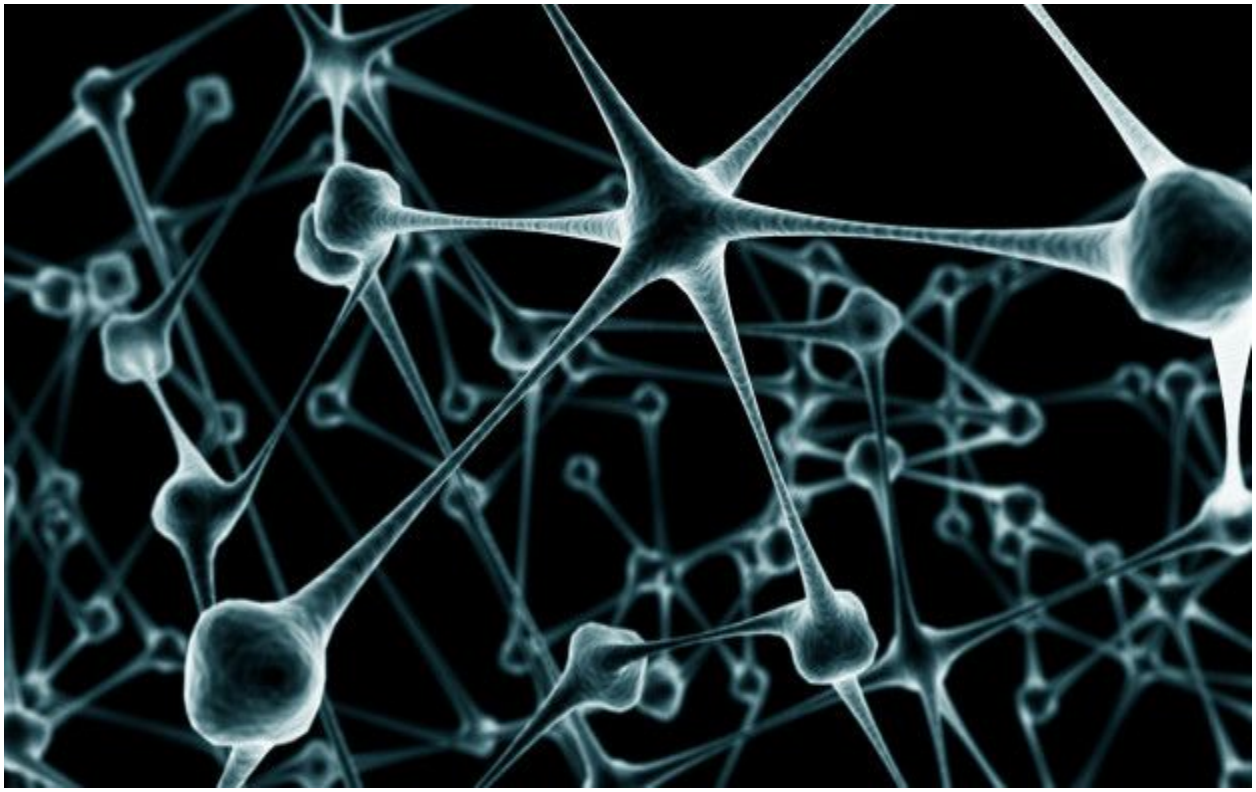
DEEP LEARNING IN OPTIONS PRICING



AISWARYA SUBRAMANIAN

DESCRIPTION OF DATA

PART I : Predicting Stock Price Movement

Our dataset, which is stored in the .csv file named 'RELIANCE.NS.csv'. The csv file contains daily OHLC data for the stock of Reliance trading on NSE for the time period from 1st January 1996 to 10th April 2019.
We then prepare the various input features which will be used by the artificial neural network to train itself for making the predictions. We define the following input features:

- High minus Low price
- Close minus Open price
- Three day moving average
- Ten day moving average
- 30 day moving average
- Standard deviation for a period of 5 days
- Relative Strength Index
- Williams %R

We then define the output value as price rise, which is a binary variable storing 1 when the closing price of tomorrow is greater than the closing price of today.

Some definitions: (from Wikipedia)

Moving Average : In statistics, a **moving average** (**rolling average** or **running average**) is a calculation to analyze data points by creating a series of averages of different subsets of the full data set.

