CS898 Deep Learning and Application

Deep Learning for Time Series Analysis

Bo Wang

Scientific Computation Lab

Department of Computer Science
University of Waterloo

Outline

1. Background Knowledge
2. RNN and LSTM
3. Time Series Analysis
4. Future Works

Part I

Background

Time Series Forecasting

- Time series tracks the movement of the chosen data points
  - A sequence of numerical data points in successive order
  - Such as a S&P 500 index value, over a specified period (1994-2007) with data points recorded at regular intervals (daily, weekly,…)
- Uses historical values and associated patterns to predict future activity
  - Include trend analysis, cyclical fluctuation, and issues of seasonality
  - As with all forecasting methods, success is not guaranteed!
  - Collaborate with fundamental analysis to make trading decision.

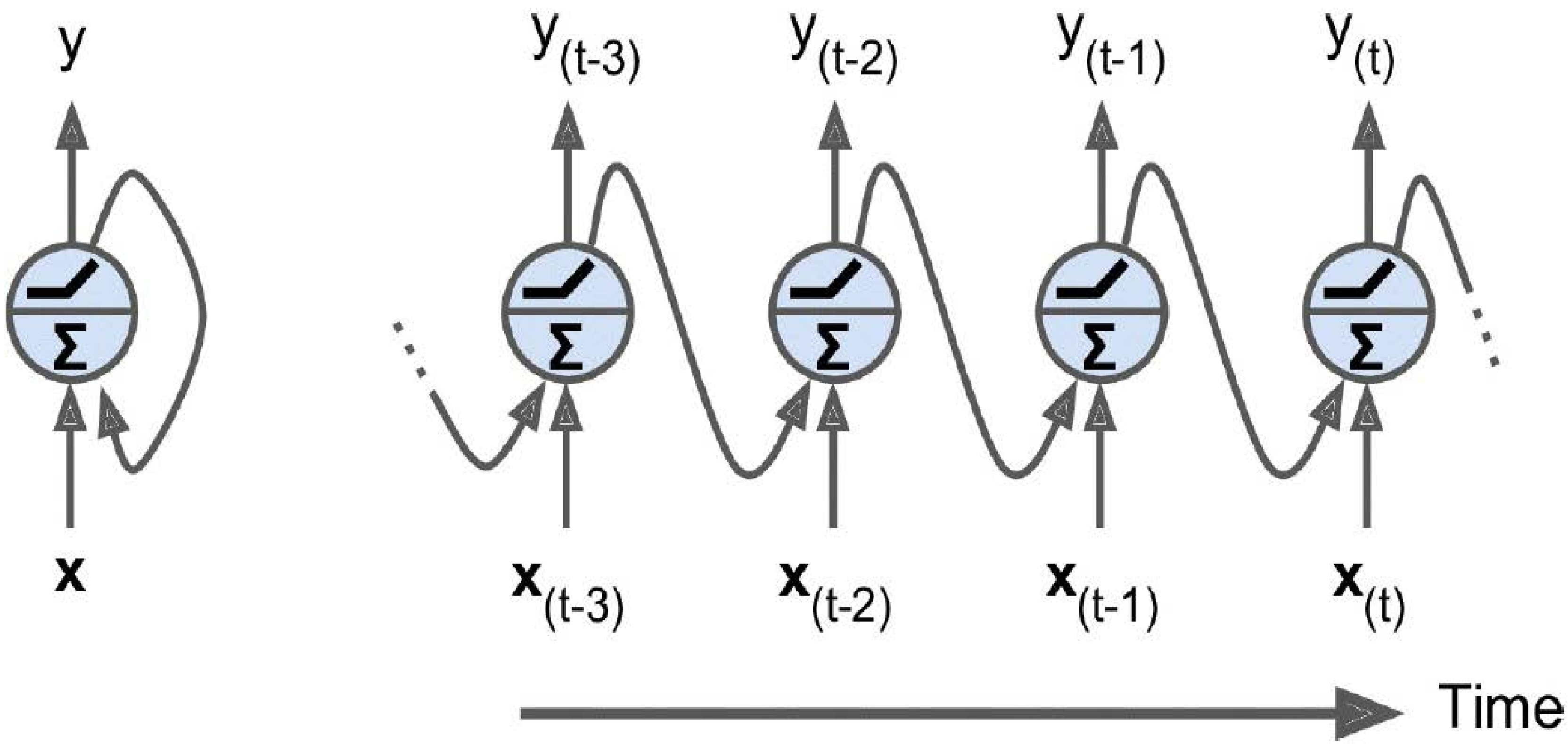
S&P 500 Index

- Standard & Poor's 500 is an American stock market index based on the market capitalizations of 500 large companies, seen as a leading indicator of U.S. equities and a reflection of performance of large-cap sector of the market.
- Analysts and economists at Standard & Poor's select 500 stocks by considering factors as market size, liquidity and industry grouping.
- It uses a market cap methodology, giving a higher weighting to larger companies.
- Products for replicating S&P 500: S&P 500 index funds, S&P 500 ETFs

