Options Pricing using Deep Learning

A Presentation by
Visaj Nirav Shah (201801016)
Dhruvil Bhatt (201801056)

Project Objective

- Predict options prices using two approaches:-
  - **Model I: Deep Learning (DL)**-based
  - **Model II: Black-Scholes-Merton (BSM)**

- Compare the performance of Model I and Model II using a suitable metric

Project Output

- A large dataset with options prices and other required variables for both the models
- Implementation of a BSM price prediction model for Call and Put options
- An efficient DL model built for options price prediction, each for Call and Put options
- A comparison between the performance of the Model I and Model II

Finance 101

What are options?

- Financial contract that allows the owner to buy/sell an underlying asset (like a stock) at a fixed rate (strike price) on or before the expiry date

- Call and Put options

- European and American options

Some common symbols:-

- r: Risk-free interest rate
- σ: Volatility of the underlying asset
- t: Time period

Dataset

Preparation

- Lack of existing quality datasets for Indian financial markets
- Build our dataset:
  - NSE Website
  - RBI Website for r
  - Mathematical computation for σ
- Significance of this dataset

Options details

- Asian Paints Ltd. (ASIANPAINT) options
- Timeframe: 2017, 2018, 2019, 2020
- European options
- 111,866 Call options
- 111,866 Put options

Dataset

- Volatility (σ) calculation
  - Historical values of last 20 trading days were considered
- 91-Day Treasury Bill (Primary) Yield is considered for r
  - Declared by RBI on a weekly basis
- Close Price on a particular trading day is considered to be the price of option for that day
- t = (option expiry date) - (option bought date)

Options Pricing Techniques

Traditional models/methods:-

- BSM model
- Binomial Pricing model
- Monte Carlo simulations

Such models might not be able to detect patterns in price variation, and determine factors affecting it.

Recent advancements:-

- Increase in data available (historical + current)
- Rise of Machine Learning (ML) and DL
- Use of neural networks in finance

As computational capabilities increase, our capacity of working with large datasets increases.

- Beneficial for ML/DL models

Black-Scholes Model

Model I

- What is Black-Scholes Model (BSM)?
- Different parameters of BSM
- Formula of BSM
- Implementation
- Results

What is Black-Scholes Model (BSM)?

- The Black-Scholes model, also known as the Black-Scholes-Merton (BSM) model, is one of the most important concepts in modern financial theory. This mathematical equation estimates the theoretical value of derivatives other investment instruments, taking into account the impact of time and other risk factors.

- The standard BSM model is only used to price European options, as it does not take into account that American options could be exercised before the expiration date.

Different parameters essential for BSM

- Black-Scholes posits that instruments, such as stock shares or futures contracts, will have a lognormal distribution of prices following a random walk with constant drift and volatility. Using this assumption and factoring in other important variables, the equation derives the price of a European-style call option.
- The Black-Scholes equation requires five variables. These inputs are volatility, the price of the underlying asset, the strike price of the option, the time until expiration of the option, and the risk-free interest rate.
- N = CDF of Normal Distribution
- S_t = Current price of the asset
- K = Strike Price of the asset
- r = Risk-free interest rate
- t = Time to maturity
- σ = Volatility of the asset

BSM Formula

The Black-Scholes call option formula is calculated by multiplying the stock price by the cumulative standard normal probability distribution function. Thereafter, the net present value (NPV) of the strike price multiplied by the cumulative standard normal distribution is subtracted from the resulting value of the previous calculation.

