Senior Project I
XiaoEx - The Exchange Expert

By
Kasperi Reinikainen 5818014
Hein Htet Naing 5818035
Asnai Narang 5815228
Content

1. Introduction
2. Motivation and background
3. In brief: Forex & Neural Networks
4. Reference study and Our initial approach
5. Development stages
6. Evaluation and assessment
7. Architecture of application use-case
Introduction

- Given problem: Predict Forex market movements using DNN’s
- Case-study to follow in our Initial approach
- Improve results from case-study or draw relevant conclusions of it
- Apply findings in tangible use-case
Introduction: goals

CHECKLIST:

☐ Comprehend and apply the Case-study’s approach in the Initial design

☐ Branch-off and find better models using our own techniques

☐ Apply the model in a tangible application use-case
Motivation and background

Main motivation
⇒ Learn and apply modern ML-techniques in a challenging use-case
⇒ Find applicability for the results

Members:
Asnai Narang, 3rd year CS major
Hein Htet Naing (Hector), 3rd year IT major
Kasperi Reinikainen, 3rd year CS major
In brief: Forex markets

- Foreign Exchange: Currency markets for trading foreign currencies in pairs
- Target users: Commercial and central banks, Investment and other large companies, Governments
- Forex trading: buy currency that expect to raise value, sell currency that is expected to lose value
In brief: Artificial Neural Networks

- Original development inspired by Brain
- Can potentially approximate problems with any level of complexity
- ‘Learns’ by adjusting weights between different layers of neurons
- 3 main components (not incl. loss-func.):
  1. Weight calc. (integration function)
  2. Activation function (scales the output)
  3. Optimization function (param. update)
Single neuron computational graph
Content

1. Introduction
2. Motivation and background
3. In brief: Forex & Neural Networks
4. Reference study and Our initial approach
   1. Reference study
   2. Our first model (initial approach)
5. Development stages
6. Evaluation and applicability
7. Architecture of application use-case
Reference studies

Studies regarding Forex Prediction using ML - techniques are not hard to find.

Similarities between all studies (incl. Case study):

- They all (except one using SVM) use some form of Artificial Neural Networks

- Features are pre-defined and selected mostly intuitively based on various statistical formulations of ‘raw’ OHLC - currency data

- Prediction accuracy is relatively low (ranging mostly between 40-60 % for classification problems)
Case study

Prediction of Exchange Rate Using Deep Neural Networks, presentation by University of Nagoya

Training conditions for case study:

- **Assumptions:**
  1. Future trend consists of past information.

- **Prediction types:**
  1. Classification: { Up, Down }

- **ANN Type:**
  1. Deep neural network

- **Features:**
  1. 10-features:
     { open, close, high, low, datetime, volume, RSI, stochastic RSI, Moving avg, %R }
  2. Concatenated (method unknown) to become 100 features

- **Dataset:**
  → USD/JPY 01/01/1991 - 31/12/2014
  Total of 97,362 instances
# Case study training settings

<table>
<thead>
<tr>
<th></th>
<th>Instances in dataset</th>
<th>Train / % train</th>
<th>Total Features</th>
<th>Layers</th>
<th>Neurons (total)</th>
<th>Activation</th>
<th>Optimisation</th>
<th>Learning-rate</th>
<th>Batch_size</th>
<th>No-epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nagoya University</td>
<td>96,366</td>
<td>46,451 / 48%</td>
<td>10 (concat to 100)</td>
<td>5</td>
<td>256</td>
<td>Sigmoid</td>
<td>Gradient Descent</td>
<td>0.00006</td>
<td>128</td>
<td>50</td>
</tr>
</tbody>
</table>
Case study: test settings and results

- Number of tests: $51,516$
- Total accuracy range for tests: $50.40\% - 53.46\%$
Our initial approach (first model)

- **Assumptions:**
  1. Future trend consists of past information.
  2. We expect that case-study followed common naming when talking about layers. 
     \[4 + 1 = 5\] layer setting expected
  3. We assume (based on the presentation) they used 48% of data for training in initial case
  4. There is no ‘stall’ when price doesn’t move. We label it as Down.

