like win/lose or yes/no. Linear Regression is a case where we look for a linear function based on the inputs to give a result which it tries to find a linear relationship.

Linear Regression

For more information on how Linear Regression works, we invite the reader to check out the previous rapport on Linear Regression Methods [5].

Logistic Regression

Logistic Regression is a statistical model that lets you, unlike linear regression, perform classification on data sets.

Neural Network

A neural network is a machine learning technique with inspiration from the brain. It uses so-called neurons, or nodes, that is a weighted sum of its input neurons. We have an input layer of neurons, an output layer of neurons, and then we have the hidden layers of neurons. A neural network will always have one input layer and one output layer, but might have an arbitrary number of hidden layers. Each of the neurons in one layer is connected via weighted edges to the neurons in the next layer. The goal of the neural network is to find weights that produce a wanted output. In figure 1 we can see the behavior of a neuron. This is further explained in the paragraph below. Lets take a look at the components of a neural network.

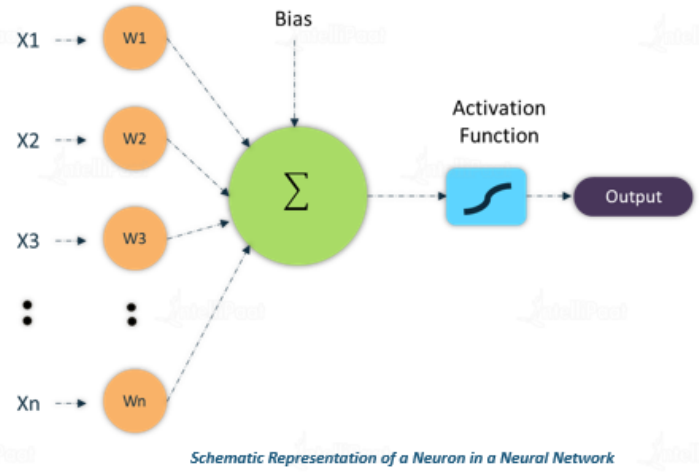


Figure 1: Figure showing the basics of a neuron in a neural network. Credit: https://intellipaat.com/

As stated above, a neural network has one input and one output layer, and an arbitrary number of hidden layers with an arbitrary number of neurons in each layer. Each neuron in one layer receives signals from all the neurons in the previous layer, and sends a signal to all the neurons in the next layer, much like a biological system. For a specific neuron y we have that

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b_i\right) \tag{1}$$

, where n is the amount of neurons in the previous layer, x_i is the value of the i-th neuron in the previous layer, w_i is the weight of the 'wire' between neuron x_i and itself. At last, the $f()$ denotes the activation function of the neuron. This is the output of a neuron, and there exists multiple different activation functions for machine learning. Examples are the sigmoid function, the Rectified Linear Unit (ReLU) and the softmax function, and they are normally used in different scenarios. We will only look at the sigmoid function in this project. You can see the equation for the sigmoid function in (2). In figure 2 you can see the sigmoid function plotted with the step function. The step function is what we would like to approximate, but because we need the activation functions derivative, we choose the sigmoid.