Williams %R : **Williams %R**, or just **%R**, is a technical analysis oscillator showing the current closing price in relation to the high and low of the past N days (for a given N). It is calculated as

$$\%R = \frac{high_{Ndays} - close_{today}}{high_{Ndays} - low_{Ndays}} \times -100$$

Relative Strength Index : The relative strength index (RSI) is a technical indicator used in the analysis of financial markets. It is intended to chart the current and historical strength or weakness of a stock or market based on the closing prices of a recent trading period.

PART II : Options pricing

Training : We generate data as per black scholes formula and train the network. We are using more than 1,00,000 examples for the training purpose. 80% of the data is fed as training data and rest 20% is fed as validation data.

The dataset test.csv consists of the following fields namely, volatility, strike price, time and the current options price. There are 1530 data points. The data is taken from Scientific consultants. This data was used to test the model to analyse its performance.

ARCHITECTURE & RESULTS

I tried two models, first one was Artificial Neural Networks, and the second one was with LSTMs.

MODEL 1 : ARTIFICIAL NEURAL NETWORKS

In this model, I had tried different architectures.

A. A 3-layer network having 3 input neurons (one for volatility, one for time remaining, and one for strike), 20 middle-layer or "hidden" neurons, and one output neuron (for theoretical option price).
B. A 3-layer network with 3 input neurons, 100 second-layer neurons, 50 third-layer neurons, and 1 output neuron.
C. A 4-layer network with 3 input neurons, 30 neurons in first layer, 60 in next two layers and 1 output neuron
D. A 5-layer network with 3 input neurons, 30 neurons in first layer, 60 in next two layers, 30 in next layer and 1 output neuron
E. A 5-layer network with 3 input neurons, 50 neurons in first layer, 100 in next two layers, 50 in next layer and 1 output neuron

In all cases, the input dimension = 3 (one for volatility, one for time remaining, and one for strike) and one output neuron. The basic function of the neural network is to predict the options price, given the above 3 inputs. Hence the output domain is the set of all positive real numbers. The given neural network architecture, shall represent an approximate function which try to fit the data.

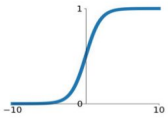
Activation Function used : In both the architectures, as the output is all positive real numbers, our activation function at the output layer is the linear function. The final layer of the neural network will have one neuron and the value it returns is a continuous numerical value.

For the hidden layers, I tried three different activation functions : sigmoid, tanh & RELU.

Activation Functions

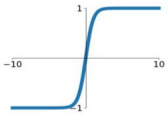
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



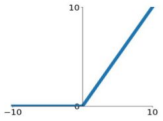
tanh

$\tanh(x)$



ReLU

$\max(0, x)$



(image courtesy : google)

Data Preprocessing : Here, as our training data has features that are of different scales.

```
[2]: data = pd.read_csv('options2.csv')
     data.head()
```

[2]:

	FCTNO	VLTY	TIME	STRIKE	OPRICE
0	1	0.2	5.0	75.0	25.0
1	2	0.2	5.0	76.0	24.0
2	3	0.2	5.0	77.0	23.0
3	4	0.2	5.0	78.0	22.0
4	5	0.2	5.0	79.0	21.0

We can clearly see that the 'VLTY' are relatively very small numbers when compared to 'STRIKE'. Hence we need to do something known as "Feature Scaling".

Feature Scaling or Standardization*: It is a step of Data Pre Processing which is applied to independent variables or features of data. It basically helps to normalise the data within a particular range.* -- GeeksforGeeks

Once we have scaled our variables, it looks like :

[46]:

	FCTNO	VLTY	TIME	STRIKE	OPRICE
0	-1.000000	-0.9	-0.5	-0.961538	1.115762
1	-0.998692	-0.9	-0.5	-0.923077	1.042261
2	-0.997384	-0.9	-0.5	-0.884615	0.968760
3	-0.996076	-0.9	-0.5	-0.846154	0.895259
4	-0.994768	-0.9	-0.5	-0.807692	0.821758

Next we try to normalise the data. Normalization is an example of preprocessing data to remove or reduce the burden from machine learning (ML) to learn certain *invariants*, that is, things which make no difference in the meaning of the symbol, but only change the representation.

