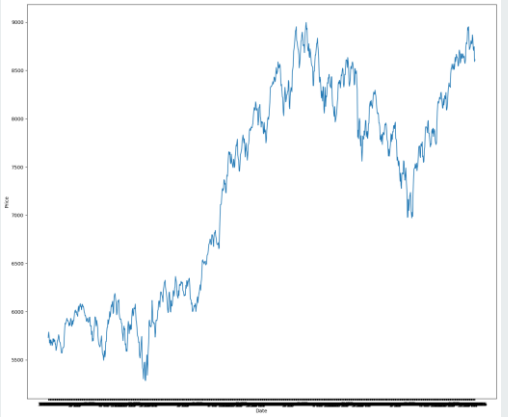
Predicting Stock Prices Using Big-Data Tools and Deep Learning

Presentation By:
Rajan Bajaj

Proposition

- Prediction of rise and fall of stock prices is very difficult task for many analysts and researches.
- Investors and traders looks for better prediction methods.
- In this project a combined technique of Big Data and Deep Learning is used to get fast and accurate prediction on Nifty 50 Data.
- Recurrent neural networks(RNN) and Long Short-Term Memory(LSTM) approach which are part of Deep Learning field and to increase the training rate the process of training will be distributed on a cluster of computers using Apache Spark.



Nifty 50 Daily Closing Price Data

Introduction

- The temporal relation of historical data to future values prediction is found using a Long Short-term Memory approach. The data is divided in data-frames and fed to the model for training it in batches.
- Training Deep Neural network is time consuming process. For this reason taking advantage of cluster to speed up training is important area of work.
- For much difficult problems such as computation on ImageNet dataset training can take much longer time on a single GPU and hence we can take advantage of Distributed platform for training Deep Learning models.
- Apache Spark for data parallelization as well as model parallelization is used.
- Model parallelization is partitioning network over different machines.

Advantages

- Fast and scalable training and prediction which is very important in stock market analysis for optimization of profit.
- GPUs are not required for heavy computation as model is also distributed over many machines.
- Low cost approach.

Different Works

- DistBelief from Google
- Elephas: Distributed Deep Learning with Keras & Spark*
- Project Adams from Microsoft

are both distributed frameworks meant for training large scale models for deep networks over thousands of machines and utilizing both data and model parallelism.

Elephas: Distributed Deep Learning with Keras & Spark

This framework provides:

- Data-Parallel training of deep learning models
- Distributed hyper-parameter optimization
- Distributed training of ensemble models
- Basic Spark Integration using pyspark
- Spark MLlib Integration

Nifty 50 Dataset

Sample Data:

	Date	Price	Open	High	Low	Vol.	Change %
0	3-Oct-12	5,731.25	5,727.70	5,743.25	5,715.80	173.22M	0.22%
1	4-Oct-12	5,787.60	5,751.55	5,807.25	5,751.35	179.19M	0.98%
2	5-Oct-12	5,746.95	5,815.00	5,815.35	4,888.20	255.57M	-0.70%
3	8-Oct-12	5,676.00	5,751.85	5,751.85	5,666.20	142.32M	-1.23%
4	9-Oct-12	5,704.60	5,708.15	5,728.65	5,677.90	119.30M	0.50%