$$f(t) = \frac{e^t}{1+e^t} = \frac{1}{1+e^{-t}}. \tag{2}$$

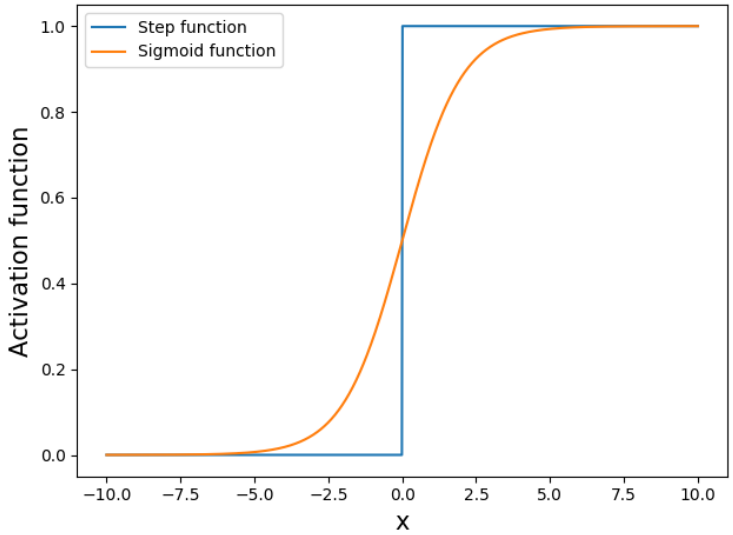


Figure 2: Figure showing the step function and the sigmoid function.

The method that takes the input variables and produces an output is called the feed forward method. It is responsible for adding the weighted sum forward through the network for each layer and neuron, through the activation functions, finally producing an output. In figure 1 you can see a visualization of this process for one neuron.

To validate whether our neural network is performing well, we need some kind of function that evaluates the fitness, or the error of the output from our neural network. This function is called the cost/loss function, and this is the function that we are trying to minimize. I.e. we want to optimize the weights and biases to minimize the cost function. There exists a variety of cost functions to choose from, but the simplest function is the Mean Squared Error (MSE) as you can see in equation (17). This cost function is probably the most popular and used cost function, but there is another cost function that is especially good in binary classification problems. This cost function is called the cross-entropy loss function, or the log likelihood, and the equation can be seen in (3).

$$C(y, \hat{y}) = -\sum_{i=1}^{n} \left(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)\right). \tag{3}$$

Where y is the wanted output data, and $\hat{y}$ is the output predicted by our neural network.

When we have all these components as described above, i.e. a cost function, activation functions and feed forward method, we need to find a way to improve the performance of the neural network. This is

done with something called backpropagation. It uses the errors at the output neurons, and "propagates" the error backwards through the network. To do this we need some kind of method that lets us change the weights to minimize the error from the cost function. One optimizer is the gradient descent, it lets you move the weights in the direction that decreases the cost function. First we look at the derivative of the cost function, using the cross-entropy loss in equation (3).

$$\frac{\partial C}{\partial \hat{y}} = -\sum_{i=1}^{n} \left(\frac{y_i}{\hat{y}_i} - \frac{1-y_i}{1-\hat{y}_i} \right). \tag{4}$$

We then propagate backwards using the chain rule.

$$\hat{y} = \text{sigmoid}(\hat{y}) = f(\hat{y}) \tag{5}$$

$$\frac{\partial \hat{y}}{\partial z_o} = f'(\hat{y}) = f(z_o) \cdot (1 - f(z_o)) = \hat{y} \cdot (1 - \hat{y}) \tag{6}$$

$$\frac{\partial C}{\partial z_o} = \frac{\partial C}{\partial \hat{y}} \cdot \frac{\partial C}{\partial z_o} \tag{7}$$

$$z_o = w_o \cdot a_h + b_o \tag{8}$$

$$\frac{\partial z_o}{\partial a_h} = w_o \tag{9}$$

$$\frac{\partial C}{\partial a_h} = \frac{\partial C}{\partial z_o} \cdot w_o \tag{10}$$

$$a_h = f(z_h) \tag{11}$$

$$\frac{\partial a_h}{\partial z_h} = f'(z_h) = a_h \cdot (1 - a_h) \tag{12}$$

$$\frac{\partial C}{\partial z_h} = \frac{\partial C}{\partial a_h} \cdot \frac{\partial a_h}{\partial z_h} \tag{13}$$

$$z_h = w_h \cdot X + b_h \tag{14}$$

$$\frac{\partial z_h}{\partial w_h} = X \tag{15}$$

$$\frac{\partial C}{\partial w_h} = \frac{\partial C}{\partial z_h} \cdot X \tag{16}$$

Now we have found all the derivatives we need to update the weights and biases for the neural network.