Once we normalise the data, it looks like :

[51]:

	FCTNO	VLTY	TIME	STRIKE	OPRICE
0	-0.486246	-0.437622	-0.243123	-0.467545	0.542535
1	-0.499611	-0.450238	-0.250132	-0.461783	0.521407
2	-0.513283	-0.463166	-0.257315	-0.455249	0.498553
3	-0.527206	-0.476355	-0.264642	-0.447855	0.473846
4	-0.541305	-0.489737	-0.272076	-0.439508	0.447162

Now, the preprocessing of the data has been done, and can be used for training the model. As per the above architecture, using MSE as loss function, we train the model.

Loss Function : To understand the accuracy of the prediction, it is compared with the true value which is also a continuous number. As a regression type of problem, we use

Mean Squared Error : this finds the average squared difference between the predicted value and the true value.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where $\hat{y}$ is the predicted value and y is the true value

Image Courtesy : Medium

Optimizer : In this model, I used different kinds of optimizers namely : Momentum, Adadelta, ADAgrad, RMSprop, Adam and SGD.

The ADAgrad optimizer essentially uses a different learning rate for every parameter and every time step. The reasoning behind ADAgrad is that the parameters that are infrequent must have larger learning rates while parameters that are frequent must have smaller learning rates.

RMSprop considers fixing the diminishing learning rate by only using a certain number of previous gradients.

Adaptive Moment Estimation, or Adam, computes the adaptive learning rates for each parameter by considering the exponentially decaying average of past squared gradients and the exponentially decaying average of past gradients.

Gradient Descent calculates gradient for the whole dataset and updates values in direction opposite to the gradients until we find a local minima. Stochastic Gradient Descent performs a parameter update for each training example unlike normal Gradient Descent which performs only one update. Thus it is much faster.

Name	Update Rule
SGD	$\Delta\theta_t = -\alpha g_t$
Momentum	$m_t = \gamma m_{t-1} + (1-\gamma) g_t,$ $\Delta\theta_t = -\alpha\, m_t$
Adagrad	$G_t = G_{t-1} + g_t^2,$ $\Delta\theta_t = -\alpha\, g_t G_t^{-1/2}$
Adadelta	$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2,$ $\Delta\theta_t = -\alpha g_t v_t^{-1/2} D_{t-1}^{1/2},$ $D_t = \beta_1 D_{t-1} + (1-\beta_1)(\Delta\theta_t/\alpha)^2$
RMSprop	$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2,$ $\Delta\theta_t = -\alpha\, g_t v_t^{-1/2}$
Adam	$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t,$ $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2,$ $\hat{m}_t = m_t/(1-\beta_1^t),$ $\hat{v}_t = v_t/(1-\beta_2^t),$ $\Delta\theta_t = -\alpha\, \hat{m}_t \hat{v}_t^{-1/2}$

(Image courtesy : google)

MODEL 2: LONG-SHORT TERM MEMORY

In this model, I have tried an architecture with 100 lstm cells in each LSTM layer, and our LSTM model is composed of a sequential input layer followed by 3 LSTM layers and dense layer with activation and then finally a dense output layer with linear activation function. We have a batch of 100 samples, each sample is a sequence of TIME STEP = 5. Hence we have 1520 sequences to train, with 75% as training data and 25% as test data. I took ideas from this source.

Data Preprocessing: The pre-processing stage involves :

a) Data discretization: Part of data reduction but with particular importance, especially for numerical data

b) Data transformation: Normalization.

c) Data cleaning: Fill in missing values.

d) Data integration: Integration of data files. After the dataset is transformed into a clean dataset, the dataset is divided into training and testing sets so as to evaluate.

Optimizer : The type of optimizer used can greatly affect how fast the algorithm converges to the minimum value. Also, it is important that there is some notion of randomness to avoid getting stuck in a local minimum and not reach the global minimum.

I have chosen to use Adam optimizer, which combines the perks of two other optimizers: ADAgrad and RMSprop. Adaptive Moment Estimation, or Adam, computes the adaptive learning rates for each parameter by considering the exponentially decaying average of past squared gradients and the exponentially decaying average of past gradients.

