Big Data Mining

Deep Learning for Finance Big Data with TensorFlow

Min-Yuh Day, Ph.D.
Assistant Professor
Department of Information Management
Tamkang University

http://mail.tku.edu.tw/myday

2018-11-26
### Course Schedule (1/2)

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject/Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2018/09/10</td>
<td>Course Orientation for Big Data Mining</td>
</tr>
<tr>
<td>2</td>
<td>2018/09/17</td>
<td>ABC: AI, Big Data, Cloud Computing</td>
</tr>
<tr>
<td>3</td>
<td>2018/09/24</td>
<td>Mid-Autumn Festival (Day off)</td>
</tr>
<tr>
<td>4</td>
<td>2018/10/01</td>
<td>Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data</td>
</tr>
<tr>
<td>5</td>
<td>2018/10/08</td>
<td>Fundamental Big Data: MapReduce Paradigm, Hadoop and Spark Ecosystem</td>
</tr>
<tr>
<td>6</td>
<td>2018/10/15</td>
<td>Foundations of Big Data Mining in Python</td>
</tr>
<tr>
<td>7</td>
<td>2018/10/22</td>
<td>Supervised Learning: Classification and Prediction</td>
</tr>
<tr>
<td>8</td>
<td>2018/10/29</td>
<td>Unsupervised Learning: Cluster Analysis</td>
</tr>
<tr>
<td>9</td>
<td>2018/11/05</td>
<td>Unsupervised Learning: Association Analysis</td>
</tr>
<tr>
<td>Week</td>
<td>Date</td>
<td>Subject/Topics</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>10</td>
<td>2018/11/12</td>
<td>Midterm Project Report</td>
</tr>
<tr>
<td>11</td>
<td>2018/11/19</td>
<td>Machine Learning with Scikit-Learn in Python</td>
</tr>
<tr>
<td>12</td>
<td>2018/11/26</td>
<td>Deep Learning for Finance Big Data with TensorFlow</td>
</tr>
<tr>
<td>13</td>
<td>2018/12/03</td>
<td>Convolutional Neural Networks (CNN)</td>
</tr>
<tr>
<td>14</td>
<td>2018/12/10</td>
<td>Recurrent Neural Networks (RNN)</td>
</tr>
<tr>
<td>15</td>
<td>2018/12/17</td>
<td>Reinforcement Learning (RL)</td>
</tr>
<tr>
<td>16</td>
<td>2018/12/24</td>
<td>Social Network Analysis (SNA)</td>
</tr>
<tr>
<td>17</td>
<td>2018/12/31</td>
<td>Bridge Holiday (Extra Day Off)</td>
</tr>
<tr>
<td>18</td>
<td>2019/01/07</td>
<td>Final Project Presentation</td>
</tr>
</tbody>
</table>
Deep Learning for Finance Big Data with TensorFlow
Outline

• Deep Learning for Finance Big Data with TensorFlow
  – Deep Learning
  – Finance Big Data
  – TensorFlow
Artificial Intelligence (AI)

Machine Learning (ML)

Supervised Learning

Unsupervised Learning

Deep Learning (DL)

CNN

RNN LSTM GRU

GAN

Semi-supervised Learning

Reinforcement Learning

Source: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/deep_learning.html
Deep Learning
and
Neural Networks
Deep Learning Foundations: Neural Networks
Deep Learning and Neural Networks

Input Layer (X)
Hidden Layer (H)
Output Layer (Y)

X1
X2

Y
Deep Learning and Neural Networks

Input Layer (X) -> Hidden Layer (H) -> Output Layer (Y)
Deep Learning and Neural Networks

Input Layer
\((X)\)

Hidden Layers
\((H)\)

Output Layer
\((Y)\)

Deep Neural Networks
Deep Learning
Deep Learning and Deep Neural Networks
LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton.

"Deep learning."

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

Machine-learning technology powers many aspects of modern society: from web searches to content filtering on social networks to recommendations on e-commerce websites, and it is increasingly present in consumer products such as cameras and smartphones. Machine-learning systems are used to identify objects in images, transcribe speech into text, match news items, posts or products with users' interests, and select relevant results of search. Increasingly, these applications make use of a class of techniques called deep learning.

Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. For decades, conventional approaches to machine learning were based on shallow models, such as single-layer neural networks or linear classifiers. These methods were limited in their ability to capture complex patterns and relationships in the data.

In contrast, deep learning models use multiple layers of processing, allowing them to learn increasingly abstract representations of the input data. Each layer of a deep learning model builds on the representations learned by the layers below it, enabling the model to learn increasingly complex and abstract features.