Part II

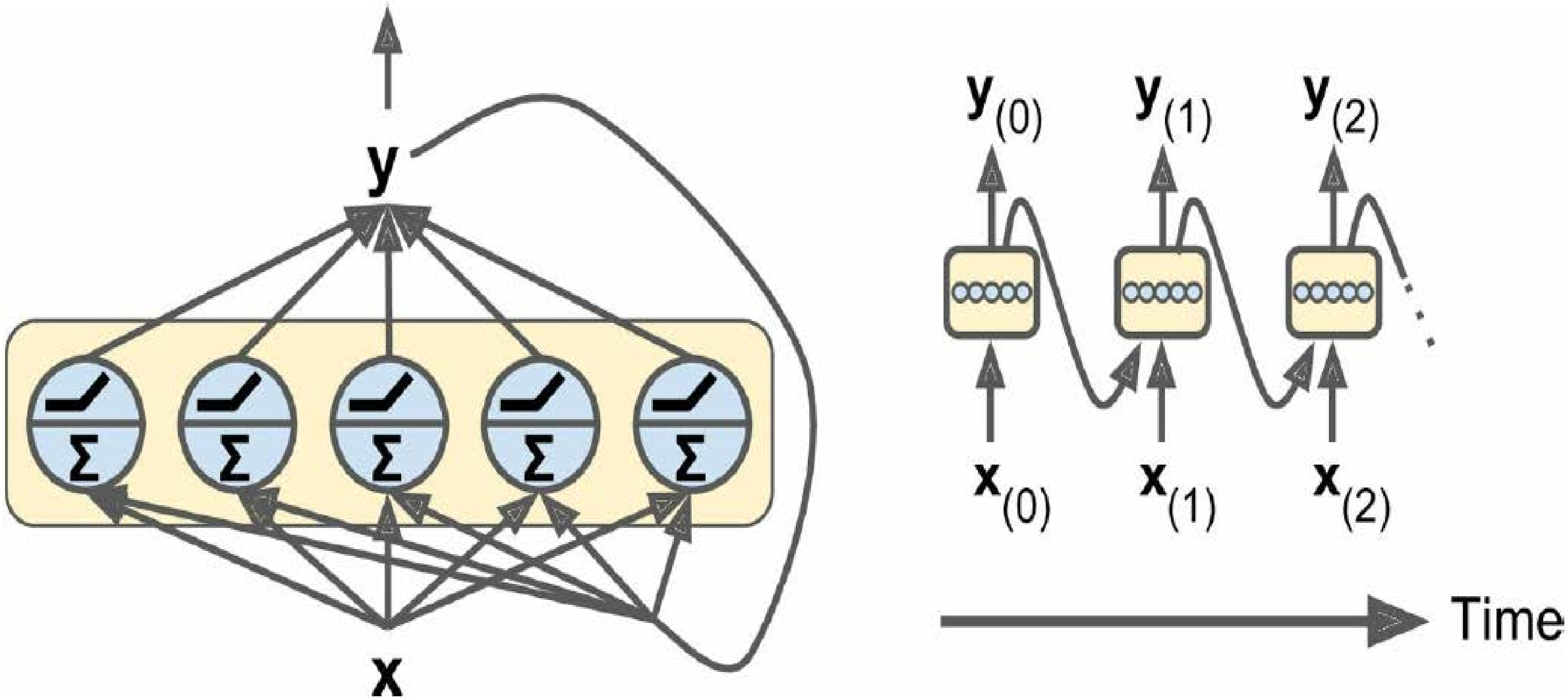
RNN and LSTM

Recurrent Neurons



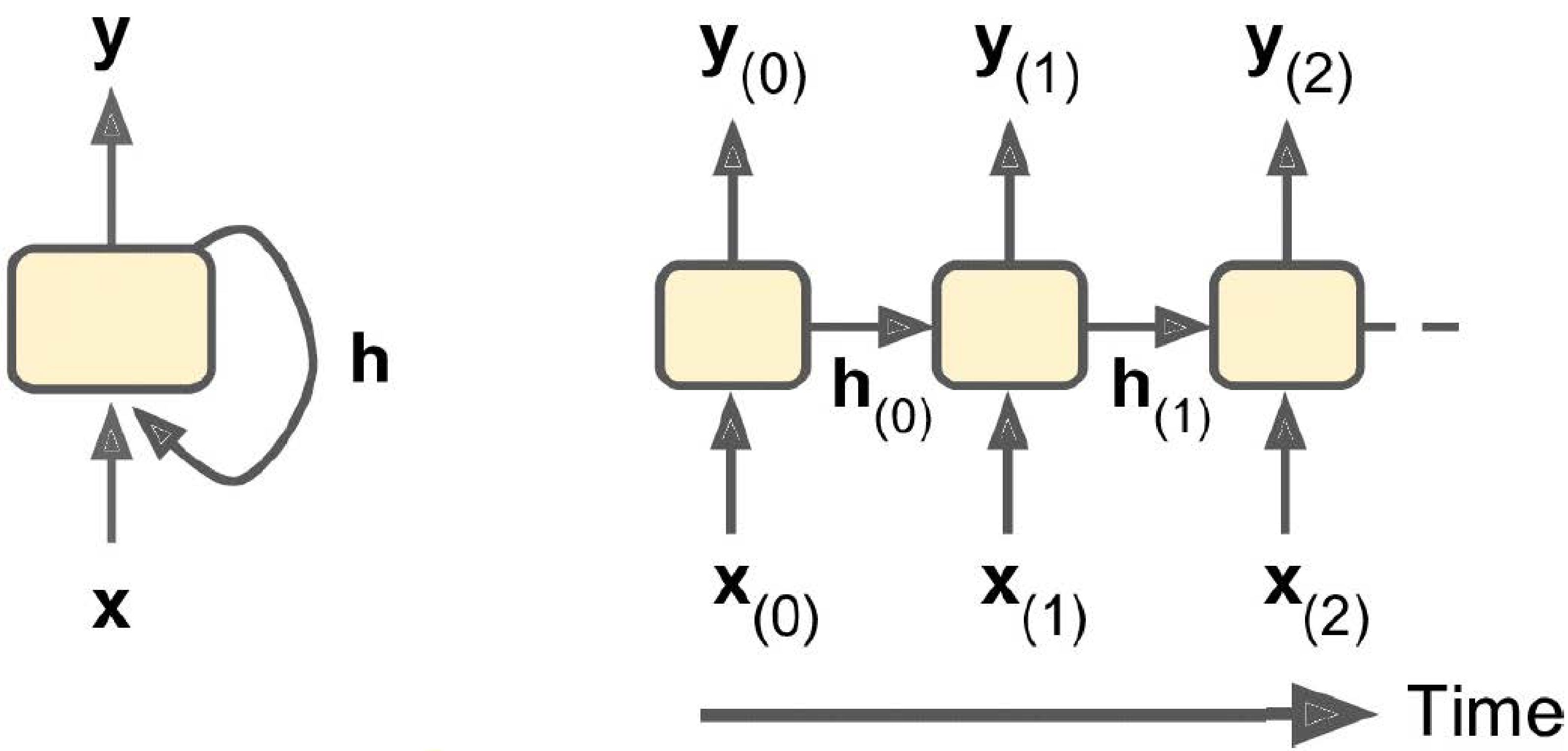
- A recurrent neuron (RN, the simplest possible RNN) on the left is unrolled through time on the right.
- RN looks like feedforward neuron, except it has connections pointing backward.
- At each time step **t**, RN just has one neuron receiving inputs, producing an output, and sending that output back to itself.