- **Prediction types:** Classification: \{ Up, Down \}

- **Dataset:**
  - USD/THB → 13/2/2017 ~ 13/10/2017 by Dukascopy online
  - At first 5833 instances, after removing 0-volume (noises) days: 3785 instances
First model: Data preprocessing

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.02.2017 13:00.0C</td>
<td>35.0803</td>
<td>35.0825</td>
<td>35.0397</td>
<td>35.0439</td>
<td>121259961</td>
</tr>
<tr>
<td>13.02.2017 14:00.0C</td>
<td>35.0499</td>
<td>35.0917</td>
<td>35.0512</td>
<td>2651639893</td>
<td></td>
</tr>
<tr>
<td>13.02.2017 15:00.0C</td>
<td>35.0498</td>
<td>35.0795</td>
<td>35.0289</td>
<td>2515360107</td>
<td></td>
</tr>
<tr>
<td>13.02.2017 16:00.0C</td>
<td>35.0738</td>
<td>35.0831</td>
<td>35.0267</td>
<td>3030510010</td>
<td></td>
</tr>
<tr>
<td>13.02.2017 17:00.0C</td>
<td>35.0589</td>
<td>35.0822</td>
<td>35.0131</td>
<td>2871660107</td>
<td></td>
</tr>
<tr>
<td>13.02.2017 18:00.0C</td>
<td>35.0486</td>
<td>35.0525</td>
<td>35.0162</td>
<td>1476300049</td>
<td></td>
</tr>
<tr>
<td>13.02.2017 19:00.0C</td>
<td>35.0479</td>
<td>35.0722</td>
<td>35.0171</td>
<td>307040039</td>
<td></td>
</tr>
</tbody>
</table>

**Raw data**

**Processed data**
## First model: Training settings

<table>
<thead>
<tr>
<th>T1</th>
<th>Instances in dataset</th>
<th>Train / % train</th>
<th>Features</th>
<th>Layers</th>
<th>Neurons (total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nagoya University</td>
<td>96,366</td>
<td>46,451 / 48%</td>
<td>10 (concat to 100)</td>
<td>5</td>
<td>256</td>
</tr>
<tr>
<td>Assumption University</td>
<td>3785</td>
<td>1821 / 48%</td>
<td>10</td>
<td>5</td>
<td>256 in hidden layers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T2</th>
<th>Activation</th>
<th>Optimization</th>
<th>Learning-rate</th>
<th>Batch_size</th>
<th>No-epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nagoya University</td>
<td>Sigmoid</td>
<td>Gradient Descent</td>
<td>0.00006</td>
<td>128</td>
<td>50</td>
</tr>
<tr>
<td>Assumption University</td>
<td>Sigmoid</td>
<td>Gradient Descent</td>
<td>0.00006</td>
<td>128</td>
<td>50</td>
</tr>
</tbody>
</table>
First model: test settings and outcomes

Test settings:

- 4 tests, each having 400 test instances and testing different parts of the dataset.

<table>
<thead>
<tr>
<th>T3</th>
<th># test instances</th>
<th>% accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nagoya University</td>
<td>744 - 51516</td>
<td>50.40 % - 53.46 %</td>
</tr>
<tr>
<td>Assumption University</td>
<td>400 - 1600</td>
<td>50.50 % - 54.75 %</td>
</tr>
</tbody>
</table>
First model: Conclusion

- Accuracy of our initial model and the case study are almost exactly alike
- Assumptions were not affecting negatively
- The intentional changes did not affect negatively (as expected)
- Even though successfully followed the case study’s results
  → Not really a great level of accuracy
Content

1. Introduction
2. Motivation and background
3. In brief: Forex & Neural Networks
4. Reference study and Our initial approach
5. Development stages
   1. Finding optimal neuron-layer setup
   2. Optimizing training-instance settings
   3. Intuition of the tests
   4. Optimal prediction times
6. Evaluation and assessment
7. Architecture of application use-case
Development stage: neuron-layer setup (setting)