Deep learning has been successful in a wide range of applications, from image and speech recognition to natural language processing and autonomous driving. It has also been applied to tasks like protein structure prediction and drug discovery, demonstrating its potential for solving complex problems in various domains.

In recent years, deep learning has become a cornerstone of artificial intelligence research, and its impact is expected to continue growing as new applications and techniques are developed.
Deep Learning

• A powerful class of machine learning model
• Modern reincarnation of artificial neural networks
• Collection of simple, trainable mathematical functions
• Compatible with many variants of machine learning

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
What is Deep Learning?

- Loosely based on (what little) we know about the brain

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
Neural Networks (NN)

Source: http://www.asimovinstitute.org/neural-network-zoo/
A mostly complete chart of Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

Source: http://www.asimovinstitute.org/neural-network-zoo/
Convolutional Neural Networks
(CNN or Deep Convolutional Neural Networks, DCNN)


Source: http://www.asimovinstitute.org/neural-network-zoo/
Recurrent Neural Networks (RNN)

Source: http://www.asimovinstitute.org/neural-network-zoo/
Long / Short Term Memory (LSTM)
Gated Recurrent Units (GRU)


Source: http://www.asimovinstitute.org/neural-network-zoo/
Generative Adversarial Networks (GAN)


Source: http://www.asimovinstitute.org/neural-network-zoo/
Support Vector Machines (SVM)


Source: http://www.asimovinstitute.org/neural-network-zoo/
Neural networks (NN)
1960

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
Neural Networks

Input Layer (X)  Hidden Layer (H)  Output Layer (Y)

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
Multilayer Perceptrons (MLP) 1985
Support Vector Machine (SVM) 1995
Hinton presents the Deep Belief Network (DBN)

New interests in deep learning and RBM

State of the art MNIST 2005

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
Deep Recurrent Neural Network (RNN) 2009

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
Convolutional DBN
2010

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
Max-Pooling CDBN 2011
Deep Learning

Geoffrey Hinton
Yann LeCun
Yoshua Bengio
Andrew Y. Ng
A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.
From image to text

Image: deep convolution neural network (CNN)
Text: recurrent neural network (RNN)

A group of people sitting on a boat in the water.

Recurrent Neural Network (RNN)

Course Description

Natural language processing (NLP) is one of the most important technologies of the information age. Understanding complex language utterances is also a crucial part of artificial intelligence. Applications of NLP are everywhere because people communicate most everything in language: web search, advertisement, emails, customer service, language translation, radiology reports, etc. There are a large variety of underlying tasks and machine learning models powering NLP applications. Recently, deep learning approaches have obtained very high performance across many different NLP tasks. These models can often be trained with a single end-to-end model and do not require traditional, task-specific feature engineering. In this spring quarter course students will learn to implement, train, debug, visualize and invent their own neural network models. The course provides a deep excursion into cutting-edge research in deep learning applied to NLP. The final project will involve training a complex recurrent neural network and applying it to a large scale NLP problem. On the model side we will cover word vector representations,
Recurrent Neural Networks (RNNs)
RNN long-term dependencies

I grew up in France... I speak fluent French.

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
RNN LSTM

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Long Short Term Memory (LSTM)

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Gated Recurrent Unit (GRU)

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
LSTM vs GRU

LSTM

i, f and o are the input, forget and output gates, respectively. c and c~ denote the memory cell and the new memory cell content.

GRU

r and z are the reset and update gates, and h and h~ are the activation and the candidate activation.

LSTM Recurrent Neural Network

Source: https://github.com/Vict0rSch/deep_learning/tree/master/keras/recurrent
Long Short Term Memory (LSTM) for Time Series Forecasting
The Sequence to Sequence model (seq2seq)
Sequence to Sequence (Seq2Seq)

Source: https://google.github.io/seq2seq/
Neural Networks

Input Layer (X)  Hidden Layer (H)  Output Layer (Y)

Source: https://www.youtube.com/watch?v=bxET8XR&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
The Neuron

$\mathbf{x}_1 \rightarrow w_1 \rightarrow \mathbf{y}$

$\mathbf{x}_2 \rightarrow w_2 \rightarrow \mathbf{y}$

$\cdots \rightarrow \cdots \rightarrow \mathbf{y}$

$\mathbf{x}_n \rightarrow w_n \rightarrow \mathbf{y}$

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
Neuron and Synapse

Source: https://en.wikipedia.org/wiki/Neuron
The Neuron

\[ y = F \left( \sum_i w_i x_i \right) \]

\[ F(x) = \max(0, x) \]
\[ y = \max (0, -0.21 \times x_1 + 0.3 \times x_2 + 0.7 \times x_3) \]
Neural Networks
Neural Networks

Input Layer (X)  Hidden Layer (H)  Output Layer (Y)