$$C = N(d_1)S_t - N(d_2)Ke^{-rt}$$

$$P = N(-d_2)Ke^{-rt} - N(-d_1)S_t$$

where $d_1 = (\ln(S_t/K) + (r + \sigma^2/2)t)/\sigma\sqrt{t}$

$d_2 = d_1 - \sigma\sqrt{t}$

C = Call option Price

P = Put Option Price

Actual Implementation

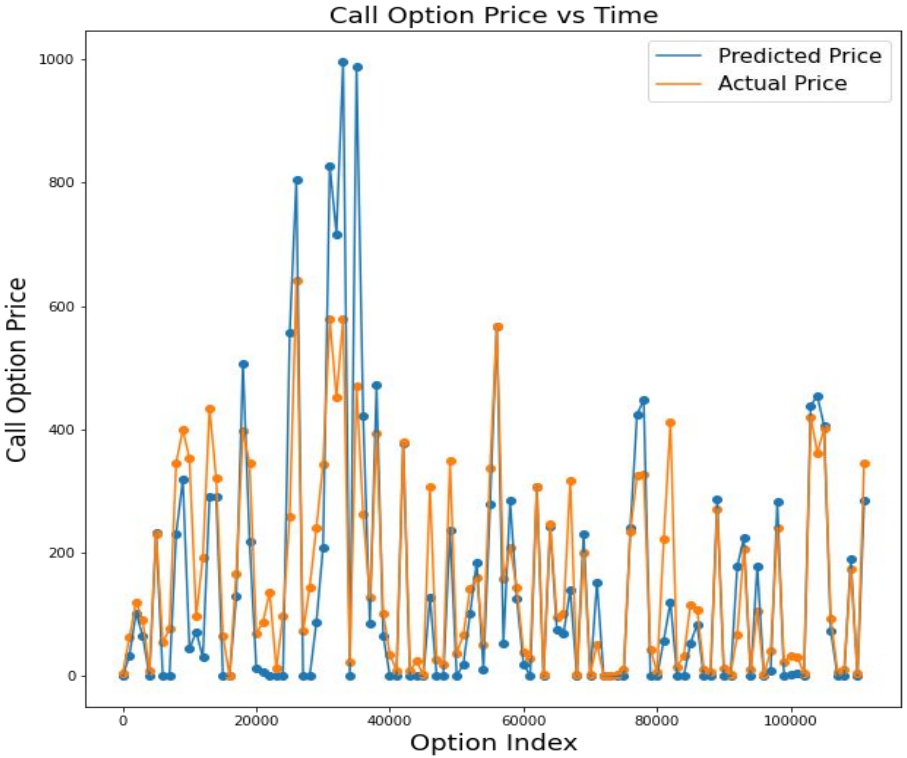
Call Option

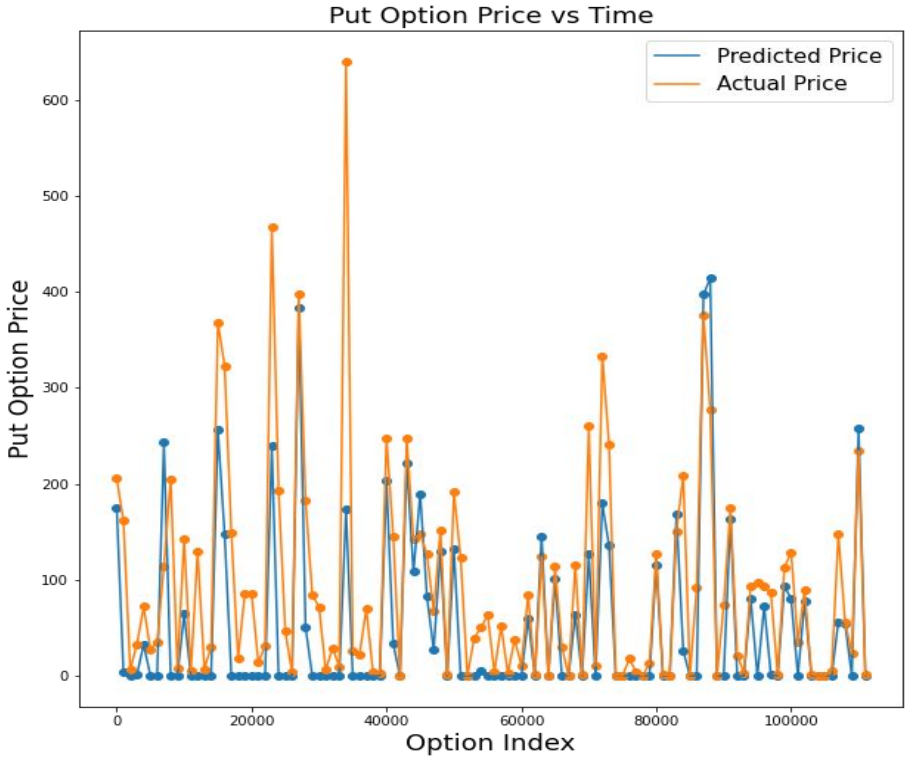
```
def d1(S, K, T, r, sigma):
    return (np.log(S/K)+(r+sigma**2/2)*T)/sigma*sqrt(T)

def d2(S,K,T,r,sigma):
    return d1(S,K,T,r,sigma)-sigma*sqrt(T)

def black_scholes_call(S,K,T,r,sigma):
    return S*norm.cdf(d1(S,K,T,r,sigma))-K*exp(-r*T)*norm.cdf(d2(S,K,T,r,sigma))
```

