Lab 5: Perceptron / Neural Networks

Artificial Intelligence

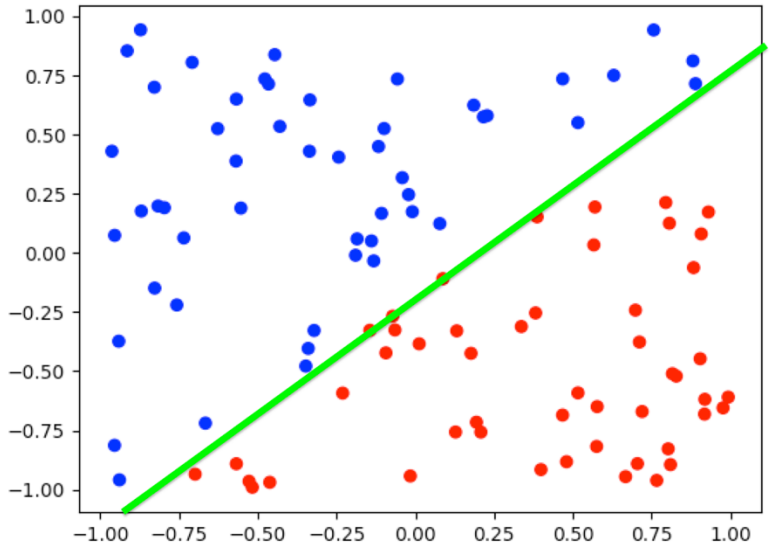
Eduardo Vaca

A01207563

Part I

In this lab we had to implement a perceptron and train it with several examples to determine a set of predictions or print *no solution found* in case the data wasn't linear separable.

In one of those examples there was one with data linear separable. If we create a scatter plot of that data, we obtain the following:



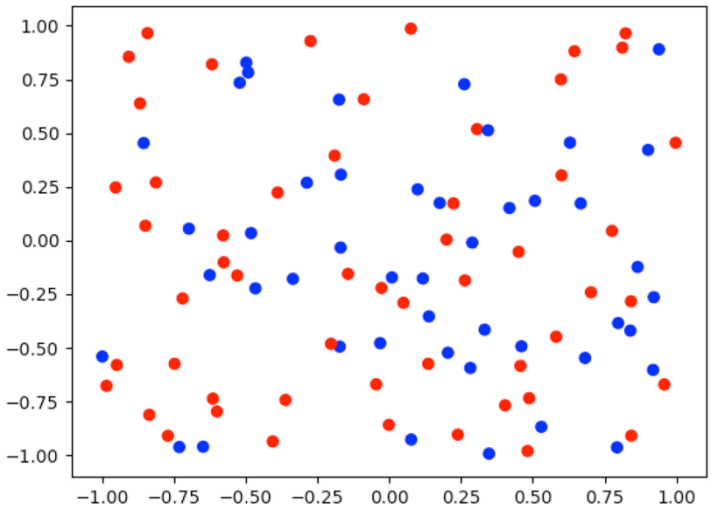
As you can see with the green line, the data is linear separable.

This perceptron function could be represented with:

$$f(x) = 1 \; if \; (x[0] * 0.0124 + x[1] * -0.014255) > threshold \; else \; 0$$

(Approx.)

Another example from the tests section contained data which wasn't linear separable. If we plot this data, we obtain the following:

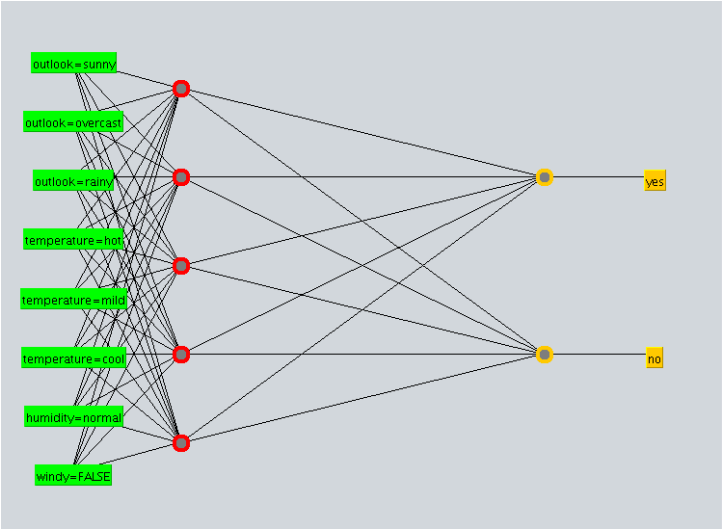


Notice how it's not possible to create a line which separate both colors.

