Department of Mathematics

SCP8082660 - Machine Learning

Looking at neural networks

Author:
Håkon Bygdås Høgset

January, 2022

Table of Contents

1 Introduction

For this project I wanted to model my own neural network from scratch using python and the backpropagation algorithm. Depending on the input parameter my model uses either one hidden layer or no hidden layer at all. I wanted to observe how the accuracy of the model is affected by using different amount og nodes in the hidden layer. In addition I wanted to see how the accuracy changes when the learning rate α and number of epochs are adjusted.

2 Dataset

The dataset I am using is taken from (Nick Street 1996) and consists of 569 instances of data with attributes and values related to breast cancer.

The dataset consists of training data (x_training and y_training) and testdata (x_test and y_test). Each instance in the dataset consists of 30 attributes including a binary classifier saying if a tumour is cancerous (1) or if it is noncancerous(0).

The x_test dataset is fed to the network and passed forward to the output. At the output layer the result is being compared to see if it has the same value corresponding to the value in y_test dataset.

3 Code

In my model I use the backpropagation algorithm to create my model. The pseudocode was found from *2.5 backpropagation* 2015.

```
1  Initialize all weights and biases in network;
2  while terminating condition is not satisfied{
3      for each training tuple \bold{X} in D{
4          // Propagate the inputs forward:
5          for each input layer unit j {
6              O_j = I_J; // output of an input unit is its actual input value
7          for each hidden or output layer unit j{
8              I_j = ∑(w_ij O_i + θ_j //compute the net input of unit j with respect to the
9                  previous layer, i
10         O_j = 1/(1+e^-I_j) // compute the output of each unit j
11         // Backpropagate the errors:
12         for each unit in the output layer
13             Err_j = O_j(1 − O)(T_j − O_j); // compute the error
14         for each unit j in the hidden layers, from the last to the first hidden layer
15             Err_j = O_j(1 − O_j)Σ_k Err_k w_jk; // compute the error with respect to the
16                 next higher layer, k
17         for each weight w_ij in network {
18             Δw_ij = (l)Err_j O_i;// weight increment
19             w_ij = w_ij + Δw_ij; // weight update
20         for each bias θ_j in network {
21             Δθ_j = (l)Err_j; // bias increment
22             θ_j = θ_j + δθ_j;} // bias update
23         }}
```

In my code file I have three different classes; the Node class, a class for the neural network and a unittest.

The Node class is meant to represent a perceptron in the neural network and takes three parametres; weights, input and output.

The neural network class creates instances of neural nets based on the dataset. A neural net consists of different number of hidden layers depending on the implementation. In my model I use either one hidden layer or none hidden layers, depending on the input parameter.

In the unittest the test data is used to observe the accuracy of the model. In this section I created some instances of a neural network to compare the accuracy together with the running time based on different seettings. For example to change the learning rate and change number of nodes in the hidden layer and cpmpare the results.

4 Theory

In the neural network there are multiple layers consisting of multiple nodes. The nodes represents perceptrons which is an algorithm to classify linearly speparable outputs.

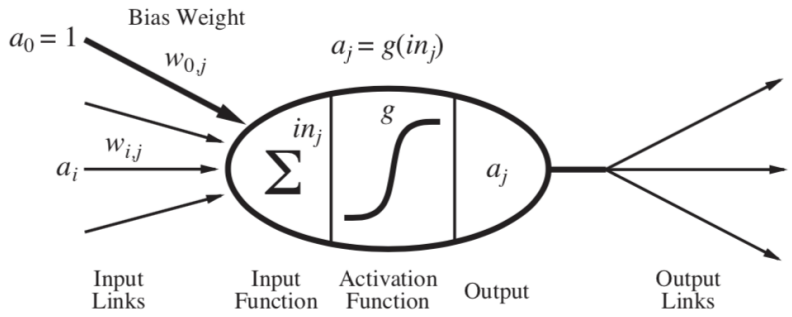

Figure 1: Perceptron (from Nóbrega 2021)

The perceptron takes in outputs from every node in the previous layer. Every input is multiplied with a weight which determines the strength of the signal. These weights are initiated randomly in the start and later tuned for every iteration. The sum of these inputs are added together and taken as input in an activation function. The activation function uses this data to produce an output and varies from model to model. In this model I am using the sigmoid function which produces a probability value between 0 and 1, and works good for classifying each instance of my dataset. These value are then passed to the next layer which again produces an output for the next layer and so on. This is called the forward pass. When the output values reaches the output node the opposite happens. Instead of feeding the input forward, the error is being tracked backwards. After making a classification, a check to see whether the output is correct or not is made and how far off the result is. The error is then being processed and propagate backwards in the network and thus adjusting the weights of the nodes accordingly.