Put Option

```
def d1(S, K, T, r, sigma):
    return (np.log(S/K)+(r+sigma**2/2)*T)/sigma*sqrt(T)

def d2(S,K,T,r,sigma):
    return d1(S,K,T,r,sigma)-sigma*sqrt(T)

def black_scholes_put(S,K,T,r,sigma):
    return K*exp(-r*T)*(1 - norm.cdf(d2(S,K,T,r,sigma))) - S*(1 - norm.cdf(d1(S,K,T,r,sigma)))
```

Results


Call Option Price vs Time
Predicted Price
Actual Price
Call Option Price
Option Index
0
200
400
600
800
1000
0
20000
40000
60000
80000
100000


Put Option Price vs Time
Predicted Price
Actual Price
Put Option Price
Option Index
0
100
200
300
400
500
600
0
20000
40000
60000
80000
100000

Deep Learning Model

Model II

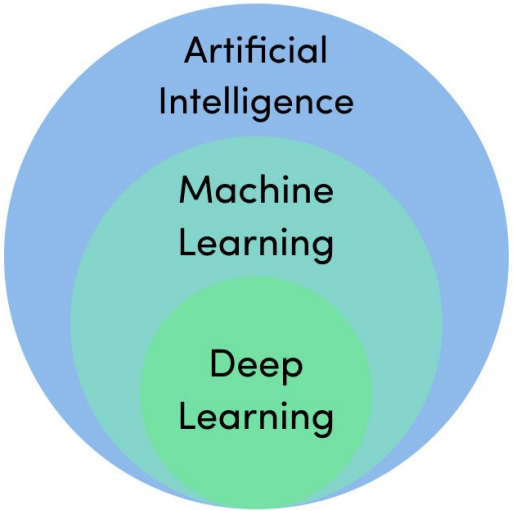
- What is Deep Learning?
- Model Architecture
- DL Basics and Terminology
- Normalization
- Implementation
- Results

What is Deep Learning?

- Deep Learning is a subset of Machine Learning

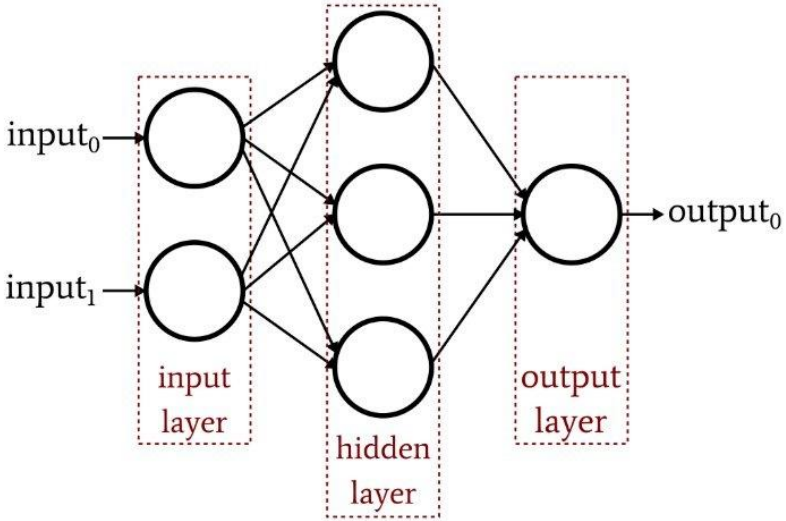
- Neural networks
  - Inspired by the “neuron network” of our brain
  - Multiple layers