Part II

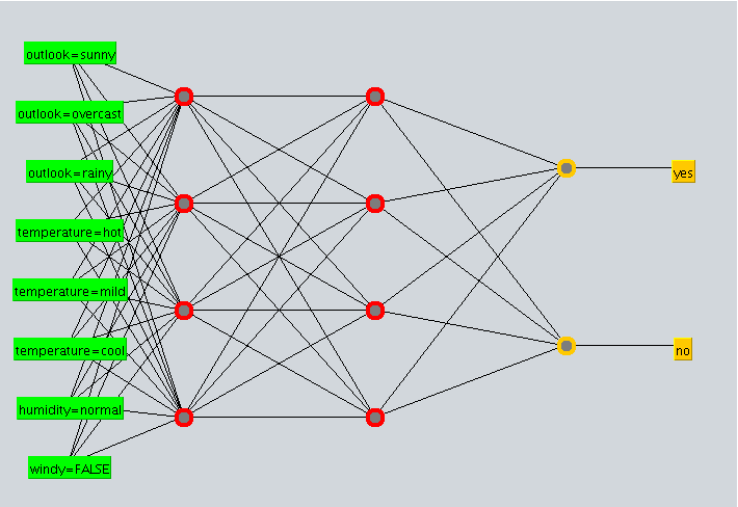
Another part of the lab was to use WEKA tool to play around different attributes from a ANN such as number of nodes, hidden layers, etc.
I used as data the famous weather dataset (can be found in https://gist.githubusercontent.com/myui/2c9df50db3de93a71b92/raw/3f6b4ecfd4045008059e1a2d1c4064fb8a3d372a/weather.nominal.arff)

Here are some of the ANN I created:



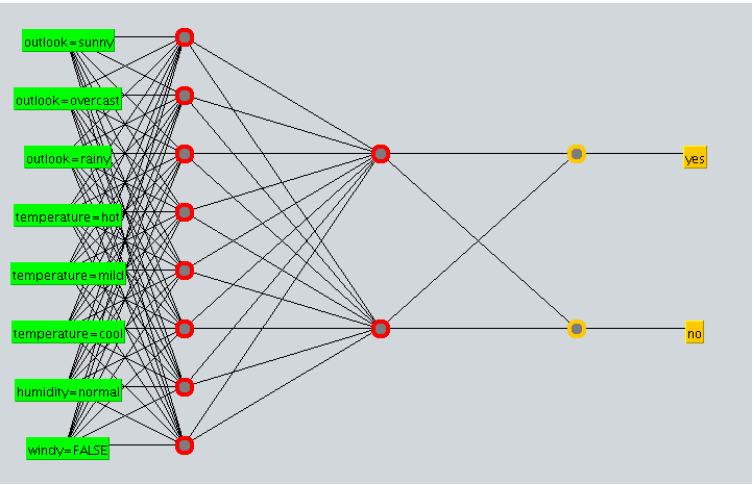
Example #1

Learning rate	0.3
Momentum	0.2
Hidden Layers	1
Nodes	5
Time to build model	75.2 sec
Time taken to test	0 sec
Correct classify instances	100%