A Layer of Recurrent Neurons



- A layer of 5 RNs as a cell, both the inputs and outputs are vectors
- Each RN has 2 sets of weights: 1 for the inputs x(t) and the other for the outputs of the previous time step y(t–1)
- y(t) is a function of x(t) and y(t–1), which is a function of x(t–1) and y(t–2), and so on. So, y(t) a function of all the inputs since t = 0 (x(0),x(1), …, x(t)), and y(-1) is assumed to be all 0.

Memory Cell

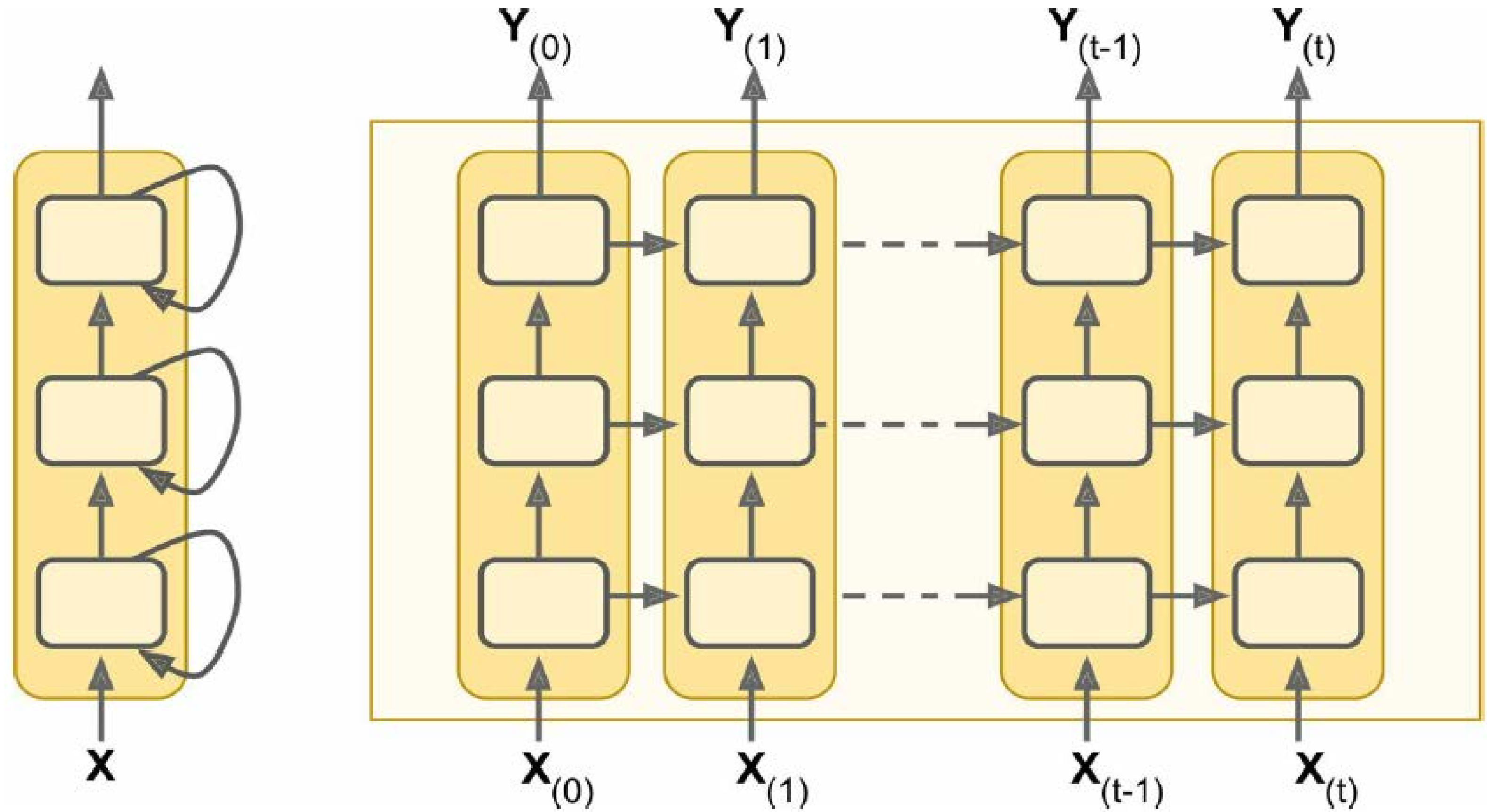


- Since y(t) a function of all the inputs since t = 0, the part of RN preserving states across time axis is a "Memory Cell" ("Cell")
- A cell's sate at t is h(t)=f(h(t-1), x(t)), where x(t) is current inputs, h(t-1) is cell's state at t-1.
- For single RN or a layer of RNs, y(t)=h(t). But for complex cells, y(t)!=h(t), i.e. the LSTM cell

Types of RNN

- Sequence (input) to Sequence (output)
  - Simultaneously take a Seq. of inputs and produce a Seq. of outputs
  - Predicting time series: feed RNN the prices over the last N days, and it output the prices shifted by 1 day into the future (i.e., from N – 1 days ago to tomorrow)
- Sequence (input) to Vector (output)
  - Feed the RNN a Seq. of inputs, and ignore all outputs except for the last one (only output last one)
- Vector (input) to Sequence (output)
  - Feed the RNN a single input at t=0 and zeros for all other time steps, and it output a sequence
- Seq-to-Vec (Encoder), Vec-to-Seq (Decoder)
  - Language Translating: feed the Encoder of RNN a sentence of one language, Decoder of RNN outputs a sentence in another language.