Nifty 50 Dataset

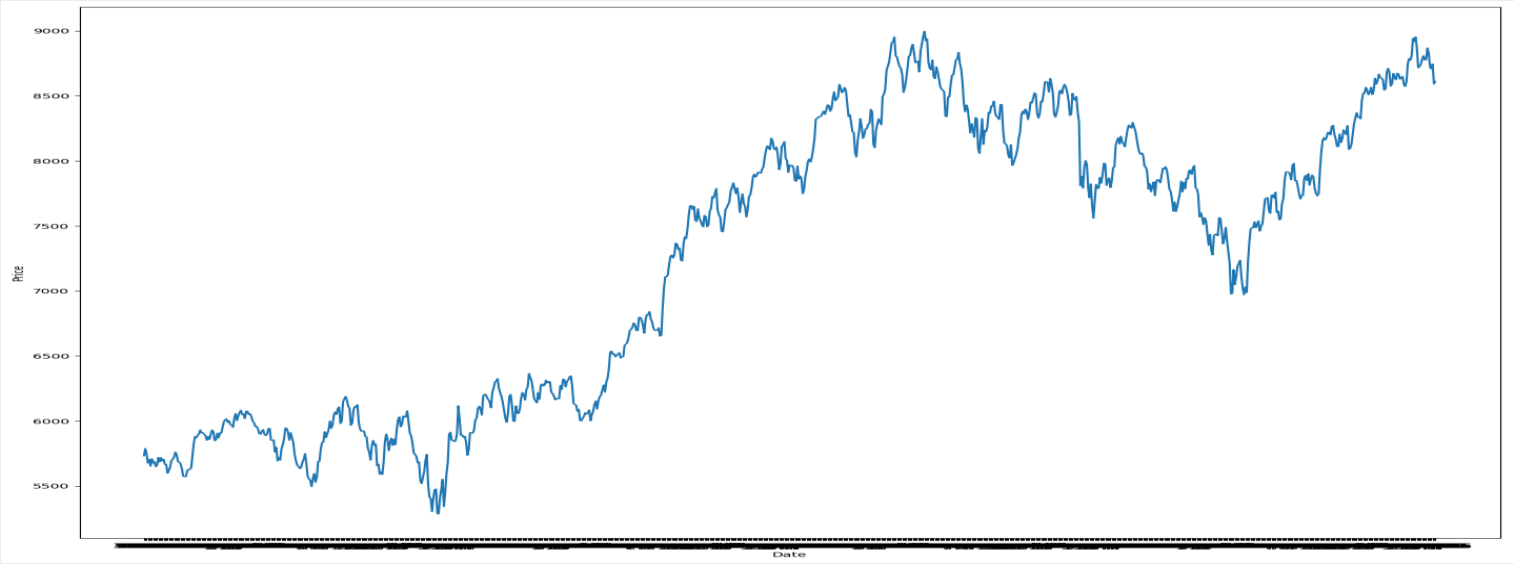
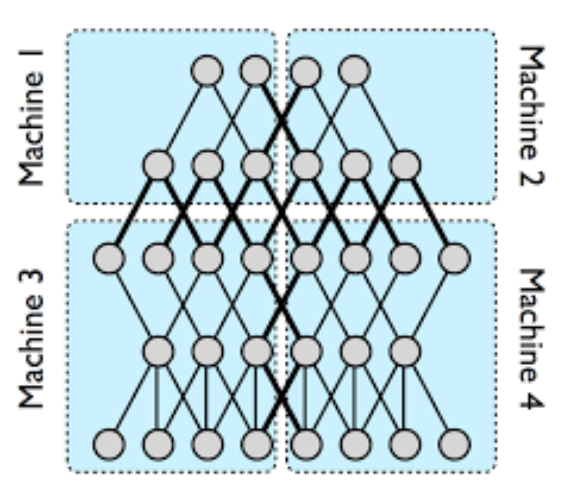


Fig: Historical data of Nifty 50 of past 7 years.