- Rapid advancements in specialized learning tasks
  - Natural Language Processing
  - Computer Vision, Image Recognition
  - Text-to-Speech



Model Architecture and Construction

- Multi-Layer Perceptron
  - Sequential, fully connected neural network
- Call options model
  - 1 input layer + 3 hidden layers + 1 output layer
  - 400 neurons in each hidden layer
- Put options model
  - 1 input layer + 4 hidden layers + 1 output layer
  - 400 neurons in each hidden layer



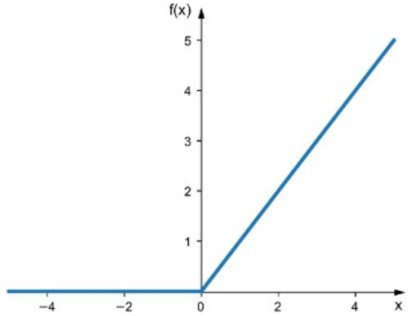
MLP network with 1 hidden layer

DL Basics and Terminology

ReLU Activation

- Rectified Linear Unit
- y = max(0, x)
- Variations
    - LeakyRelu, PReLU, ELU, CReLU

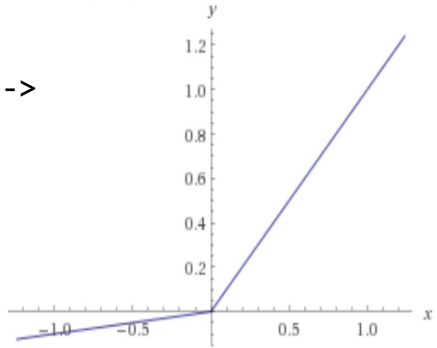
ReLU ->



LeakyReLU Activation

- Dying ReLU
- y = ax when x < 0, max(0, x) when x >= 0

LeakyReLU ->



DL Basics and Terminology

Test Train Split

- We split the dataset into training and testing datasets
- 80-20 split

Validation set

- Used to test model performance while in training
  - Helps in tuning model hyperparameters
- 1% of training set is used as validation set

Batch size

- It is difficult to process the entire data at once
- After each batch is processed, the network gets a chance to update the internal weights.
- Batch size = 128

Epoch

- 1 epoch is equivalent to all the batches passing through the network once
  - Analogous to iteration
- The model reduces the error with each passing epoch
- Max. epochs = 30

Callbacks

EarlyStopping

- There is no definite way to determine the ideal number of epochs
- If the monitored quantity is not moving in required direction by a significant value, we stop

```
[ ]  # EarlyStopping Callback

     cb_put = tf.keras.callbacks.EarlyStopping(\
         monitor = 'val_loss',\
         patience = 5,\
         restore_best_weights = True)
```

ReduceLROnPlateau

- Not possible to manually control the learning rate (LR) during execution
- As the loss approaches the gradient descent minima, we start reducing the LR so that we do not miss the minima.

```
[ ]  # Learning Rate Callback

     LrCb_put = tf.keras.callbacks.ReduceLROnPlateau(\
         monitor = 'val_loss',\
         factor = 0.2,\
         patience = 3)
```

Normalization

- Bring all inputs to a common scale
  - Graph shape remain same
  - Output variables remain unaffected
    - No need to rescale
  - We are mapping to [0, 1]

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

- Why normalization?
  - Numerically larger input variables have a greater impact on predicting output

- Batch Normalization
  - Output of a layer which is not normalized is the input of the next layer
  - This input is normalized by Batch Normalization for next layer

```
[4]  df_call.describe()
```