Deep RNN

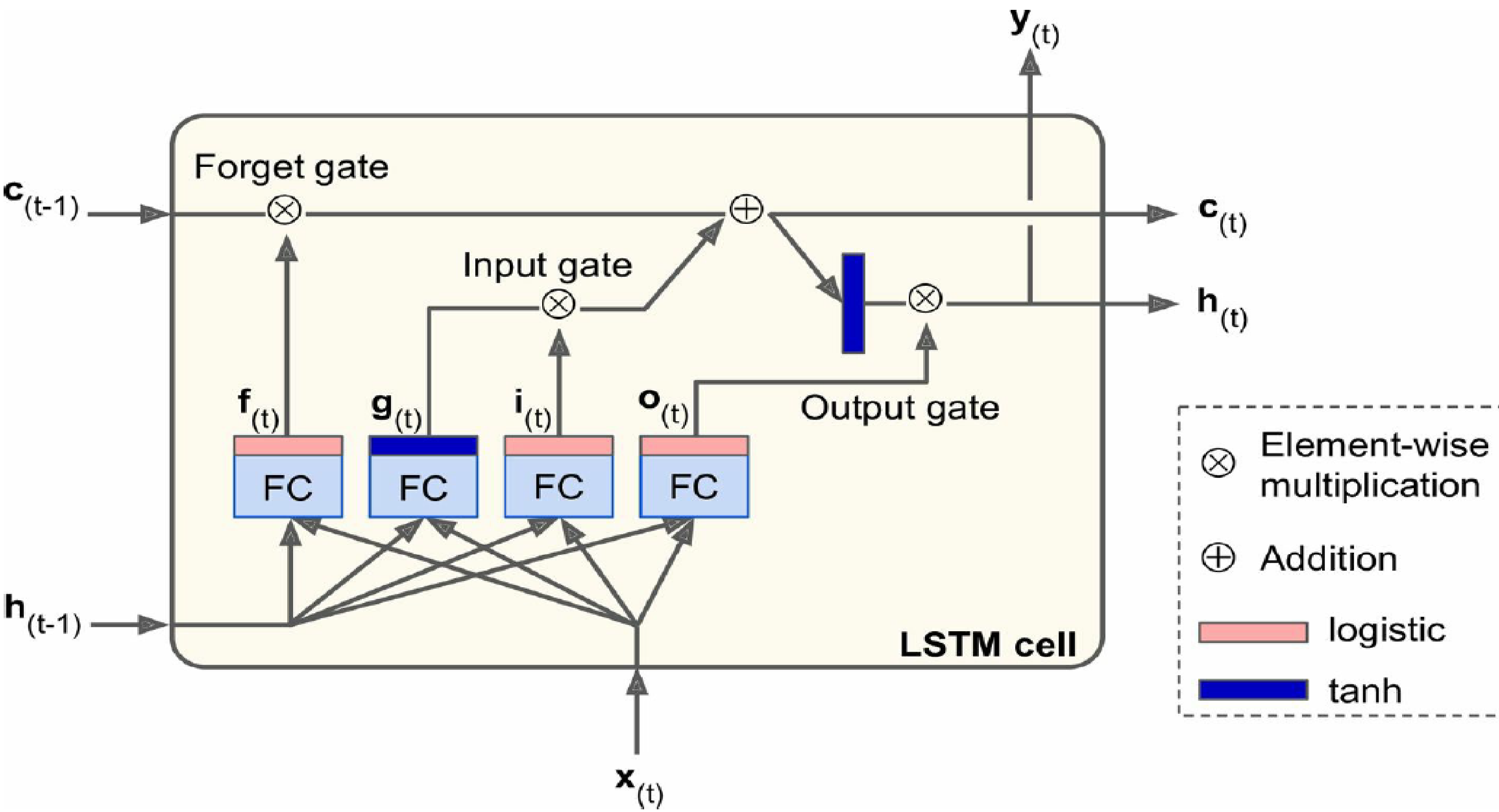


- Stack multiple layers of cells to create a deep RNN
- Stack identical cells into a deep RNN
- Use various kinds of cells (BasicRNNCell, BasicLSTMCell, …) with different number of neurons

Training over Many Time Steps

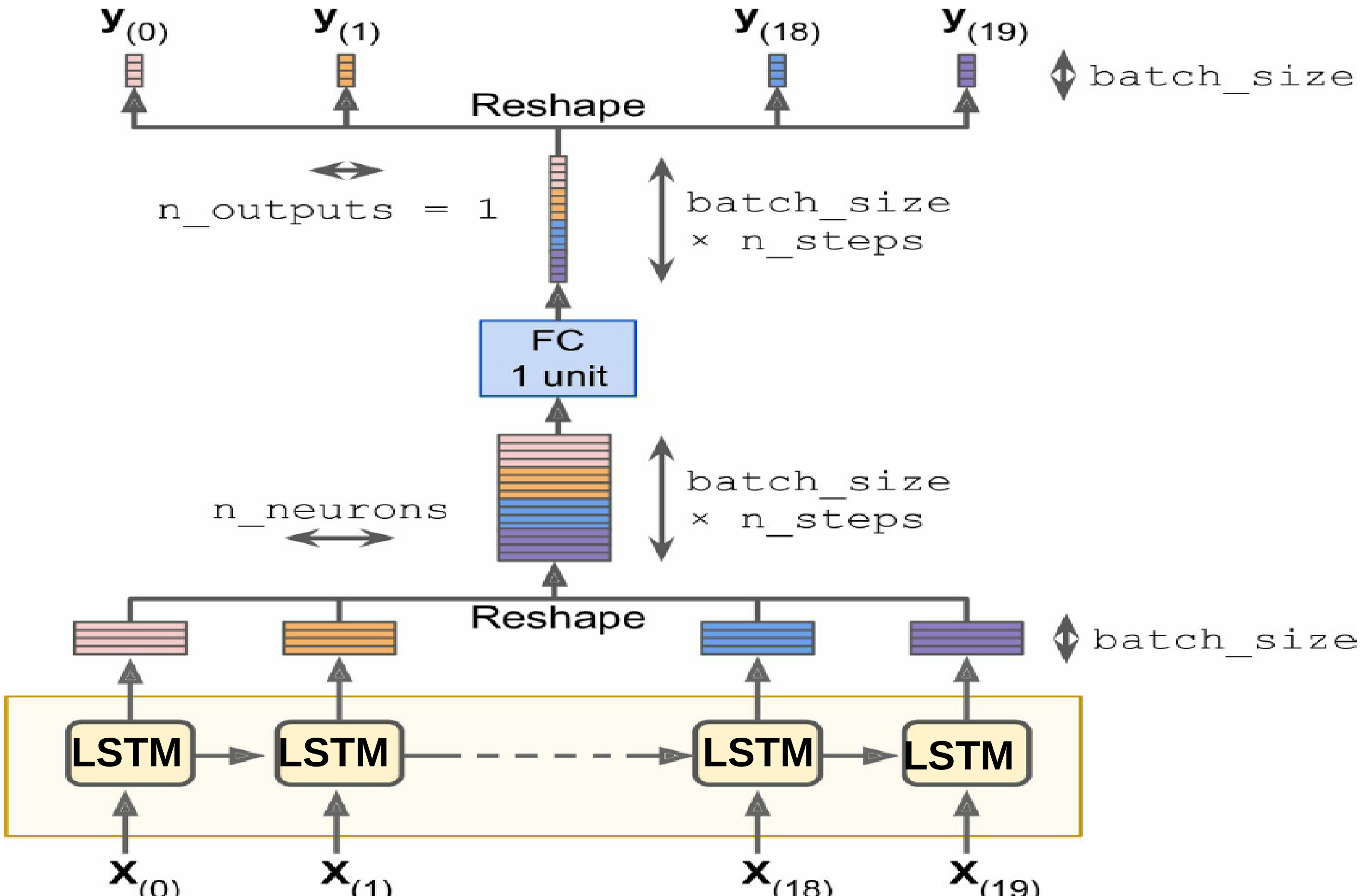
- Suffer from the vanishing/exploding gradients (train forever)
  - Tricks: parameter initialization, ReLU activation function, batch normalization, gradient clipping, faster optimizer
  - Training still be very slow for a moderate long sequence (i.e. 50 steps)
  - Truncated backpropagation through time: unroll the RNN only over a limited number of time steps during training by simply truncating the input sequences
  - Model will not be able to learn long-term patterns.
- Memory of those first inputs gradually fades away
  - Some information is lost after each time step, due to the transformation of data when it traverse an RNN.
  - After a few time steps, RNN's state hardly contains the information from those first inputs.
- A popular solution: Long Short-Term Memory (LSTM)

