Home assignment 3

Exercise 1. Choice 1: Neural network

(implemented in Python)

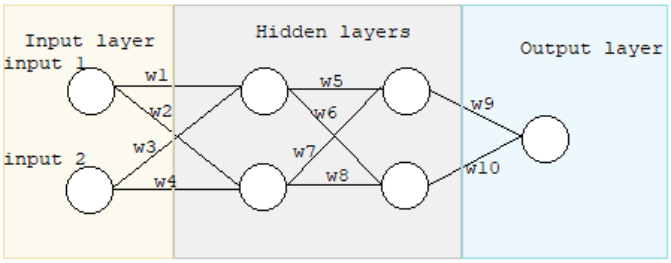


Figure 1. Neural network with 2 hidden layers and 2 neurons in each hidden layer

1. Building a neural network begins with network initialization. The weights, bias and the structure of the network are initialized.
2. Next step is forward propagation. In forward propagation, we apply a set of weights to the input data and calculate an output. For the first forward propagation, the set of weights is selected randomly. [1] Example: for the first neuron in hidden layer the value is calculated: value = w1 * input1 + w3 * input2. Then the activation function is applied. In this case Sigmoid function. $f(x) = \frac{1}{1+e^{-x}}$
3. The second step is repeated up to output layer.
4. Next step is back propagation. In back propagation, we measure the margin of error of the output and adjust the weights accordingly to decrease the error. [1] The error is calculated using sum of squares error, where y is the target value and $\hat{y}$ is the predicted value. $\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
5. All steps are repeated. Epoch is the measure of the number of times all the training vectors are used to update the weights.

Results

The target value was 1.

Epochs	Predicted value
50	0.9183001933804461
100	0.9570134498802589
200	0.978091080730645
500	0.9911751505906806
1000	0.9955844189868166

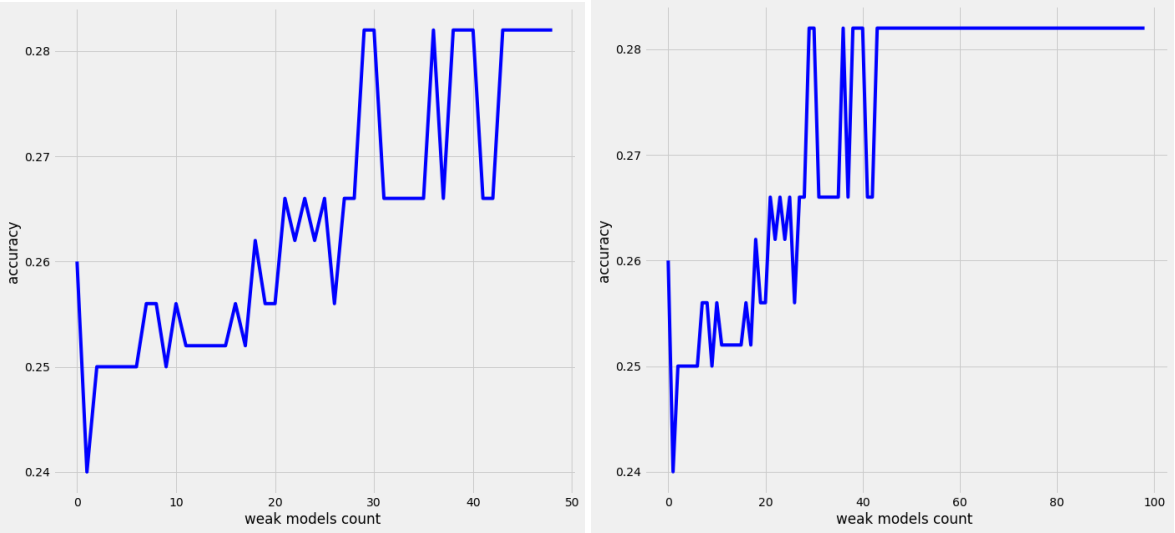
It can be seen, that the best result is with 1000 epochs, but after 500 epochs the result didn't get much better, so it wouldn't be necessary to train too much, 500 epochs would be enough.

Exercise 2. Ada Boost algorithm

Ada Boost combines several models to improve the final predictive performance. It combines several weak learners, these are classifiers that produce prediction that is slightly better than random guessing.

AdaBoost assigns a “weight” to each training example, which determines the probability that each example should appear in the training set. Examples with higher weights are more likely to be included in the training set, and vice versa. After training a classifier, AdaBoost increases the weight on the misclassified examples so that these examples will make up a larger part of the next classifiers training set, and hopefully the next classifier trained will perform better on them. [6]

1. Assign weights to each training sample
2. Train weak classifier
3. Update weights, weights of the misclassified training instances are increased.
4. Next classifier is trained and acknowledges the updated weights and repeats the procedure.



Code:

https://gitlab.cs.ttu.ee/Tiina.Sumeri/iti8665-2019/tree/master/homework3

Materials:

[1] https://stevenmiller888.github.io/mind-how-to-build-a-neural-network/

[2] https://blog.zhaytam.com/2018/08/15/implement-neural-network-backpropagation/

[3] https://enlight.nyc/projects/neural-network/

[4] https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf

[5] https://towardsdatascience.com/adaboost-for-dummies-breaking-down-the-math-and-its-equations-into-simple-terms-87f439757dcf

[6] http://mccormickml.com/2013/12/13/adaboost-tutorial/

[7] https://www.python-course.eu/Boosting.php