Neural Network Classifier

Write stuff specific to the classifier. Cost function, and therefore backpropagation also. Accuracy stuff

Neural Network Regression

Write stuff specific to the regression. Cost function, and therefore backpropagation also. Accuracy stuff

Testing and modeling error

Classification accuracy score and cross-entropy loss

Wriiiite

MSE and R^2-score

In order to test our regression models, we will use multiple methods to determine how well our models fit the data. Here we use an expression of the Mean Squared Error. This tells us the average of all the errors squared, which gives us an idea on how good a model is.

$$MSE(\hat{y},\hat{\tilde{y}}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \mathbb{E}[(y-\tilde{y})^2] \tag{17}$$

Here the $\tilde{y}_i$ is the model, y_i is the data, and $\mathbb{E}[]$ is the expectation value of an expression.

We will use an R^2-score function, which tells us how well the fitted line is to the data. The best score is 1 and the worst score is $-\infty$. An R^2 score of 0 is a straight line at the mean, which is considered noise. See equation (18) for the formula.

$$R^2(\hat{y},\tilde{\hat{y}}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2} \tag{18}$$

Franke function

The Franke function is a common test for reggression analysis with two inputs.

$$f(x,y) = \frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) \tag{19}$$

$$+\frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) \tag{20}$$

$$+\frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) \tag{21}$$

$$-\frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right). \tag{22}$$

This function is a common test for regression analysis and you can see the shape function in plot 3

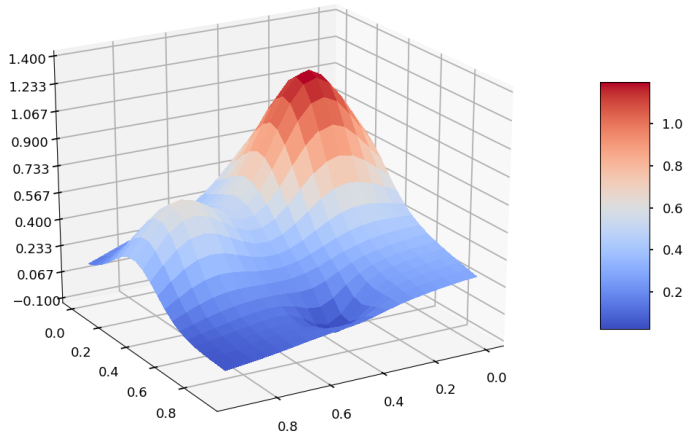


Figure 3: This plot shows the Franke function in a 3D plot

Credit Card data

Credit card data is used in economics to determine if an individual is suited to receive a loan from a bank. A loan is at a risk for the bank, although the bank is will demand more than the loaned amount, because might not be able to pay back the loaned amount, because of anything from economic succession to purchasing habits to to. A failure to pay back the loan is called a default. Credit Card data can provide a background to whether a person resources to determine if a person has the necessary resources to pay the loan back.

Before employees had to go through this information, but now we can use machine learning to solve this problem.

