Evaluating ANNs

Summary

Artificial neural networks (ANNs) are an area of main focus when it comes to machine learning. Inspired by how our own brains work, many functions and processes of ANNs have taken up similar biological terminology. Information in our brain is passed along by the firing of synapses between neurons; the first ANNs followed similar properties by utilizing binary inputs and outputs. As time passed the ANN's evolved to develop perceptrons. Perceptrons are similar to the original neurons, but are based on Linear Threshold Units (LTU) instead of binary values, the output is the collective weighted sum of the inputs. These algorithms further evolved with the introduction of backpropagation, which takes into consideration the error gradient across the weights in a reverse direction.

The point being, neural networks unlike the algorithms we have worked with thus far can be fine-tuned using many different hyper-parameters, and the layers in between the input and output (hidden layers) can sometimes be a black box model. Therefore the purpose of this paper is to understand the effect some of these parameters such as number of layers and neurons within each layer, have on the performance of neural networks.

Research Design

We will once again leverage the MNIST dataset and try fit an artificial neural network to it. An ANN will be obtained using both Scikit Learn and TensorFlow. The parameters that will be altered are the number of layers (2 and 5) and the number of nodes within each layer (10 and 20). For this reason a total of four different models will be evaluated. The two main evaluation criteria's that will be used are the processing time and accuracy of the predictions.

Data Analysis

Evaluating ANNs

In the MNIST dataset there is a total of 70000 observations. For all the models we will set aside 60000 for the training and 10000 as a hold outset. However in the models created using Scikit Learn, 5000 out of the training set will be set aside for validation within training.

Model Evaluation

The evaluation results obtained from ANNs using Scikit Learn are illustrated in Appendix A, Figure 1. Keeping the number of layers the same and increasing the number of nodes per layer, increases the accuracy slightly but the processing time differs depending on whether we used 2 layers or 5 layers. The same can be said with increasing the number of layers and keeping the number of nodes the same, the accuracy increases but the there is no real correlation with processing time.

A similar evaluation was carried out using TensorFlow, results seen in Appendix A, Figure, 2, 3, 4 and 5.

From the results obtained it seems that increasing the number of nodes while keeping the layers the same decreases the accuracy and increases the time taken, and similar results are obtained by increasing the number of layers and keeping the nodes the same. One thing that can be said is that increasing the number of layers does not increase the performance time as much as increasing the number of nodes.

Recommendation

As mentioned before, when it comes ANN's there are so many different parameters that can be tweaked, and for this reason there does not seem to be an optimized general solution for all cases. With that being said, it is common practice to increase the number of layers, over the number of nodes. One reason being, as seen in the study above it is more efficient with time and if a similar study was to be conducted with a few alterations, parts of the network can be re-used.

Appendix A

Benchmark Experiment: Scikit Learn Artificial Neural Networks

	Method Name	Layers	Nodes per Layer	Processing Time	Training Set Accuracy	Test Set Accuracy
0	ANN-2-Layers-10-Nodes-per-Layer	2	10	47.415090	0.934983	0.9273
1	ANN-2-Layers-20-Nodes-per-Layer	2	20	77.890954	0.966950	0.9517
2	ANN-5-Layers-10-Nodes-per-Layer	5	10	68.044559	0.943900	0.9330
3	ANN-5-Layers-20-Nodes-per-Layer	5	20	47.426777	0.964783	0.9522

Figure 1 - ANN's using Scikit Learn

TensorFlow Results

Number of Layers	Number of Nodes	Time	Accuracy
2	10	12.41	0.909
2	20	13.87	0.902
5	10	13.3	0.879
5	20	14.644	0.844

```
Epoch: 0001 cost=9.633348904
Epoch: 0002 cost=0.761321889
Epoch: 0003 cost=0.617183418
Epoch: 0004 cost=0.556968375
Epoch: 0005 cost=0.500605868
Epoch: 0006 cost=0.470640866
Epoch: 0007 cost=0.450197609
Epoch: 0008 cost=0.414479060
Epoch: 0009 cost=0.397536228
Epoch: 0010 cost=0.381756193
Epoch: 0011 cost=0.370890546
Epoch: 0012 cost=0.354351870
Epoch: 0013 cost=0.347110664
Epoch: 0014 cost=0.334347414
Epoch: 0015 cost=0.324128019
Optimization Finished!
Accuracy: 0.909
Run-Time 12.411495936135907
```

Figure 2 - TF - 2 layers 20 Nodes

```
Epoch: 0001 cost=11.178631398
Epoch: 0002 cost=1.040832857
Epoch: 0003 cost=0.747507055
Epoch: 0004 cost=0.654902359
Epoch: 0005 cost=0.597206114
Epoch: 0006 cost=0.554083844
Epoch: 0007 cost=0.524316343
Epoch: 0008 cost=0.480851139
Epoch: 0009 cost=0.459353968
Epoch: 0010 cost=0.443696011
Epoch: 0011 cost=0.416194199
Epoch: 0012 cost=0.400115483
Epoch: 0013 cost=0.382050519
Epoch: 0014 cost=0.373875024
Epoch: 0015 cost=0.356030320
Optimization Finished!
Accuracy: 0.9023
Run-Time 13.87509731641694
```

Figure 3 - TF - 2 layers 10 Nodes

```
Epoch: 0001 cost=63.260178695
Epoch: 0002 cost=3.757996063
Epoch: 0003 cost=2.000138554
Epoch: 0004 cost=1.351651843
Epoch: 0005 cost=1.039729709
Epoch: 0006 cost=0.791677189
Epoch: 0007 cost=0.699256705
Epoch: 0008 cost=0.632828035
Epoch: 0009 cost=0.580336414
Epoch: 0010 cost=0.528928408
Epoch: 0011 cost=0.496697944
Epoch: 0012 cost=0.470359781
Epoch: 0013 cost=0.457056732
Epoch: 0014 cost=0.434656379
Epoch: 0015 cost=0.414225865
Optimization Finished!
Accuracy: 0.8796
Run-Time 13.377550145989517
```

Figure 4 - TF - 5 layers 20 Nodes

```
Epoch: 0001 cost=1455.359750560
Epoch: 0002 cost=179.443396405
Epoch: 0003 cost=83.540002323
Epoch: 0004 cost=48.051421859
Epoch: 0005 cost=32.740688268
Epoch: 0006 cost=25.069775935
Epoch: 0007 cost=20.415140414
Epoch: 0008 cost=17.758795603
Epoch: 0009 cost=16.336436468
Epoch: 0010 cost=13.497063031
Epoch: 0011 cost=13.098206316
Epoch: 0012 cost=12.380232951
Epoch: 0013 cost=11.207362630
Epoch: 0014 cost=9.932238005
Epoch: 0015 cost=9.545390205
Optimization Finished!
Accuracy: 0.844
Run-Time 14.644018636619876
```

Figure 5 - TF - 5 layers 10 Nodes