- Permutations (6,4) ⇒ 360 possible rounds
- Dataset ⇒ 3,785 instances
- Training set ⇒ 100 instances
- Num_Test ⇒ 100
- Optimizer ⇒ Gradient Descent
- Activation func. ⇒ ReLU
- Number of epoch ⇒ 50
- Batch size ⇒ 38
- Optimization steps ⇒ (100 / 38 * 50) = 198 steps
Development stage: neuron-layer setup (result)

- Mean: 0.51
- Mode: 0.52
- Range: 0.42 ~ 0.55
- Best Result: \{4, 16, 64, 32\}
Development stage: no. of training-instances (setting)

- Number of rounds ⇒ 17 rounds with each training instance
- Training instances:
  - [30, 60, 90, 120, 150, 180, 250, 300, 400, 500, 750, 1000, 1250, 1500, 2000, 2500, 3000]
- Dataset ⇒ 3,785 instances
- Testing set ⇒ 400 instances of sample size
- Optimizer ⇒ ProximalAdagradOptimizer
- Activation func. ⇒ ReLU
- Learning_rate ⇒ 0.00006
- Number of epoch ⇒ 50
- Batch size ⇒ 128
Development stage: no. of training-instances (setting)

- Mean  ⇒  0.50
- Mode  ⇒  0.49 & 0.52
- Range ⇒  0.46 ~ 0.57
- Best Result  ⇒  3,000 with 57% accuracy
Development stage: intuition from the tests

Focus:

- Adjusting the named parameters don’t improve accuracy much
- Along with adjustment, optimal number of training instances becomes smaller
- Movement of the market affects on overall accuracy

Intuition:

- Num_epoch $\Rightarrow$ 250
- Learning rate $\Rightarrow$ 0.0006
- Batch size $\Rightarrow$ 38
Development stage: optimal prediction times

- Tested train-instance numbers: [30, 60, 90, 120, 150, 180, 250, 300, 400, 500, 750, 1000, 1250]
- Dataset ⇒ 3,785 instances
- Testing set ⇒ 500 tests (for each train-instance test)
- Optimizer ⇒ ProxmialAdagradOptmizer
- Activation func. ⇒ ReLU
- Learning_rate ⇒ 0.0006
- Number of epoch ⇒ 250
- Batch size ⇒ 38
optimal prediction times with 150 instances

<table>
<thead>
<tr>
<th>Hours of the day</th>
<th>Accuracy</th>
<th>Hours of the day</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>68.18%</td>
<td>12</td>
<td>65.00%</td>
</tr>
<tr>
<td>1</td>
<td>38.89%</td>
<td>13</td>
<td>50.00%</td>
</tr>
<tr>
<td>2</td>
<td>63.16%</td>
<td>14</td>
<td>60.00%</td>
</tr>
<tr>
<td>3</td>
<td>55.56%</td>
<td>15</td>
<td>50.00%</td>
</tr>
<tr>
<td>4</td>
<td>44.44%</td>
<td>16</td>
<td>50.00%</td>
</tr>
<tr>
<td>5</td>
<td>72.22%</td>
<td>17</td>
<td>59.09%</td>
</tr>
<tr>
<td>6</td>
<td>60.00%</td>
<td>18</td>
<td>65.22%</td>
</tr>
<tr>
<td>7</td>
<td>50.00%</td>
<td>19</td>
<td>80.95%</td>
</tr>
<tr>
<td>8</td>
<td>57.14%</td>
<td>20</td>
<td>47.62%</td>
</tr>
<tr>
<td>9</td>
<td>52.38%</td>
<td>21</td>
<td>40.91%</td>
</tr>
<tr>
<td>10</td>
<td>45.45%</td>
<td>22</td>
<td>60.87%</td>
</tr>
<tr>
<td>11</td>
<td>59.09%</td>
<td>23</td>
<td>52.38%</td>
</tr>
</tbody>
</table>
Development stage: optimal prediction times