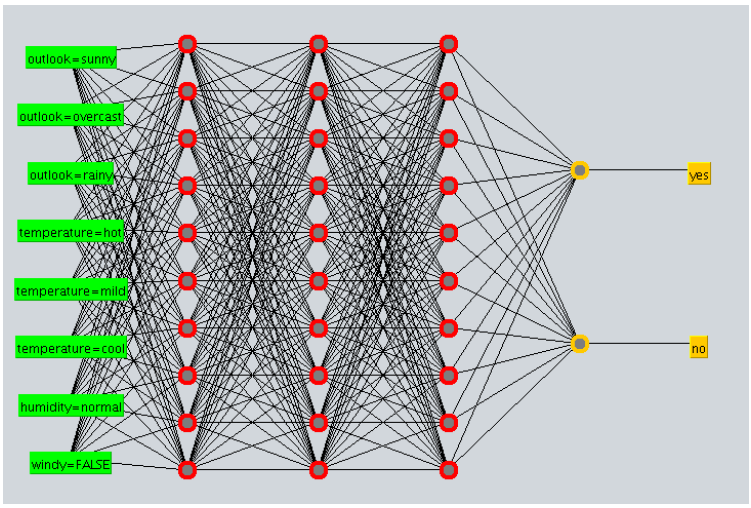
Example #2

Learning rate	0.3
Momentum	0.2
Hidden Layers	2
Nodes	4, 4
Time to build model	13.82 sec
Time taken to test	0 sec
Correct classify instances	100%



Example #3

Learning rate	0.15
Momentum	0.2
Hidden Layers	2
Nodes	8, 4
Time to build model	11.2 sec
Time taken to test	0 sec
Correct classify instances	100%



Example #4

Learning rate	0.4
Momentum	0.2
Hidden Layers	3
Nodes	10, 10, 10
Time to build model	11.42 sec
Time taken to test	0 sec
Correct classify instances	100%

Summary provided by WEKA

Example #1

```
Time taken to build model: 75.2 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances          14               100      %
Incorrectly Classified Instances         0                 0      %
Kappa statistic                          1
Mean absolute error                      0.0245
Root mean squared error                  0.0354
Relative absolute error                  5.2713 %
Root relative squared error              7.3845 %
Total Number of Instances               14
```

Example #2

```
Time taken to build model: 13.84 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances          14               100      %
Incorrectly Classified Instances         0                 0      %
Kappa statistic                          1
Mean absolute error                      0.0252
Root mean squared error                  0.0322
Relative absolute error                  5.429  %
Root relative squared error              6.7075 %
Total Number of Instances               14
```

Example #3

```
Time taken to build model: 11.25 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances          14               100      %
Incorrectly Classified Instances         0                 0      %
Kappa statistic                          1
Mean absolute error                      0.09
Root mean squared error                  0.1059
Relative absolute error                 19.3948 %
Root relative squared error             22.0954 %
Total Number of Instances               14
```

Example #4

```
Time taken to build model: 11.48 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances          14               100      %
Incorrectly Classified Instances         0                 0      %
Kappa statistic                          1
Mean absolute error                      0.0396
Root mean squared error                  0.0514
Relative absolute error                  8.5306 %
Root relative squared error             10.7272 %
Total Number of Instances               14
```

As you may notice from the graphic representation of the networks, these change drastically by alerting some attributes of it. We can have a very basic network like the first example or a very complex as the last example with 3 hidden layers with 10 nodes each.
Another difference were the values of Relative absolute error and root relative error, which make sense they vary because we are changing the complexity, the equation that represents the network and also the speed we mode through the graph with the learning rate.

Questions

What are the ANNs good for?

They are good to approximate any function, also for complex problems like image recognition. They scale well to larger datasets mimicking the brain. Also, they are good for organic learning, artificial neural networks have the ability to generalize their inputs. Nonlinear systems have the capability of finding shortcuts to reach computationally expensive solutions and with this, ANNs are very valuable in commercial big data analysis.

Where would you use them?

Some of their most use applications are:

- Financial: Stock market prediction, corporate financial analysis, document readers, etc.
- Medical: Cancer cell analysis, breast cancer prediction
- Speech: Speech recognition, speech classification, etc.
- Software: Pattern recognition/ facial recognition, optical character recognition
- Control: Make steering decisions of physical vehicles

I would sue them specially in Medical applications like predicting the existing of any type of cancer given some symptoms, this would cause huge benefits for humanity.

Are they worth the effort implementing or not?

For sure! They are one of the most accurate techniques of machine learning. They could be slow but their accuracy essential to many fields in these days. Something important to say is that developers need to distinguish when a ANNs is needed or a simple Decision Tree could do the trick.

What kinds of problems do they not solve?

Probably the main downside of ANNs is that you don't get much information of what is actually happening inside the network. This limits the understanding of many events.

Having said that, some of the problems that an ANN cannot solve are:

- Understanding natural language
- Robust analysis of video streams
- Question answering
- Basically, any problem that cannot be implemented in a learning model is going to be a problem for a neural network.