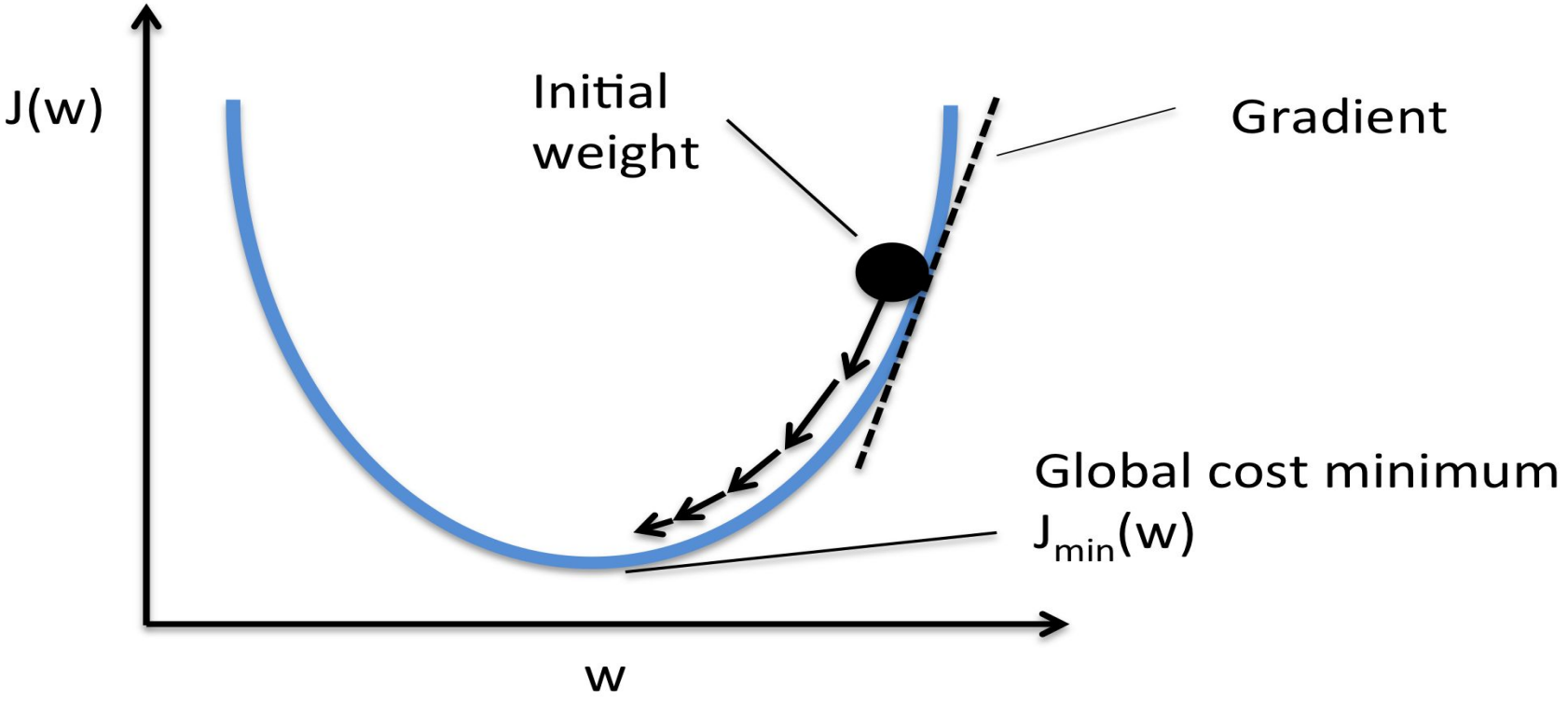
	t	strike_price	underlying_value	sigma	r	close
count	111866.000000	111866.000000	111866.000000	111866.000000	111866.000000	111866.000000
mean	43.752159	1430.032360	1515.434932	0.015271	0.053584	154.421568
std	26.443382	413.833394	396.529738	0.006671	0.013361	147.074145
min	0.000000	660.000000	896.600000	0.006076	0.029300	0.050000
25%	21.000000	1120.000000	1174.150000	0.011451	0.041500	25.800000
50%	43.000000	1380.000000	1401.950000	0.013581	0.061100	113.250000
75%	66.000000	1660.000000	1770.950000	0.017459	0.063100	255.300000
max	97.000000	3140.000000	2764.500000	0.052411	0.071900	1360.000000