<table>
<thead>
<tr>
<th>Instances</th>
<th>Hour</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.00</td>
<td>13</td>
<td>68.18%</td>
</tr>
<tr>
<td>60.00</td>
<td>8</td>
<td>66.67%</td>
</tr>
<tr>
<td>90.00</td>
<td>17</td>
<td>68.18%</td>
</tr>
<tr>
<td>120.00</td>
<td>16</td>
<td>77.27%</td>
</tr>
<tr>
<td>150.00</td>
<td>19</td>
<td><strong>80.95%</strong></td>
</tr>
<tr>
<td>180.00</td>
<td>8</td>
<td>76.19%</td>
</tr>
<tr>
<td>250.00</td>
<td>11</td>
<td>72.73%</td>
</tr>
<tr>
<td>300.00</td>
<td>4</td>
<td>77.78%</td>
</tr>
<tr>
<td>400.00</td>
<td>5</td>
<td>72.22%</td>
</tr>
<tr>
<td>500.00</td>
<td>2</td>
<td>78.95%</td>
</tr>
<tr>
<td>750.00</td>
<td>2</td>
<td>68.42%</td>
</tr>
<tr>
<td>1,000.00</td>
<td>23</td>
<td><strong>83.33%</strong></td>
</tr>
</tbody>
</table>
Content

1. Introduction
2. Motivation and background
3. In brief: Forex & Neural Networks
4. Reference study and Our initial approach
5. Development stages
6. Evaluation and assessment
   1. First ANN-learning case
   2. Development stages
7. Architecture of application use-case
First ANN-learning case

- Very first model based on case study and other assumptions
- Results obtained: range of 50-54%
- Able to obtain exactly same range of accuracy as the case study
- Result range was as expected as the case study provided result
- Provided us a good foundations for deeper level experiments for future testings
Goals

CHECKLIST:

- Comprehend and apply the Case-study's approach in the Initial design

- Branch-off and find better models using our own techniques

- Apply the model in a tangible application use-case
Development stages

1. Finding optimal neuron-layer setup ⇒ \{4, 16, 64, 32\} with 55.00%
2. Optimizing training-instance settings ⇒ 3,000 training instances with 57% accuracy
3. Intuition of the tests ⇒ Num\_epoch : 250 , Learning rate : 0.0006 , Batch size : 38
4. Optimal prediction times ⇒ \{150 , 19th hr , 80.95\%\} \& \{1,000 , 23rd hr , 83.33\%\}
Goals

CHECKLIST:

- Comprehend and apply the Case-study’s approach in the Initial design
- Branch-off and find better models using our own techniques
- Apply the model in a tangible application use-case
Content

1. Introduction
2. Motivation and background
3. In brief: Forex & Neural Networks
4. Reference study and Our initial approach
5. Development stages
6. Evaluation and assessment
7. Architecture of application use-case
Application use-case

- Python back-end service

- A hybrid mobile app serve for currency predictions and exchange rate
Architecture of the App use-case

**Front-end:**
- Welcome to Xiaox
- The Exchange

**Back-end:**
- Firebase Realtime DB
- Forex-Python API
- Raw currency data .csv
- Python Machine Learning & Currency update scripts
Application Demo

https://xiaoexau.firebaseapp.com/
Goals

CHECKLIST:

- Comprehend and apply the Case-study’s approach in the Initial design
- Branch-off and find better models using our own techniques
- Apply the model in a tangible application use-case
XiaoEx's timeline

- Case study
- Neural Network & Readings
- Data processing
- Testing before building models

First ANN-learning model:
Range of 50-54%

Optimal neuron-layer setup:
\{4, 16, 64, 32\} with 55.00%

Optimizing training-instance:
3,000 with 57% accuracy

Intuitions from all the previous tests

Optimal prediction times:
\{150, 19th hr, 80.95\} & \{1,000, 23rd hr, 83.33\}

XiaoEx, The Exchange Expert
Conclusion

This project has provided us a very interesting insight of the possibilities of Machine Learning.

To be the first machine learning task we think we succeeded in the task given the complexity of the given problem and background knowledge of the members.

We feel this project serves as an excellent foundation to dig deeper into the field of Machine Learning.
Questions & Answers