Basic LSTM Cell



- Its training will converge faster and detect long-term dependencies in the data
- Its state is split in two vectors: h(t) as short-term state, c(t) as long-term state.
- c(t-1) first drop some previous memories, then add some new current memory
- After addition, c(t-1) is copied and passed through "tanh" and "Output gate" o(t)

LSTM Cell for Time Series

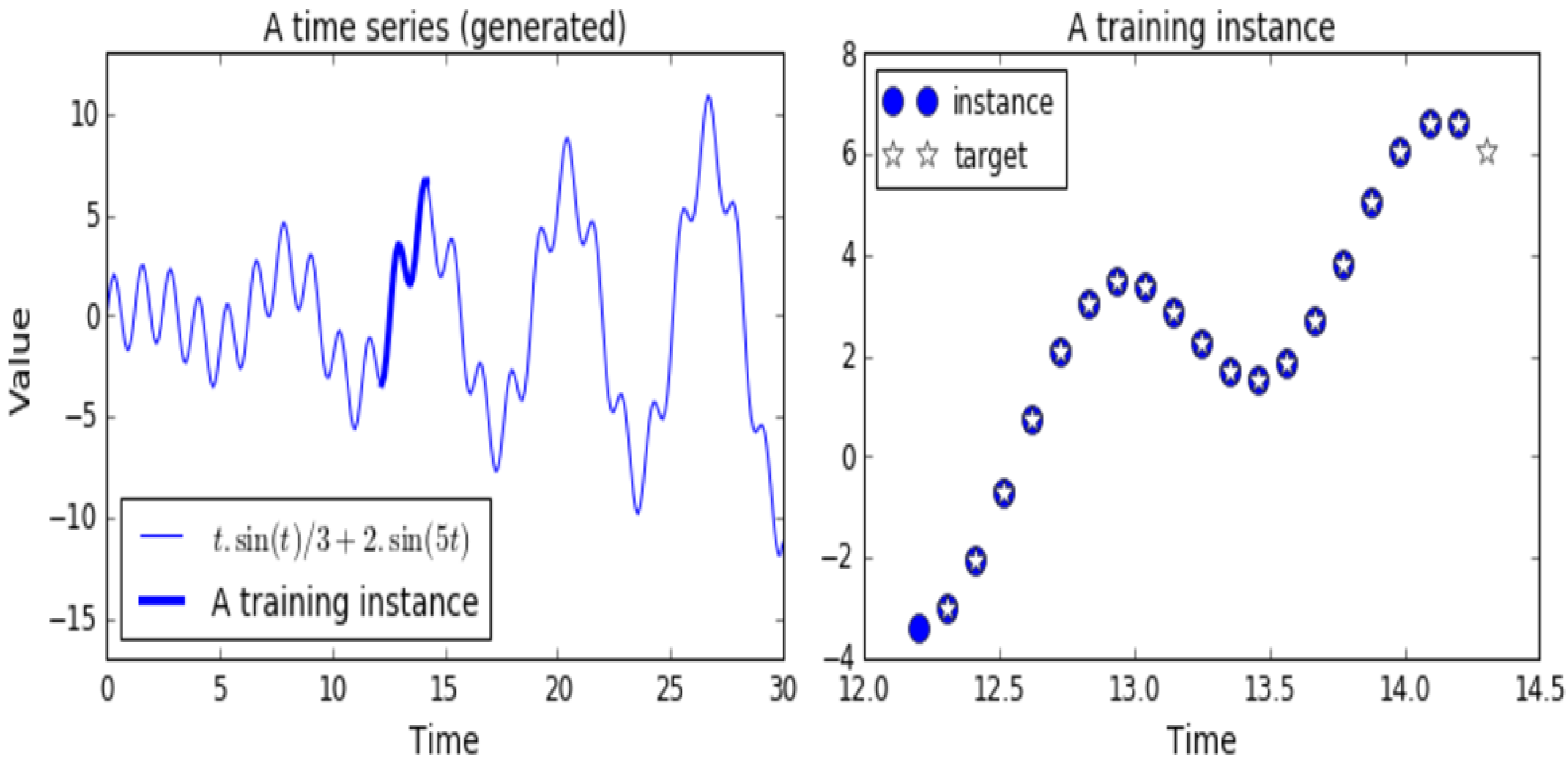


- LSTM has n_neurons neutrons and unroll over 20 steps, 4 instances per batch
- Stack all the outputs of 20 time steps, projecting output vector of size n_neurons into a single output value at each time step by a Fully Connected layer (FC)
- FC only consists linear neurons, without any activation function.