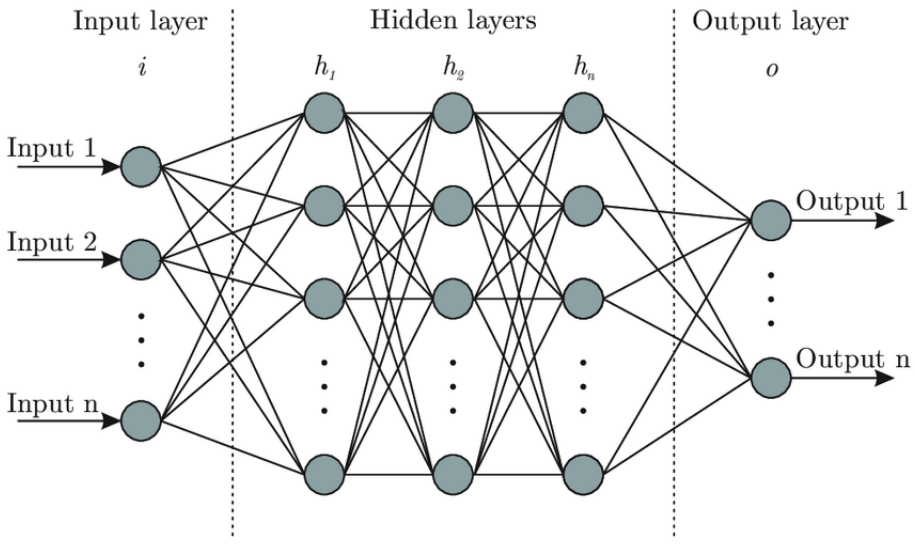

Figure 2: Neural Net (from Halstensen 2020) - the data is fed to the input layer and fed forward. At the output layer the error is calculated and is being propagated back adjusting the weights

5 Results

The number of times a dataset is being iterated is known as epochs. For every epoch the perceptrons in the neural network are getting closer to their potential for learning. The changes will therefore be larger in the models using fewer epochs than the models employing more iterations as figure 3, 4 and 5 also shows. This is because the adjustments will be more and more finely tuned for every epoch. In figure 4 showing the models using 20 epochs it can be observed that the difference in accuracy are larger than in the figure 4 and 5 using more epochs.

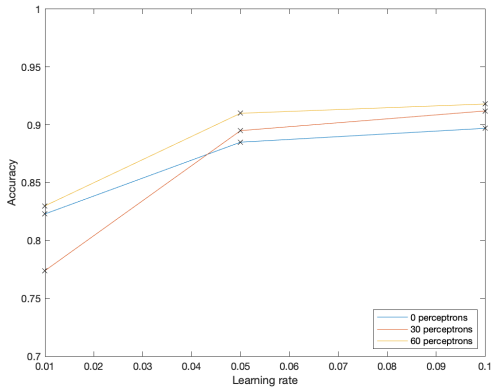

Figure 3: Accuracy as function of learning rate with 20 epochs

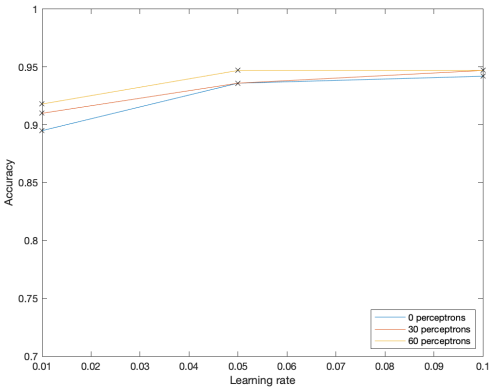

Figure 4: Accuracy as function of learning rate with 100 epochs

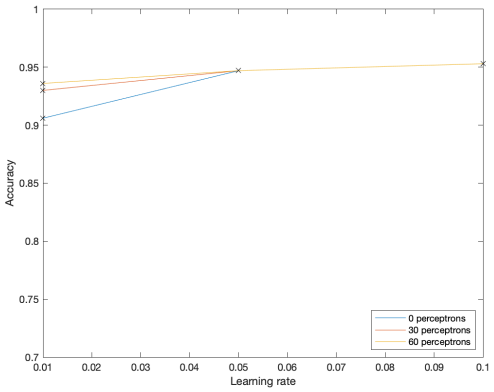

Figure 5: Accuracy as function of learning rate with 200 epochs

By comparing figure 3 with all different models using 20 epochs, figure 4 with using 100 epochs and figure 5 showing models using 200 epochs it can be observed that there is a noticable increase in accuracy when increasing the learning rate. This goes for all graphs with a slightly larger difference for the graph with the least epochs. A lower learning value will make the model learn slower than with a higher value thus also affecting the model with lower number of epochs the most.

Also number of nodes in the hidden layer will have a impact on the accuracy of the model. This can also be observed in the figures above. A hidden layer with too few nodes will be a bottleneck for the result as the parameters will be limited. Too many will increase the runtime without necessarily return better results. Therefore I chose number of nodes for the hidden layer to be 30 and 60.