Previously Proposed Methodology

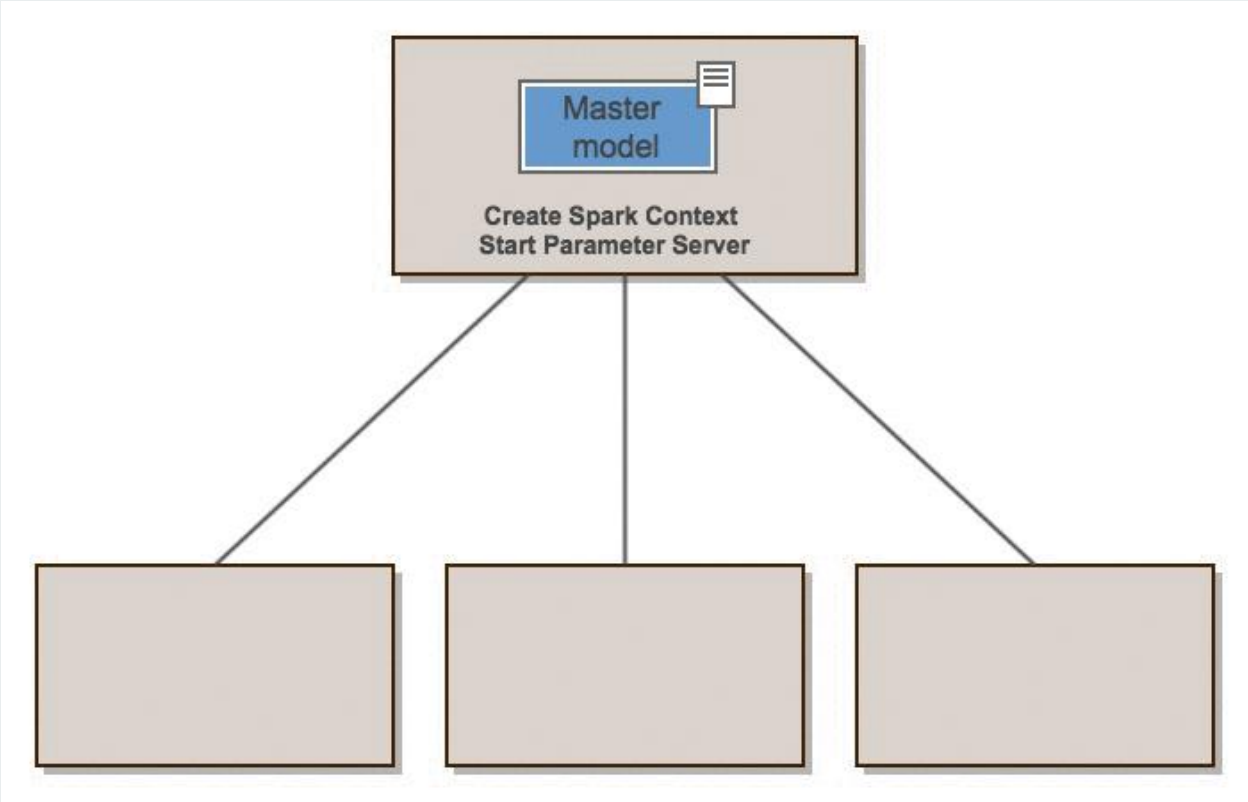
- Downpour SGD:
  - provides asynchronous and distributed implementation of stochastic gradient descent.
  - The training instances are divided across different machines (Data Parallelism).

- Model Parallelism:
  - For large deep learning neural network the model layers itself can distributed across as different machines. This will speed up the training process on CPUs and requirement of GPUs are reduced.



Model Parallelization

Final Methodology



LSTM Training

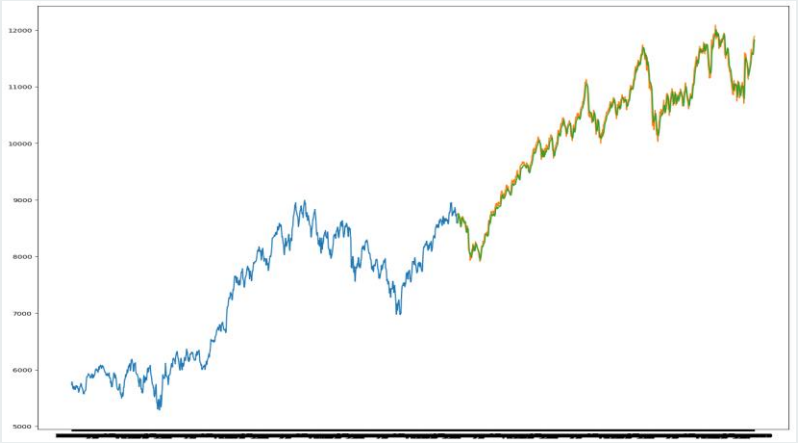
- LSTM stands for Long Short-Term Memory, and it is artificial recurrent neural network (RNN) architecture.
- The difference between feed-forward neural networks and LSTM is that LSTM has a feedback loop to previous layers which helps in prediction using history data and present data.
- A common LSTM cell has an input gate, an output gate and a forget gate. So the cell remembers the information for some time interval and the gates controls the inflow and outflow of the information.
- **LSTMs are used to make temporal inferences on historical data.**
- This Network can be updated on daily basis for new data an it is called Rolling Window.
- This LSTM Model will be distributed as explained in Methodology above.

Parameters To be used for training

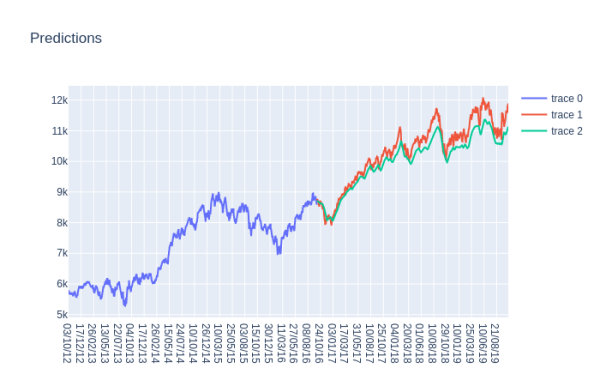
- Daily closing price of stock.
- High/Low prices of stock.
- High/Open/Low prices of stock.
- High/Open/Close/Low prices of stock.