Part III

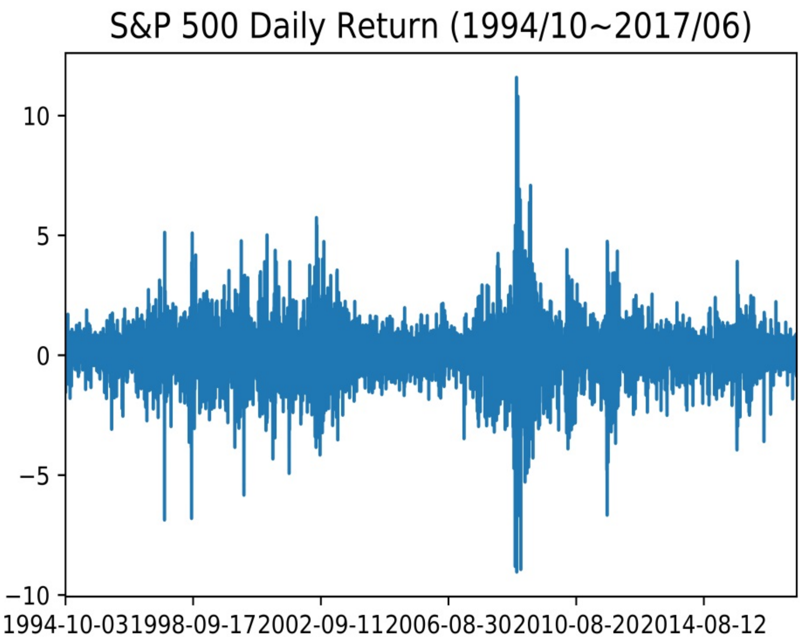
Time Series Analysis

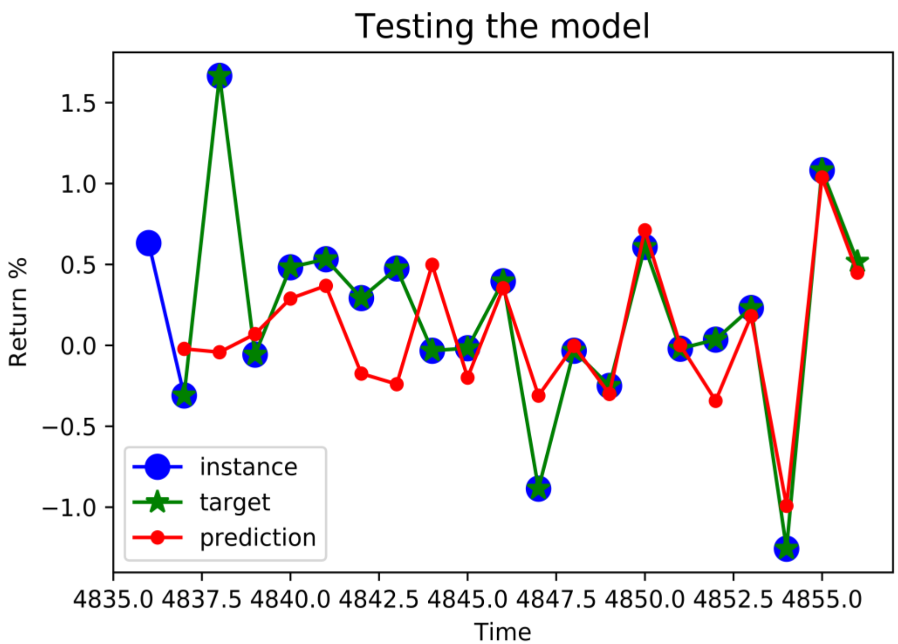
Data of Time Series

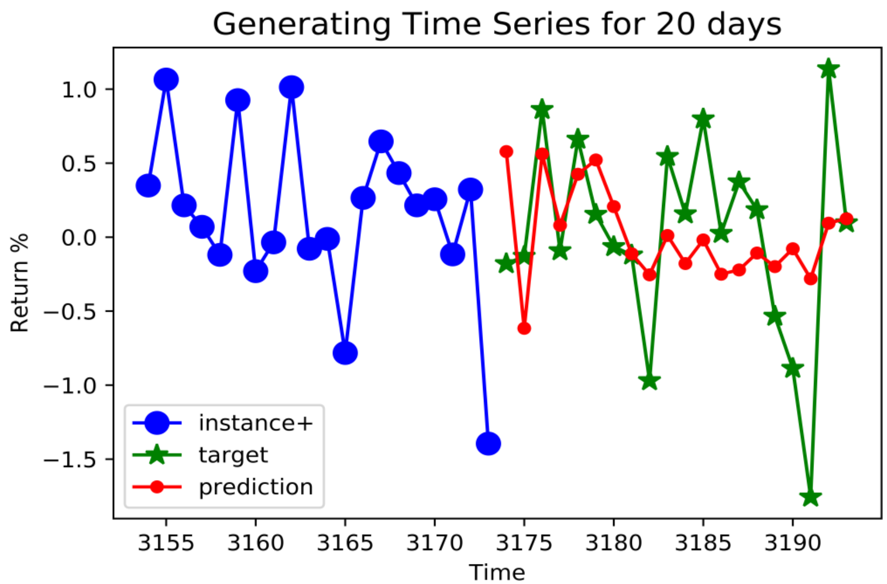


- It's about predicting the next value in a generated time series.
- A training instance is a randomly selected sequence of 20 consecutive values from the time series
- A target sequence is shifted of input instance by one time step into the future.
- Further, append the predicted value to the sequence, feed the last 20 values to the model to predict the next value, and so on, generating a creative sequence.