Summary of Call Options dataset: We observe the widely varying ranges of input variables

Implementation

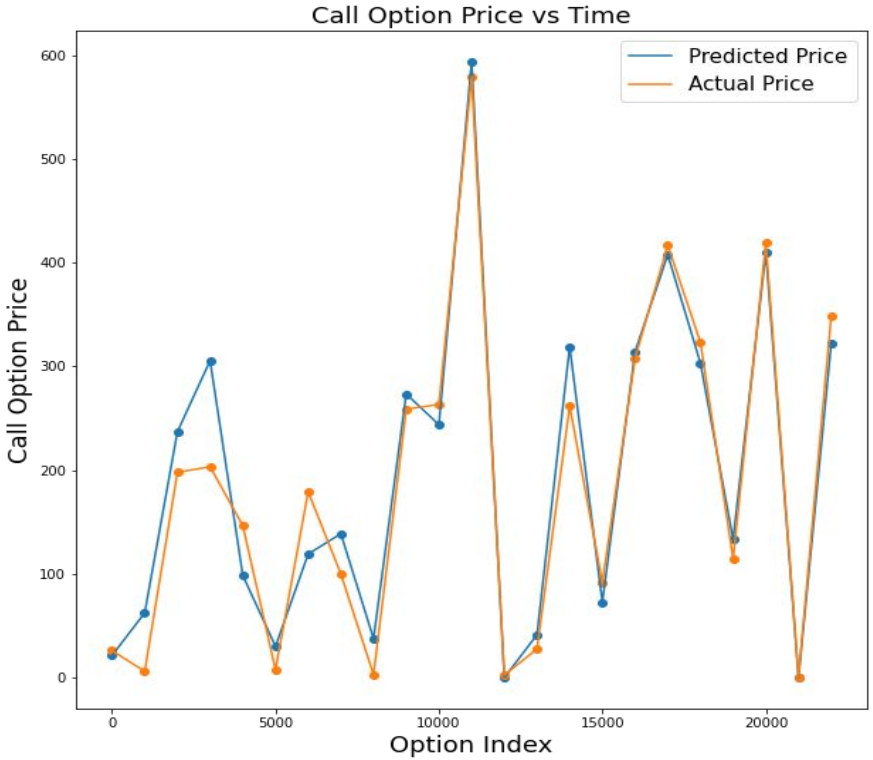
- Sequential and Dense classes
  - To create MLP
- Adam optimizer
  - Similar to gradient descent
- MSE as loss function

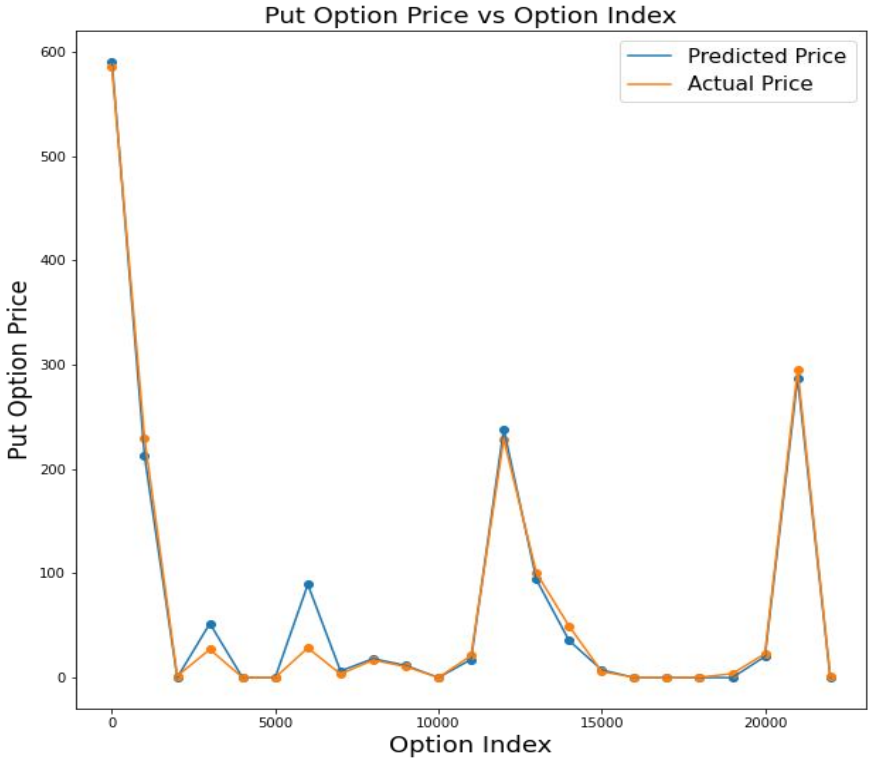
```
[ ]  # Deep Learning model

     model = Sequential()
     model.add(Dense(n_units, input_dim = X_train.shape[1]))
     model.add(LeakyReLU())

     for i in range(layers - 1):
         model.add(Dense(n_units))
         model.add(BatchNormalization())
         model.add(LeakyReLU())

     model.add(Dense(1, activation = 'relu'))

     model.compile(loss = 'mse', optimizer = Adam())

     model.summary()
```



