AUTOMATED STOCK TRADING USING NEURAL NETWORKS

MATTHEW KNOWLES

1. Mathematical Background I: Activation

We first look at the mathematical background behind the neural network. Assume we have k hidden layers, sandwiched by an input layer and an output layer. Since the aim of neural networks is to model how a brain processes information, we often refer to the connections between nodes in each layer as "synapses". Theses synapses are characterised by their weights and biases. Let $W^{(l)}$ be the matrix containing the weights between layer l and $l-1$. If layer l has n nodes, and $l-1$ has m nodes, then we have:

$$W^{(l)} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,m} \\ w_{2,1} & w_{2,2} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \cdots & w_{n,m} \end{bmatrix} \tag{1}$$

Which gives the weights of the synapses between layer l and $l-1$. Combining this with a vector of biases for the same inter-layer, $b^{(l)}$, we are almost able to calculate the values of the nodes in layer l. However before this can be done we need an activation function. In a lot of cases, as was found in [**?**], the choice of activation function doesn't really matter. The most common choice is the "sigmoid function" defined in equation 2.

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{2}$$

Finally, piecing this all together, for the layer l, we get the values of the nodes in the layer using the matrix equation 3.

$$a^{(l)} = \sigma(W^{(1)}a^{(l-1)} + \mathbf{b}^l) \tag{3}$$

Where $a^{(l-1)}$ is the activation values for the previous layer, and $\sigma()$ is applied to all values in the resulting vector.

2. Mathematical Background II: Backpropogation

The values in the network are initially random. The way the network learns is by an algorithm called "backpropogation".The idea is to define an error function:

$$E(X,\theta) = \frac{1}{2N}\sum_{i=1}^{N}(y_i' - y_i)^2.$$

The parameter θ incorperates all the weights and biases, while X is a vector of input values. The y_i' term is the approximated value by the network, and y_i is the actual value for the X vector.

Given the number of parameters in this equation, finding minima is not as simple as simply taking a derivative. The method of minimising this function is done using "gradient descent". As discussed in [**?**], there are 3 varients of this algorithm. For a learning rate α, we have the following variants.

Date: Autumn Term 2021.

- Batch Gradient Descent: $\theta = \theta - \alpha \nabla_\theta E(X, \theta)$
- Mini-batch Gradient Descent (n training examples): $\theta = \theta - \alpha \nabla_\theta E(x^{(i:i+n)}; y^{(i:i+n)}; \theta)$
- Stochastic Gradient Descent (SGD): $\theta = \theta - \alpha \nabla_\theta E(x^{(i)}; y^{(i)}; \theta)$

SGD compares to Mini-batch by performing parameter updates for each training example. It is noted that mini-batch is the most common algorithm used in training neural networks. One key issue is chosing the optimal learning rate α. Too low, and convergence will take too long, too big and convergence won't be achieved.

Finally, for the k^{th} layer, we update the weight between node i to j in layer $k+1$ via:

$$\Delta w_{ij}^k = -\alpha \frac{\partial E(X, \theta)}{\partial w_{ij}^k}. \tag{4}$$

3. Why use Neural Networks in Stock Trading?

Historically, the use of financial ratios, the Efficient Market Hypothesis and The Capital Asset Pricing Model

4. How do we use Neural Networks in Stock Trading?

barbar

5. Improvements to Automated Trading

foofoobarbar

6. Conclusion

barbarfoofoo

References

[1] P. Sibi, S. A. Jones, and P. Siddarth, "Analysis of different activation functions using back propagation neural networks," *Journal of theoretical and applied information technology*, vol. 47, no. 3, pp. 1264–1268, 2013.
[2] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.