Regularization: To make sure the weights do not get too large and start focusing on one data point, hence overfitting. I have used Tikhonov regularization.

Dropout : This forces the model to not be over dependent on any groups of neurons, and consider all of them. Dropouts have found their use in making the neurons more robust and hence allowing them to predict the trend without focusing on any one neuron.

PERFORMANCE MEASUREMENT

The pricing performance of different models are quantified based on some error measurement parameters. In most of the studies mean squared error is used as a performance measurement criteria and is more intuitive. MSE is the workhorse of basic loss functions: it's easy to understand and implement and generally works pretty well. To calculate MSE, we take the difference between our predictions and the ground truth, square it, and average it out across the whole dataset. In the next section, I have shown the comparison of different results. But, the results are shown only for one 3, 4 & 5 layer networks.

Comparison of the different results (only for ANN models):

In MODEL 1, all layers are given same activation functions. The table representation shall help us see which combination gives the maximum accuracy for a particular model. Each cell in the table gives the accuracy (in percentage) obtained using that (Activation function, Optimizer) pair. In all cases, the output layer has linear activation function.

One important thing is that the accuracy values can change in different trials, the table gives only one trial instance (and I assume that the accuracy may not change from given values drastically).

In the model, I used 80-20 split for training-validation data. The metric used in training is custom made as shown below:

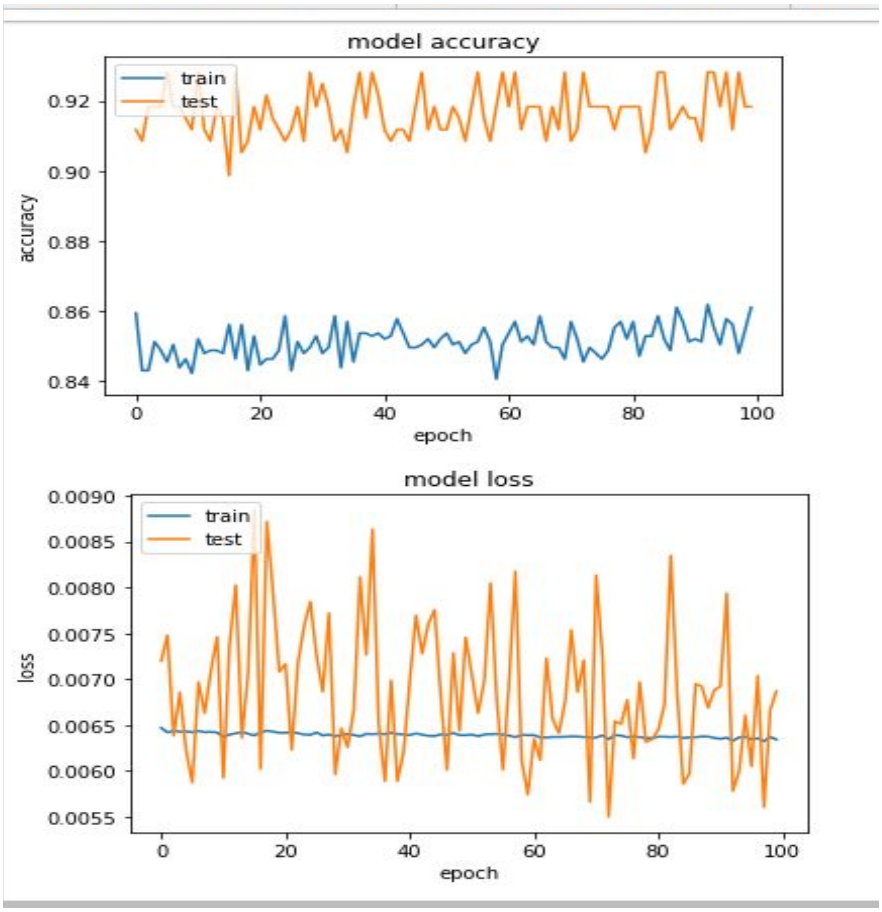
```
import tensorflow as tf
accepted_diff = 0.01
def linear_regression_equality(y_true, y_pred):
      diff = K.abs(y_true-y_pred)
      return K.mean(K.cast(diff < accepted_diff, tf.float32))
```

MODEL 1.A :