Gradient Descent

Results


Call Option Price vs Time
Predicted Price
Actual Price
Call Option Price
Option Index
0
100
200
300
400
500
600
0
5000
10000
15000
20000


Put Option Price vs Option Index
Predicted Price
Actual Price
Put Option Price
Option Index
0
100
200
300
400
500
600
0
5000
10000
15000
20000

Comparison

- Metric for Comparison
- Comparison of all the models
- Why Deep Learning Performed better?

Metric For Comparison

- For comparison, we have chosen Mean Squared Error (MSE) as the metric.

- Since our problem statement is a “Regression Problem”, it is preferable to have MSE as the metric for comparison rather than “Accuracy”, which is more suitable for a “Classification Problem”.

Comparison Of All The Models Implemented

	MSE for Call Option	MSE for Put Option
Black-Scholes Model	10208.14668548929	8537.405058233906
Deep Learning Un-Normalized	2874.894010878188	2507.2077137547585
Deep Learning Normalized	1051.1803597885755	1060.6603938529304

	% Improvement w.r.t BSM (Call Option)	% Improvement w.r.t BSM (Put Option)
Deep Learning Un-Normalized	71.83725803	70.63267238
Deep Learning Normalized	89.70253473	87.5763140366

Why Deep Learning Performed Significantly Better?

- Black-Scholes Model starts from assuming a specific form of movement for the stock price, namely a geometric Brownian motion (GBM).
- The BSM delivers a formula that is used to price call and put options and is the solution to a special partial differential equation (PDE).
- Complex non-linearities in the option prices are not captured via a closed form PDE solution as given by BSM.
- However, Deep Learning techniques do not have any assumption that defines the movement of the stock prices.
- Non-linearities are very well learnt using Deep Learning architectures that have several neurons in each layer to induce a complex mathematical expression for each layer's output and hence could capture a highly skewed option price dataset as well.

Conclusion

- Deep Learning gives significantly better prediction performance than the traditional Black-Scholes Model.

- Deep Learning can extract patterns in the option price dataset, that BSM cannot capture.
  - Feature extraction

- Optimum choice of parameters for Deep Learning architectures is of vital importance.
  - Very high number of neutrons tend to overfit the training dataset.

Future Work

- An even larger dataset can be much useful for training DL models
  - More the data, better the predictions
- Determine and include other factors that affect options pricing
  - Right now, we are using the same parameters as the BSM model for DL models
- DL model implementation for other options
  - Exotic options
  - American options

References

- R. Culkin and S. R. Das, “Machine learning in finance : The case of deep learning for option pricing,” 2017.
- A. Ke and A. Yang, “Option pricing with deep learning,” 2019.
- S. Liu, C. W. Oosterlee, and S. M.Bohte, “Pricing options and computing
- implied volatilities using neural networks,” 2019.
- Black-Scholes-Merton Model, Corporate Finance Institute

Thank you

A Presentation by
Visaj Nirav Shah (201801016)
Dhruvil Bhatt (201801056)