Results

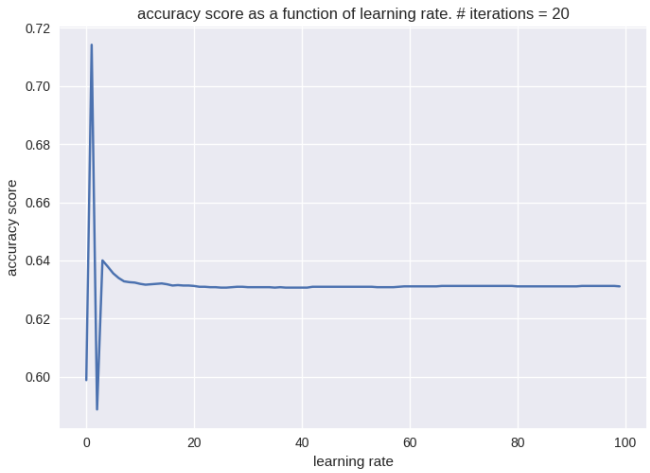
Data processing: Credit Card data

For our logistic Regression and Neural net we have used the Credit Card data. This Credit card data is credit card information from individuals from Taiwan and it shows information about an individual and whether they pay their bill or default. This data was retrieved from the site `https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients#` and is a common example for Logistic Machine Learning Algorithms. We have a description of the data given on the site and in our repository. We saw however that in the data that the were certain classifications that was not mentioned in the description. We decided to remove these individuals from the the data set and focus on the ones we can identify. We then have roughly 23 000 individual to process.

Logistic Regression

We constructed our Logistic Regression function using Gradient descent, but we needed adjust the parameters learning rate and number of iterations to find good model for the credit card data. We plotted multiple learning rates to try find the best one in graph 4

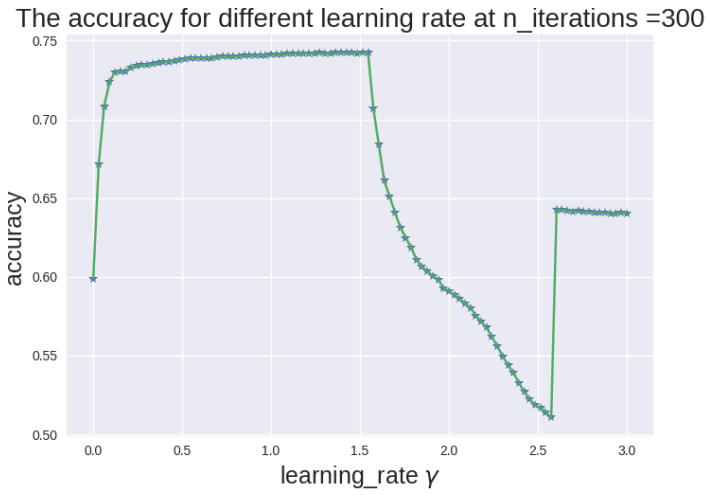
$$\text{Regression/learning}_{r}ate_{a}ccuracy_{n} =$$



$20v2.png$

Figure 4: Here we have plotted the accuracy vs the learning rate with 20 iterations in order to plot for many learning rates. Note the spike between 0 and 5

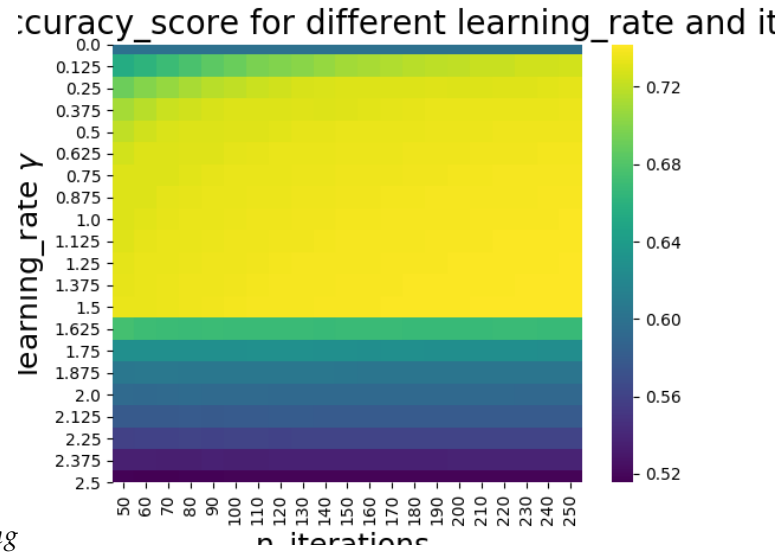
We saw a spike between 0 and 5 and decided to zoom on this area in graph 5