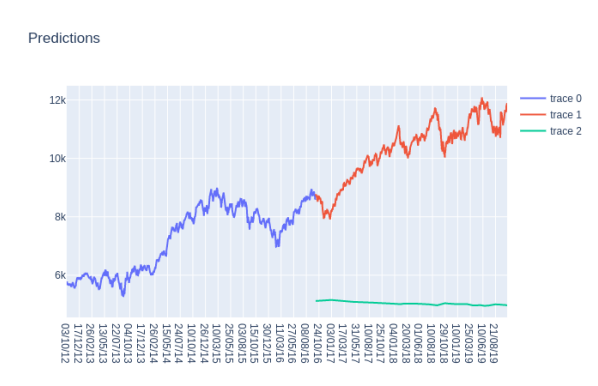
Inference Training

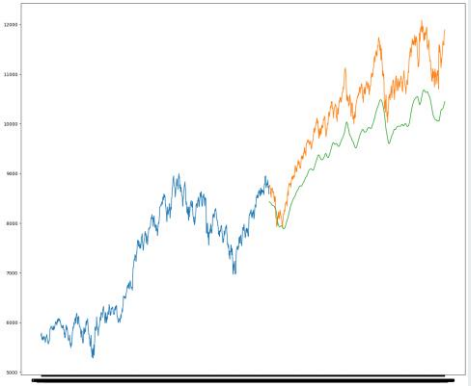
The Figure shown to the right is LSTM training data in blue validation data in yellow and prediction on validation data in green. For 500 epochs of training it and approximately 1700 data points it took 500 seconds to train and validate data. So the goal of this project is to Predict accurately and efficiently using distributed training. So in future large models can be built using the same technique and help in decreasing training time.



Inference Training







Challenges:

- Main Challenge here is the distribution of model. There are some frameworks in spark for distribution of model. Two of them are BigDL and Elephas.
- Another challenge is how the data can be handled asynchronously. This can be handled using Downpour SGD.
- Identifying the right parameters for the prediction of stock prices.
- Initially I was working on Windows 10 for creating the model but found it difficult to combine the discrete components such as LSTM, pyspark and jupyter notebook. So I switched to Linux platform and till now installed apache spark, hadoop, jupyter notebook, pyspark and elephas.

References:

1. Dean, Jeffrey, et al. "Large scale distributed deep networks." *Advances in neural information processing systems*. 2012.
2. Shrivastava, Disha, Santanu Chaudhury, and Dr Jayadeva. "A data and model-parallel, distributed and scalable framework for training of deep networks in apache spark." *arXiv preprint arXiv:1708.05840* (2017).
3. https://software.intel.com/en-us/articles/bigdl-distributed-deep-learning-on-apache-spark
4. J. Dean, G.S. Corrado, R. Monga, K. Chen, M. Devin, QV. Le, MZ. Mao, M'A. Ranzato, A. Senior, P. Tucker, K. Yang, and AY. Ng. Large Scale Distributed Deep Networks.
5. https://github.com/cerndb/dist-keras?source=post_page-----6d397c16abd----------
6. https://github.com/maxpumperla/elephas?source=post_page-----6d397c16abd----------

Code Snippets for Spark and Keras Integration

Create local pyspark context

```
from pyspark import SparkContext, SparkConf
conf = SparkConf().setAppName('Elephas_App').setMaster('local[8]')
sc = SparkContext(conf=conf)
```

Define keras Model

```
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.optimizers import SGD
model = Sequential()
model.add(Dense(128, input_dim=784))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(10))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer=SGD())
```

Create RDD from numpy arrays

```
from elephas.utils.rdd_utils import to_simple_rdd
rdd = to_simple_rdd(sc, x_train, y_train)
from elephas.spark_model import SparkModel
spark_model = SparkModel(model, frequency='epoch', mode='asynchronous')
spark_model.fit(rdd, epochs=20, batch_size=32, verbose=0, validation_split=0.1)
```

Run Script using:

```
$ spark-submit --driver-memory 1G ./your_script.py
```