Stock Prediction using Bidirectional LSTM

by Siba

Task: The goal of this assessment is to apply machine learning for stock market prediction.

Methodology: The dataset is intially loaded using the given functions. Data Preprocessing is done in 3 stages. **Minmax Scaling:** To normalize the data to be in region of 0 and 1, MinMaxScalar has been used. **Exponential Moving Average:** To smooth the data exponential moving average is used . This helps to get rid of the inherent raggedness of the data in stock prices and produce a smoother curve. **Reshaping:** The LSTM network expects the input data to be provided with a specific array structure in the form of: [samples, time steps, features]. The data set is reshaped to add an additional dimension using numpy.reshape(). **Train-Test Split:** To reduce Overfitting train-test split has been introduced and shuffling is avoided.

Model : Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence. This can provide additional context to the network and result in faster and even fuller learning on the problem. So it was intuitive to use it. This is then followed by a Dense layer with softmax activation.

Model Summary:

```
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_2 (Bidirection (None, 1, 20)             4640
_________________________________________________________________
dense_2 (Dense)              (None, 1, 3)              63
```

Results & Interpretation: There a 4 different merge modes that can be used to combine the outcomes of the Bidirectional LSTM layers.They are concatenation (default), multiplication, average, and sum. Comparing the behavior of different merge modes on test data, sum mode outperformed every other mode and yielded an accuracy of 0.745.

Boosting Accuracy: Instead of phrasing the past observations as separate input features, we can use them as time steps of the one input feature, which is indeed a more accurate framing of the problem. Stacked LSTM results in more complex architecture yielding better results.

Shuffle: Assuming that data are taken from sequences , this creates a situation where the exact same value is in both training/testing sets, but in slightly different positions. This causes the classifier to overfit to the testing data, because the two very similar sequences exist in both testing and training. So shuffling may be avoided. But generally its a good practice to shuffle data.

Details to Convey: I used a simple Bidirectional LSTM architecture to depict that classification can be done. Complex architectures can be used to further improve the accuracy. Since I dont own a gpu I had reduced the look back to 8 to speed up training. Regularization was not focussed. So method like batch normalization and dropouts would be absent. With improved hardware they can also be implemented. Since negative values could not be encoded the label -1 was changed to 2.

Libraries Used: Numpy, Matplotlib, Pandas, Sklearn, Keras.