Regression/n$_i$t200$_l$r$_0$3.*png*

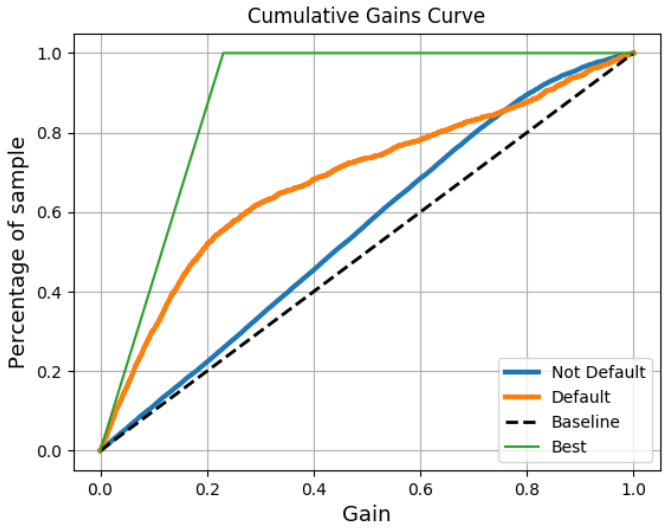
Figure 5: Here we see the a zoomed up version of graph **??** but with a higher iterations at 200.

We saw that the optimal value for the learning rate is in the interval [1.4,1.5]



Regression/LogReg$_h$*eatmap.png*

Figure 6: This is a heatmap of the accuracy where the x-axis is the learning rate and y-axis is number of iterations.



Regression/LogReg$_c umulative_g ain.png$

Figure 7: This is a cummulative gain plot that shows how well our function performs versus a perfect fit

Neural Network for Classification

Put plots and explain here.

Comparison

Classifying

Method	Training Cost	Testing Cost	Training Accuracy	Testing Accuracy
Logistic Regression	X	X	X	X
Logistic Regression Scikit-Learn	X	X	X	X
Neural Network	X	X	X	X
Neural Network Scikit-Learn	X	X	X	X

Neural Network for Regression Analysis

Put plots and explain here

Regression

Method	Accuracy	MSE
OLS	0.978	1.98e-3
Neural Network	0.781	0.
Neural Network Scikit-Learn (Single layer)	0.974	0.022
Neural Network Scikit-Learn (four layers)	0.9997	1.25e-3

Table 1: Here the mean squared error and the accuracy for different methods on the Franke Function. They were each given a mesh grid of 50 x 50 points and all have been spent time to find the best score by adjusting the values. Note that the Neural network code that we designed has only one hidden layer and scikitlearn has multiple layer. (A seed value was used to get the same values for all methods)

Discussion

Logistic Regression Neural net for classification Neural net for regression

Regression Franke function

In the previous rapport [5] we have looked at Regression method OLS and you can see the comparison between

Conclusion

Learned that OLS is quite good at Regression, when you have few points, but can be matched if you have a complicated neural network and enough computational power. We would recommend OLS, because simplicity will provide clearer results and then More layers in the code could make it a better comparison to the scikitlearns neuralnetwork

References

[1] Lecture Notes made by Morten Hjorth-Jensen in class Fys-Stk4155/3155 at uio
`https://compphysics.github.io/MachineLearning/doc/web/course.html`

[2] Hastie Tibshiran Friedman; The Elements of Statistical Learning; Second edition; Springer 2009

[3] Géron, Aurélien; Hands-On Machine-Learning with Scikit-Learn & TensorFlow; First edition; O'Reilly 2017

[4] Yeh, Lien; The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients;

[5] Grøndalen Bitney; Project Regression for the subject FYS-STK3155/4155
`https://github.uio.no/michaesb/ml_project1_mms/blob/master/report_fysstk_magnubgr_michaesb.pdf`

Acknowledgements

Thanks to Morten Hjorth-Jensen and student teachers in FYS-STK-4155/3155 for help during this project.