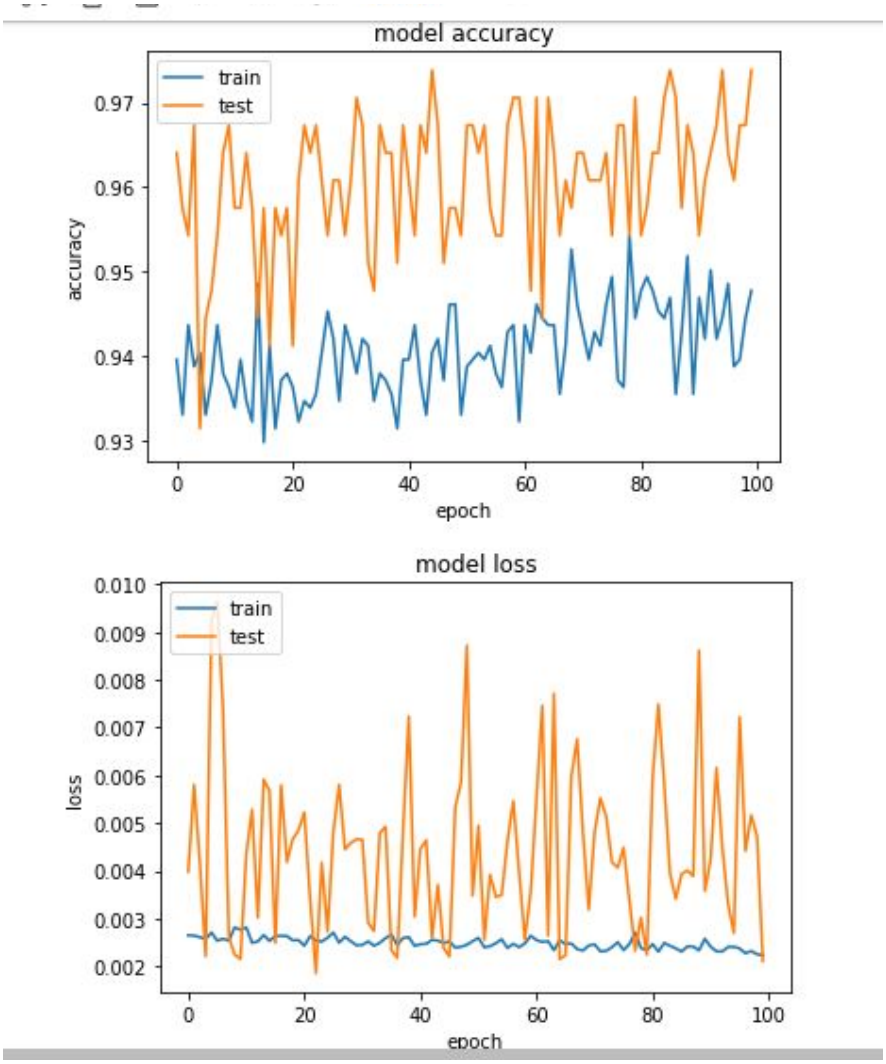
	Adam	Adadelta	Adagrad	Adamax	RMSprop	SGD	Accuracy(max)
sigmoid	79.98	81.94	81.70	81.78	84.07	86.11	86.11
tanh	94.77	92.73	94.20	93.79	89.22	81.13	94.77
RELU	94.28	96.90	97.55	97.30	95.83	97.39	97.55

The train-validation accuracy-loss plots for the maximum accuracy cases only:

Act fun: SIGMOID, Opt : SGD Accuracy Achieved : 86.11

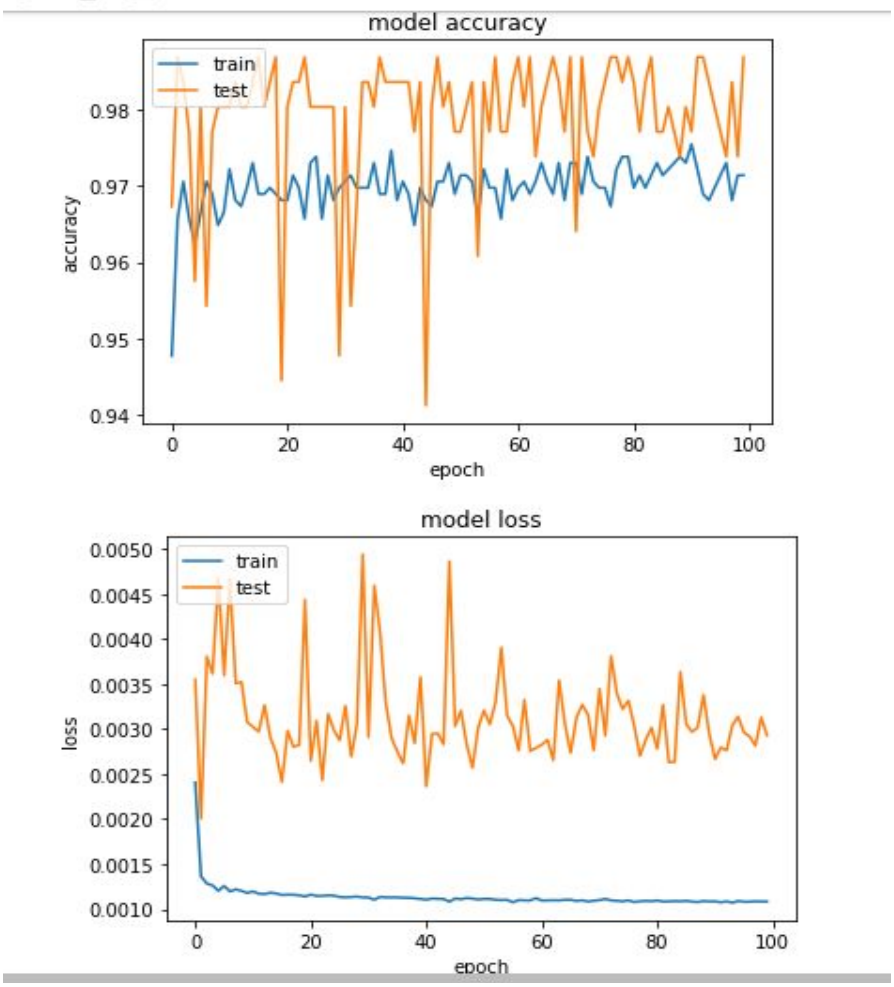


Act fun : TANH Opt : ADAM Accuracy Achieved : 94.77



Act fun : RELU, Opt : Adagrad

Accuracy Achieved : 97.55

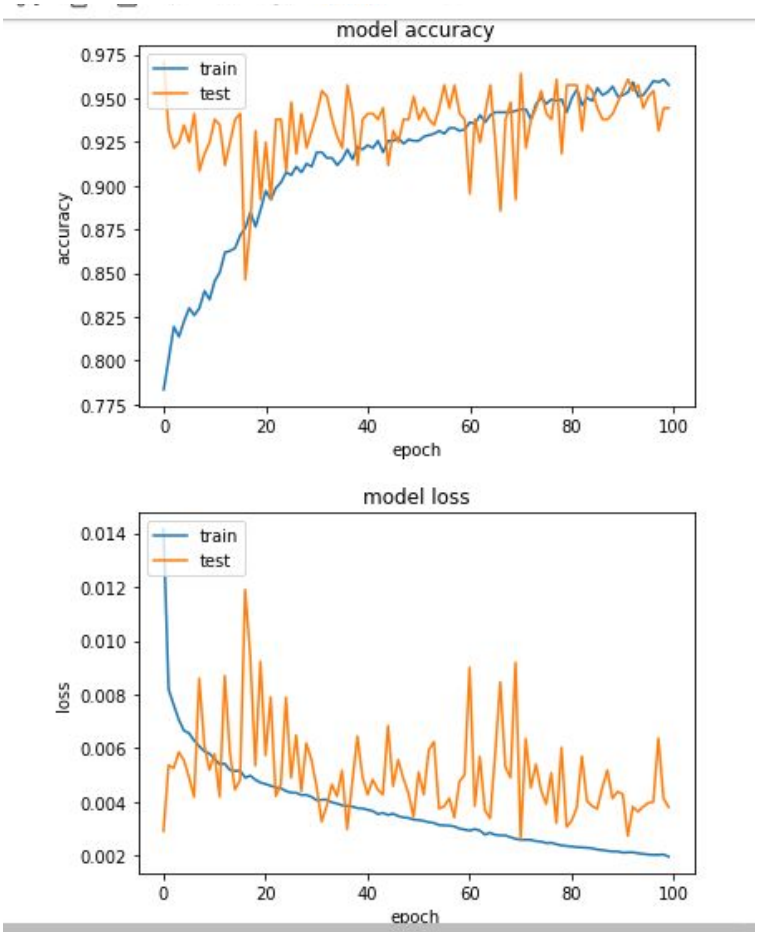


MODEL 1.C :