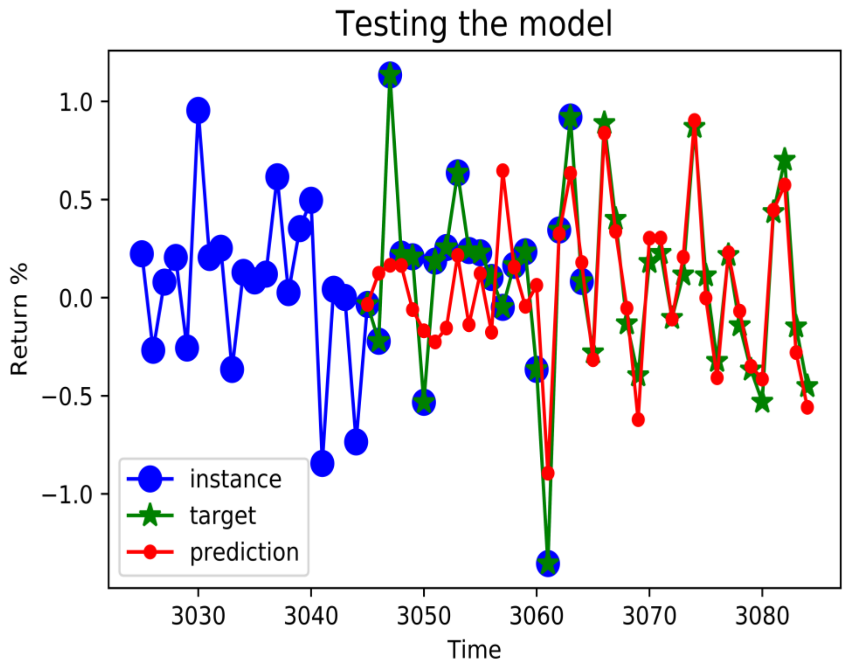
Forecast Monthly Return by Daily Data

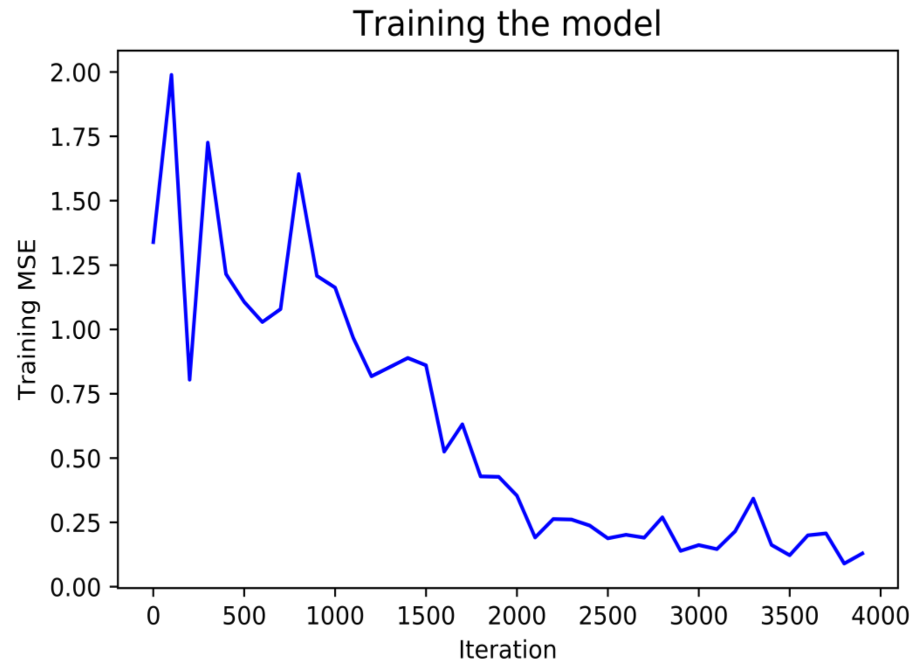






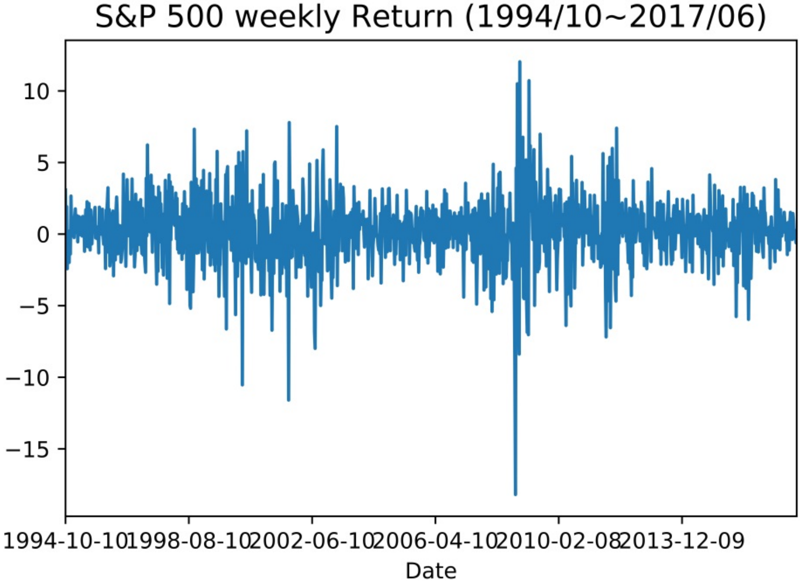
Forecast Monthly Return by Daily Data

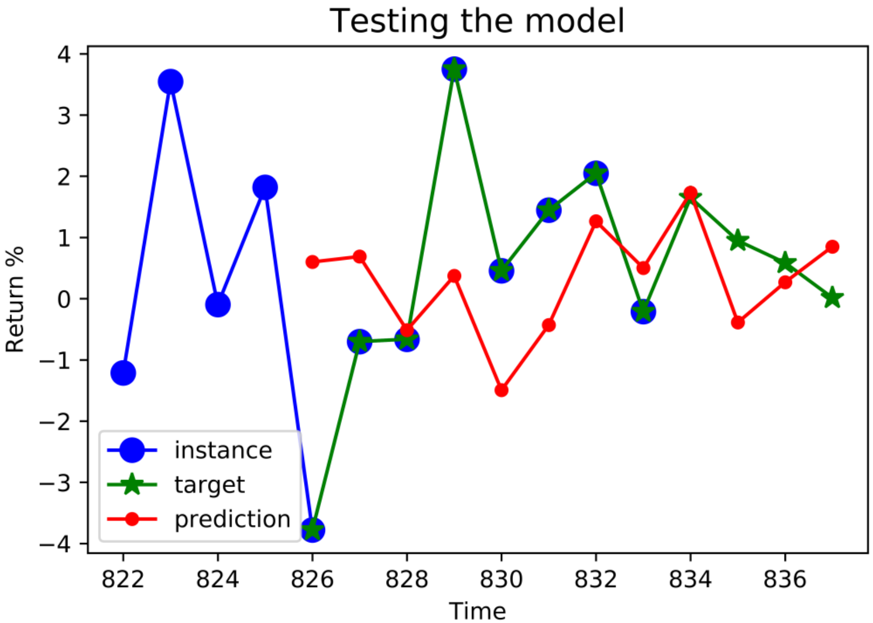




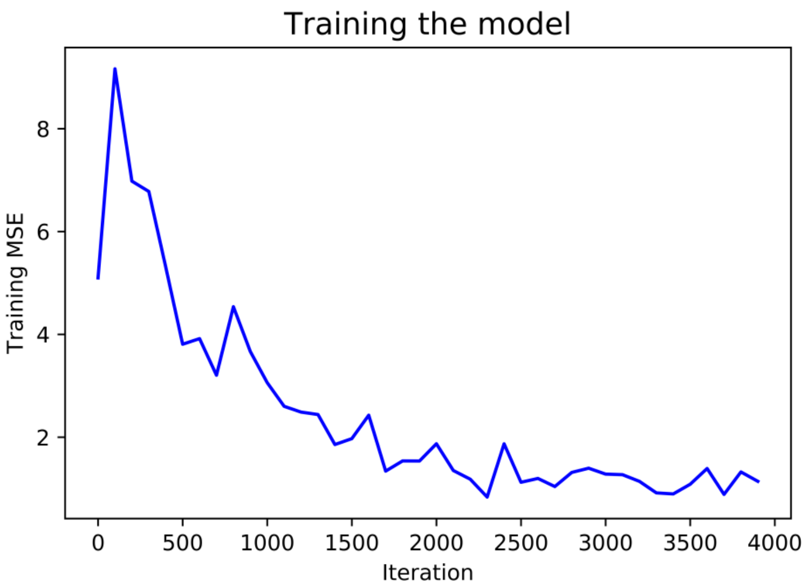
- 40 time steps to predict next 20 days return, sum to be a monthly return
- LSTM, 3 layers, "tanh" function, 1 input, 1 output, 300 neutrons
- AdamOptimizer, learning rate 0.001, batch size 20, Train/Test split ratio 0.2
- Tried but not applied: more layers, more neutrons, different batch size, dropout, ReLU, different time steps
- Performance: Testing MSE is 0.293, Predicting (20 days only) MSE is 0.012,
  Predicting Monthly Return MSE is 0.342

Forecast Monthly Return by Weekly Data

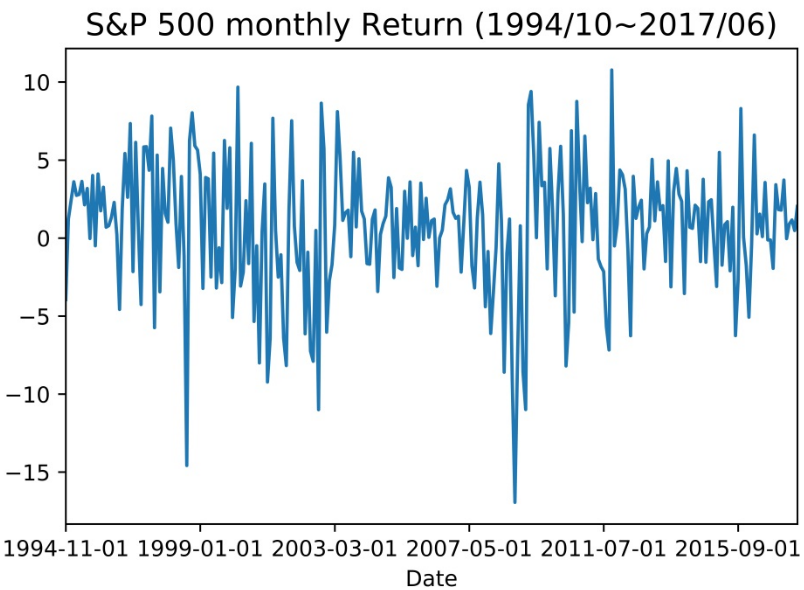




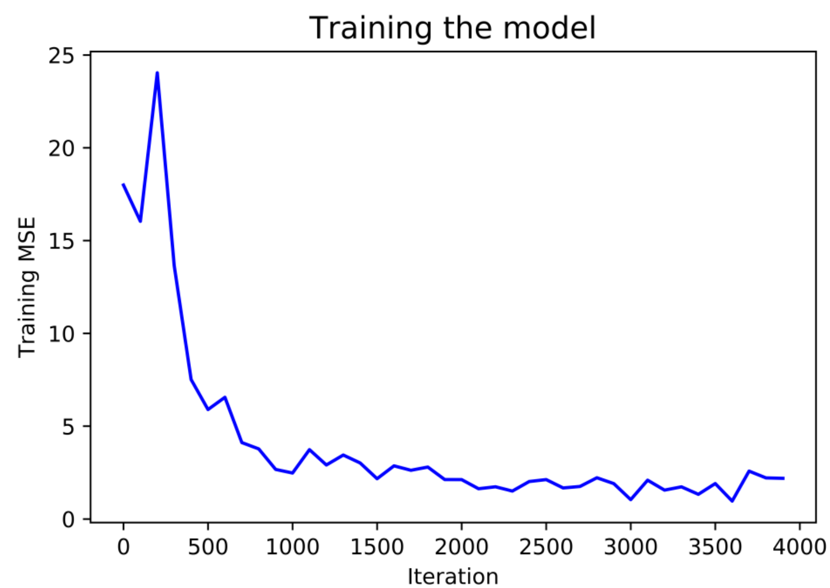
- 12 time steps to predict next 4 weeks return, sum to be a monthly return
- Testing MSE is 2.561
- Predicting (4 weeks) MSE is 0.313
- Predicting Monthly Return MSE is 1.175



Forecast Monthly Return by Monthly Data



- 12 time steps to predict next 1 month return, as a monthly return
- Testing MSE is 7.10