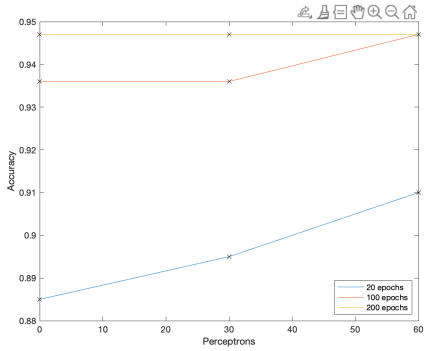

Figure 6: Models using learning rate $\alpha = 0.05$

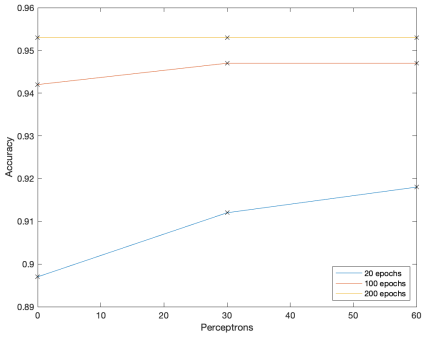

Figure 7: Models using learning rate $\alpha = 0.1$

Another interesting observation is that for the models using higher epochs the number of perceptrons doesn't seem to affect the results very much. Figure 7 shows very small to no change in increase for the models using higher epochs and a larger increase for the model using lower epochs. In other words will a high number of epochs outweigh the use of more perceptrons. This is also showed in figure 8, 9 and 10 showing the relation between number of perceptrons, accuracy and running time.

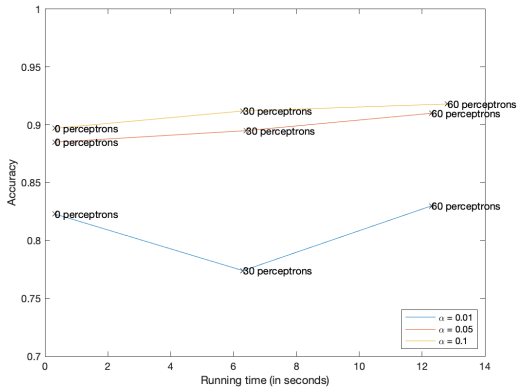

Figure 8: Accuracy as function of learning rate with 20 epochs

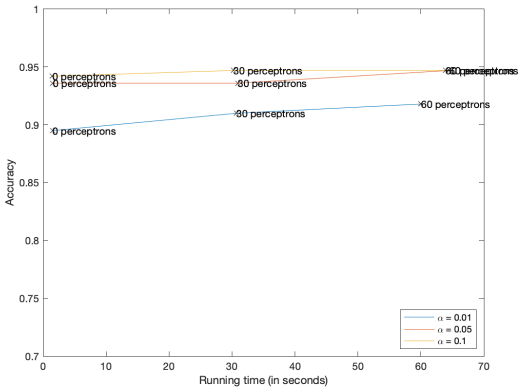

Figure 9: Accuracy as function of learning rate with 100 epochs

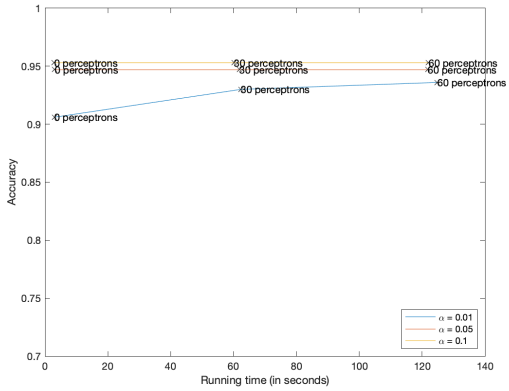

Figure 10: Accuracy as function of learning rate with 200 epochs

From these figures it can be observed that the best results will be given by reducing the number of perceptrons and increasing the number of epochs combined with a higher value for the learning rate. This will result in a running time of around 3 seconds and a accuracy of around 0.953. Also a higher value of the learning rate seems to result in more consistent calculations compared to lower value of learning rate, especially combined with lower number of epochs.

Bibliography

2.5 backpropagation (2015). URL: https://www.slideshare.net/Krish_ver2/25-backpropagation (visited on 19th Jan. 2022).

Halstensen, Simon Peder (2020). *Studentblogg*. URL: https://www.uio.no/studier/emner/matnat/fys/HON1000/v20/studentblogg/artificial-neural-network-architecture-ann-i-h-1-h-2-h-n-o-%281%29.png (visited on 19th Jan. 2022).

Nick Street Dr. William H. Wolberg, Olvi L. Mangasarian (1996). *Breast Cancer Wisconsin (Diagnostic) Data Set*. URL: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic) (visited on 19th Jan. 2022).

Nóbrega, André (2021). *Redes neurais: a utilização de IA nas artes*. URL: https://lh4.googleusercontent.com/ThRuY_tAATuDAowCgjFdfLpm1xG4K2au1k_XiGpcdNAV2lf5XAY1307wybxOtPTwlfVNnPo08at0kp__CUttMC75U75VduYiel97vl6ZaaKb7G0-6K2HPOdUVvr79N123ydt9gV0 (visited on 19th Jan. 2022).