Jiamin Xuan jx624@nyu.edu
Eric - Applied Data Science
October 20, 2014

Applied Data Science-Lab 2

Neural Network

1)

AND:
(0,0)->0 (0,1)->0 (1,0)->0 (1,1)->1
OR:
(0,0,)->0 (0,1)->1 (1,0)->1 (1,1)->1
NOT:
0->1 1->0
NOR:
(0,0) ->1 (0,1)->0 (1,0)->0 (1,1)->0

a) The neural network works perfect in this case, although sometimes the 0 is showed as 0.99989879*10^-32. Actually, the current setting is the optimal one. we might increase the epochs or alter the parameters (learning rate or momentum)

b) for this case, increasing hidden layers or hidden nodes does not improve the effectiveness. Using 1 layer can approximate any function that contains a continuous mapping from one finite space to another. For hidden nodes, there are several rules, as I quote:

- The number of hidden neurons should be between the size of the input layer and the size of the output layer.
- The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.
- The number of hidden neurons should be less than twice the size of the input layer.

As shown below, the proper number of layers or nodes would greatly affect the output:

using 1000 nodes:

```
Testing on data:
out:     [  0.000]
correct: [  0.000]
error:  0.00000000
out:     [  0.000]
correct: [  0.000]
error:  0.00000000
out:     [  0.000]
correct: [  0.000]
error:  0.00000000
out:     [  1.000]
correct: [  1.000]
error:  0.00000000
All errors: [9.5431376007519899e-16, 2.828110455310275e-16, 2.0955772502546866e-15, 2.1801040887457848e-16]
Average error: 8.87678116184e-16
('Max error:', 2.0955772502546866e-15, 'Median error:', 9.5431376007519899e-16)
[Finished in 3.9s]
```

using 4 nodes:

```
Testing on data:
out:     [ -0.000]
correct: [  0.000]
error:  0.00000000
out:     [  0.000]
correct: [  0.000]
error:  0.00000000
out:     [  0.000]
correct: [  0.000]
error:  0.00000000
out:     [  1.000]
correct: [  1.000]
error:  0.00000000
All errors: [3.0198581527991858e-31, 2.2186712959340957e-31, 2.4651903288156619e-32, 6.1629758220391547e-33]
Average error: 1.38666955996e-31
('Max error:', 3.0198581527991858e-31, 'Median error:', 2.2186712959340957e-31)
[Finished in 11.2s]
```

using 2 layers:

```
Testing on data:
out:     [ -0.012]
correct: [  0.000]
error:  0.00007667
out:     [  0.001]
correct: [  0.000]
error:  0.00000053
out:     [  0.010]
correct: [  0.000]
error:  0.00004763
out:     [  0.999]
correct: [  1.000]
error:  0.00000066
All errors: [7.6667208837793199e-05, 5.306340222941214e-07, 4.7629370853165804e-05, 6.5707964004932863e-07]
Average error: 3.13710733383e-05
('Max error:', 7.6667208837793199e-05, 'Median error:', 4.7629370853165804e-05)
[Finished in 13.2s]
```

we can see, using 1 layer and 2~4 nodes is the best option.

c) the training size increased to 8 and input nodes increased to 3, according to the optimal rules of node, we should use 2 layers and using 2 or 3 nodes.

A	B	C	Result 1)	Result 2)	Result 3)
0	0	0	0	0	1
0	0	1	1	1	1
0	1	0	0	0	1
0	1	1	1	0	0
1	0	0	0	0	1
A	B	C	Result 1)	Result 2)	Result 3)
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	0	0

2)

a) the tuning does a fare job. according to the background chosen and other image features, and the different angle of the face, the tuning could be sometimes not so useful. I believe improve the quality of the photo(in reducing size phase), using a clear background and standardize the position of the eye, mouth, etc. would be helpful.

b) it doesn't work very well on group pictures. If I want to make the ANN work here, I need to tell the ANN where is face, is it a part of face or the whole face ? (that is to set the pictures to the same structure) otherwise the ANN would not have a good performance.

c) the ANN can hardly make the right output. Different from standard photo, group photo usually is not good from those prediction/classification. it has shadows, emotions on face and sometimes there are only a part of the face(people always stand in front of others blocking other's face). And shadow is also making this more difficult.

3)

it's doing good. please see the attached code.

4)

a)Linear Reg. vs. Logistic Reg. vs. Neural Nets.

Linear Regression, assumes the there is linear correlation between the dependent variable *Y* and one or more independent variable *Xi*. By minimize the errors, one can find the optimized parameters . The dependent variable in most

cases are continuous variables, and the method is mainly used for predicting continuous dependent variables like house prices.
Logistic Regression is mainly used for classification where function is a logistic function, the optimization process is to find the best parameter that maximize the probability(likelihood) of correct classification.
Neural networks can be reduced to logistic regression if the neural network don't have hidden layers and use the logistic function as activation function. That is to say, neural nets are mainly used for classification. In many cases, people refer neural networks to those with multiple hidden layers. These Neural nets are non-linear models, they can mimic all kinds of functions, and it often provides more accurate result then logistic regression. The drawbacks are the time it takes to train the networks, and demand of generalization (avoid over-fitting). And the most different thing is, ANN has a system to automatically adjust the weight to let the model better fit the data(e.g. BP model).

b)
Machine learning has been proved to be powerful tool in many fields, including finance, medical care, biology, etc. But I think the ultimate goal for machine learning is to achieve Artificial Intelligence. There are different approaches towards this goal, which I believe the closest is the Deep Learning Approach and Graphical Models. Inspired by Biological Neural Science, the Deep Learning approach is mainly based on the Multi-layer Neural Network algorithms. The key concept of having the multi-layer structure is to enable automatic feature extraction, which other wise are selected and created by human. Though great amount of computation power is needed, new techniques and better hardware emerged in the passed decades is making this approach easier and easier. Thus, there are lots of successes in revolutionizing traditional techniques such as image recognition, NLP and so on.

I'm not sure about the definition of classical statistics, actually we have statistical learning which combines the theory for machine learning and statistical. Actually, "Statistical learning theory is a framework for machine learning drawing from the fields of statistics and functional analysis".

However, if you are talking about statistical modeling, I think it will become less important but it will always have a place in data science. In some cases, we have a simple relation or less variables so we can build the model easily using classical statistics (like in a perfect experiment environment). Moreover, knowledges in statistics sometimes help machine learning to work better.