- Predicting (1 month) MSE is 0.121
- Predicting Monthly Return MSE is 0.121



Compare and Ensemble 3 LSTM Models

- The above 3 LSTM models were trained and tested by different data set. To have a fair comparison, we use the same data (daily data, calculate corresponding weekly/monthly data) for training and testing.
  - Daily-LSTM: Training MSE is 0.147, Testing MSE is 0.178, Predicting (20 days only) MSE is 0.011, Predicting Monthly Return MSE is 0.324
  - Weekly-LSTM: Training MSE is 1.493, Testing MSE is 2.440, Predicting (4 weeks only) MSE is 0.343, Predicting Monthly Return MSE is 1.526
  - Monthly-LSTM: Training MSE is 6.323, Testing MSE is 6.588, Predicting (1 month only) MSE is 0.453, Predicting Monthly Return MSE is 0.453
- Ensemble 3 LSTM models to be a Voting Predictor
  - Output the mean of 3 LSTMs' monthly return values
  - Predicting Monthly Return MSE is 0.295
  - 0.543% error for predicting monthly return in range -10%~10%

Part IV

Future Works

Future Works

- Improving prediction performance by using bagging/boosting to ensemble Daily-LSTM, Weekly-LSTM, and Monthly-LSTM.
- Use more features and datasets to improve prediction performance.

Thank you for your patience.

Q/A