Report

Neural Network - Mini-Project Number 1:
Adaline algorithm, Backpropagation algorithm

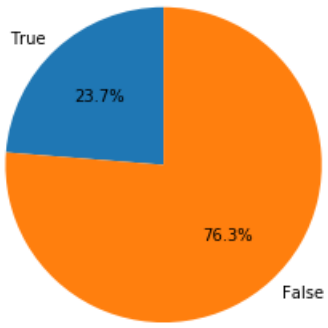
Our IDs:
Hila 207931106
Eli 206416687

Section 1:

General Explanation and Time it took to train the final model

In our project we implemented the Adaline and Backpropagation algorithms, and checked the results by splitting the data (breast-cancer-wisconsin dataset) into 2 parts: 66% of the data was used for training the algorithms, and the remaining 33% was used for testing (did not participate in model training).

But, with the aim of solving the imbalance-data problem (only 47 positive examples against 151 negative examples), we first divided the data into a group of positive results and a group of negative results. Then we split each group into train and test (took 33% for testing and 66% for training). And only then we append the groups into one and defined x_train, x_test, y_train, y_test.



This way, we promised that there would be results from both groups in each train and test group.

Regarding the imbalance: We assume that if there is more negative data than positive, it means something, and we have to take that into account in the model.

That is, we cannot change the quantities and take equally of the two groups because it distorts the data.

In addition, we did a cross-validation process, i.e we've change the 33% choice 3 times and get 3 results.

Adaline Results:

- The time it took to train the final model (after all the cross-validation processes): 3.31566 seconds.

BackPropagation Results:

- The time it took to train the final model (after all the cross-validation processes): 12.2 seconds.

Section 2:

Ease of parameter search/adaptation

Adaline:

We got the above results when we trained our model 50 times (iterations).

We chose this number because for fewer iterations, we got relatively low accuracy, and for more iterations - not much changed.

Therefore, we have concluded that 50 iterations are a sufficient number where the algorithm converges to the local minimum of error.

The learning rate we've used was 0.01.

BackPropagation:

We defined a 3-layered hidden feed forward neuron network:

the first one with 200 neurons, and the other two contain 100 neurons each.

The choice was made by many trial and error, until we reach satisfactory results.

We've trained our model with 700 iterations.

The alpha value we choose is 0.0001.

Section 3:

Performance on the training set and on the testing set, and explanation of the results

Adaline Results:

- The average of these 3 results: 78.6885%

- The standard derivation: 0.013385189851274211

Confusion matrix for the first evaluation -

	Predicted: yes	**Predicted: no**
Actual: yes	34	12
Actual: no	13	2

Confusion matrix for the second evaluation -

	Predicted: yes	**Predicted: no**
Actual: yes	40	6
Actual: no	9	6

Confusion matrix for the third evaluation -

	Predicted: yes	**Predicted: no**
Actual: yes	36	10
Actual: no	12	3

BackPropagation Results:

- The average of these 3 results: 81.967%

- The standard derivation: 0.013385189851274211

Confusion matrix for the first evaluation -

	Predicted: yes	**Predicted: no**
Actual: yes	46	0
Actual: no	10	5

Confusion matrix for the second evaluation -

	Predicted: yes	**Predicted: no**
Actual: yes	46	0
Actual: no	11	4

Confusion matrix for the third evaluation -

	Predicted: yes	**Predicted: no**
Actual: yes	46	0
Actual: no	12	3

The results made sense to us because the dataset is relatively small and does not contain much data, And the results are pretty good.

In addition, it is you can pay attention that our model is better than a "silly" model that will always predict the more common result: false.

Section 4:

Summary and discussion of process/ problem

We really enjoyed studying these important and famous algorithms in depth by implementing them.

It was a little difficult to figure out how to best implement the Adaline algorithm.

We finally decided to do a class to represent the Adaline model, and has methods of fit, activation, predict and score.

And from the main, after arranging the Database, we fitted the model and train it.

It was also challenging to find the right Python libraries to use.

In the BackPropagation algorithm we've used sklearn.neural_network library.

And in general in the project we used the following libraries: numpy, pandas, sklearn.model_selection (for train-test split).

There were also some problems of missing data (places where there were "?"s in stand of a given number), so we decided to put "0". The reason is that we check and see that there is a large amount of zeros in this column (87), and also a lot of small numbers. So it would seem to us that '0' is a result that probably makes sense out there, and won't compromise the reliability of our data.

In summary, we are pleased that we have been able to overcome the problems of working with a real set, and we have been able to apply the model we implemented ourselves! Hope the results are satisfying… ☺