	Adam	Adagrad	Adadelta	Adamax	RMSprop	SGD	Accuracy (max)
sigmoid	81.62	84.48	85.38	89.30	78.84	75.90	89.30
tanh	91.83	95.51	92.40	93.87	92.08	96.57	96.57
RELU	96.08	98.6	95.67	96.16	95.34	87.01	98.6

Act Fun: RELU, Opt : Adagrad

Accuracy Achieved : 98.6

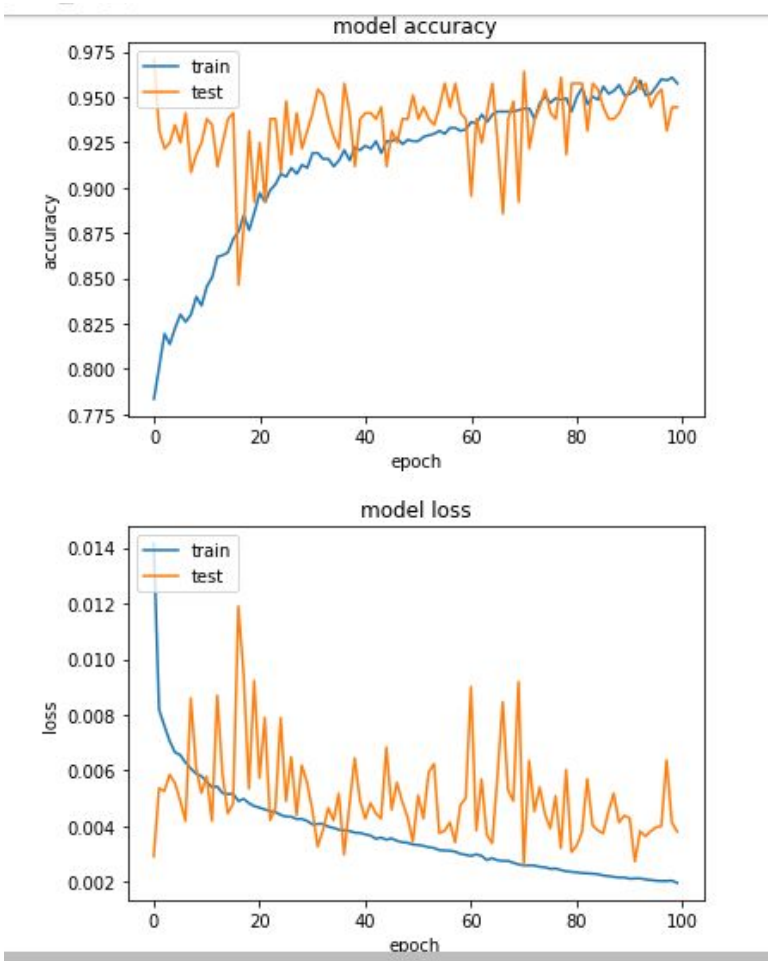


MODEL 1.D :

	Adam	Adagrad	Adadelta	Adamax	RMSprop	SGD	Accuracy (max)
sigmoid	80.72	75.98	78.27	79.25	77.94	56.29	80.72
tanh	91.83	90.60	93.55	94.69	92.73	81.21	94.69
RELU	95.92	97.63	96.49	96.00	96.16	84.80	97.63

Act Fun: RELU, Opt : Adagrad

Accuracy achieved : 97.63



OBSERVATIONS

Firstly, in the above results, we can observe that RELU activation out performs in all three cases. This might be because, RELU is finally linear for positive values. Hence we can conclude that this problem is linear, i.e. it has a linear decision boundary (the regression boundary)

Secondly, ADAM and ADAgrad optimizers give the best results compared to all other optimizers used. Hence, an adaptive optimizer can be preferred over constant rate optimizers like SGD.

Thirdly, we can see that, the maximum accuracy reached is 98.6%. This shows that artificial neural networks are indeed able to approximate a function to fit the given data. Also, if the number of neurons in each layer is increased, or even by increasing the number of layers, the model performs better.

CONCLUSION & FUTURE WORK

It has been more than 40 years since the famous option pricing formula was developed by Black and Scholes. Many improvements to their formula have been done by both the academic researchers and the practitioners since then. One of the critical point for the option pricing theory came with the release of the article with a new approach taken by Hutchinson et al. (1994) who used a neural network method to predict the option price.

In this project, I have tried to analyse the applications of Deep Learning techniques namely **Artificial Neural Networks and LSTM in options pricing**. It is intuitive that, in most cases the neural networks will have better prediction mechanism than the closed-form Black-Scholes formula, since it is very important to incorporate the data into the model, so that the temporal nature of the data is captured, in order to find out the true price of options.

Further research is suggested for experimenting with different input variables that could for example improve the approximation of the network as well as experimenting with different types of options data. Additionally, research in the future might include different types of neural networks such as Convolutional Neural Networks (CNNs) and Spiking Neural Networks (SNNs) into the comparison. The neural network models should also be compared to other types of parametric formulae, such as the ones by Merton (1976) and Heston (1993). Similarly, neural networks could be researched in areas where there are no analytical solutions to compute the price of an option.

BIBLIOGRAPHY

https://www.researchgate.net/publication/220204936_A_neural_network_model_for_estimating_option_prices

https://core.ac.uk/download/pdf/39663999.pdf

https://blog.usejournal.com/stock-market-prediction-by-recurrent-neural-network-on-lstm-model-56de700bff68

https://towardsdatascience.com/deep-learning-which-loss-and-activation-functions-should-i-use-ac02f1c56aa8

https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f

https://medium.com/data-science-group-iitr/loss-functions-and-optimization-algorithms-demystified-bb92daff331c

https://www.researchgate.net/publication/225165990_Comparison_of_new_activation_functions_in_neural_network_for_forecasting_financial_time_series

http://jultika.oulu.fi/files/nbnfioulu-201901091016.pdf

https://stackoverflow.com/questions/42665359/how-do-you-compute-accuracy-in-a-regression-model-after-rounding-predictions-to (for custom made metric in keras)

https://stackoverflow.com/questions/50797135/keras-network-acc-zero-and-loss-very-low

- (the reason for need for custom made metric for learning)

https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/

https://machinelearningmastery.com/custom-metrics-deep-learning-keras-python/

http://www.scientific-consultants.com/nnbd.html (DATA)

https://www.youtube.com/watch?v=ca7oC70BnTg

https://www.khanacademy.org/economics-finance-domain/core-finance/derivative-securities/black-scholes/v/introduction-to-the-black-scholes-formula

https://www.cse.iitb.ac.in/~saketh/research/DeodaDDP.pdf

https://www.asx.com.au/prices/pricing_models.htm

http://www.cboe.com/products/vix-index-volatility/volatility-on-stock-indexes/cboe-nasdaq-100-volatility-index-vxn

https://www.investopedia.com/articles/optioninvestor/06/newvix.asp

https://www.edupristine.com/blog/financial-instruments

https://www.investopedia.com/articles/optioninvestor/05/020205.asp

https://www.investopedia.com/articles/optioninvestor/10/etf-options-v-index-options.asp

https://www.thebalance.com/trading-stock-indexes-1031061

https://www.cse.iitb.ac.in/~saketh/research/DeodaDDP.pdf

https://pure.bond.edu.au/ws/portalfiles/portal/18243185/Option_Pricing_Using_Artificial_Neural_Networks.pdf

https://helda.helsinki.fi/dhanken/bitstream/handle/123456789/202968/huynh.pdf?sequence=1&isAllowed=y

https://www.weareworldquant.com/en/thought-leadership/beyond-black-scholes-a-new-option-for-options-pricing/

https://medium.com/@ranko.mosic/option-pricing-and-volatility-forecasting-using-deep-learning-4a57ebb1f39