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
Neural Networks

Input Layer (X)

Hidden Layers (H)

Output Layer (Y)

Deep Neural Networks
Deep Learning

Source: https://www.youtube.com/watch?v=bxetV8XR&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
Neural Networks

- **Input Layer** (X)
- **Hidden Layer** (H)
- **Output Layer** (Y)

Source: https://www.youtube.com/watch?v=bxzT-V8XR&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
Neural Networks

Input Layer \( (X) \)  | Hidden Layer \( (H) \)  | Output Layer \( (Y) \)

Source: https://www.youtube.com/watch?v=bxTbxe2T-V8XR&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
Neural Networks

Input Layer (X)  Hidden Layer (H)  Output Layer (Y)

Source: https://www.youtube.com/watch?v=P2HPcj8lRJE&list=PLjJh1vlSEYgvGod9wWiydumYl8hOXixNu&index=2
Neural Networks

Input Layer (X)  Hidden Layer (H)  Output Layer (Y)

X1  X2

Source: https://www.youtube.com/watch?v=bxe2T-V8XR&s=index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
<table>
<thead>
<tr>
<th>Hours Sleep</th>
<th>Hours Study</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>75</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>82</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>93</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>?</td>
</tr>
</tbody>
</table>

Source: [https://www.youtube.com/watch?v=bx2T-V8XRsl&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU](https://www.youtube.com/watch?v=bx2T-V8XRsl&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU)
<table>
<thead>
<tr>
<th></th>
<th>Hours</th>
<th>Hours</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>3</td>
<td>5</td>
<td>75</td>
</tr>
<tr>
<td>Study</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Training</td>
<td>5</td>
<td>1</td>
<td>82</td>
</tr>
<tr>
<td>Testing</td>
<td>10</td>
<td>2</td>
<td>93</td>
</tr>
</tbody>
</table>

Source: https://www.youtube.com/watch?v=bx2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
Y = WX + b

Source: https://www.youtube.com/watch?v=G8eNWzxOgqE
Output

\[ Y = WX + b \]

Input

Weights

Trained

Bias

Source: https://www.youtube.com/watch?v=G8eNWzOgqE
\[ W X + b = Y \]

Scores \rightarrow Probabilities

Source: https://www.youtube.com/watch?v=G8eNWzxOgqE
$W X + b = Y$

$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$

Logits
Scores
Probabilities

Source: https://www.youtube.com/watch?v=G8eNWzxOgqE
\[
S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{2.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{2.0}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.7
\]

\[
S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{1.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{1.0}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.2
\]

\[
S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{0.1}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{0.1}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.1
\]

\[
W X + b = Y
\]

Logits $\rightarrow$ Scores $\rightarrow$ Probabilities

Source: https://www.youtube.com/watch?v=G8eNWzOgqE
Training a Network

= Minimize the Cost Function

Source: https://www.youtube.com/watch?v=bxw2tp8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
Training a Network

= Minimize the **Cost** Function
Minimize the **Loss** Function

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
Error = Predict Y - Actual Y

Error : Cost : Loss

Source: https://www.youtube.com/watch?v=bxev2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7Xev2tg2Ra1PoU
Error = Predict Y - Actual Y

Error : Cost : Loss
Error = Predict Y - Actual Y

Error : Cost : Loss

Source: https://www.youtube.com/watch?v=bxetV8XR&s=index=1&list=PLHhY2iBX9hdHaR6b7XevZtgZRa1PoU
Activation Functions
Activation Functions

Sigmoid

\[ f(x) = \frac{1}{1 + e^{-x}} \]

\([0, 1]\)

TanH

\[ f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \]

\([-1, 1]\)

ReLU (Rectified Linear Unit)

\[ f(x) = \max(0, x) \]
Activation Functions

**Sigmoid**

\[ f(x) = \frac{1}{1 + e^{-x}} \]

**Tanh**

\[ tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \]

**ReLU**

\[ f(x) = \begin{cases} 
0 & \text{for } x < 0 \\
 x & \text{for } x \geq 0 
\end{cases} \]

Source: http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/
Loss
Function
Binary Classification: 2 Class

Activation Function: Sigmoid

Loss Function: Binary Cross-Entropy
Multiple Classification: 10 Class

Activation Function: SoftMAX

Loss Function: Categorical Cross-Entropy
Dropout: a simple way to prevent neural networks from overfitting

(a) Standard Neural Net

(b) After applying dropout.

Learning Algorithm

While not done:

Pick a random training example “(input, label)”
Run neural network on “input”
Adjust weights on edges to make output closer to “label”

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
\[ y = \max (0, -0.21 \times x_1 + 0.3 \times x_2 + 0.7 \times x_3) \]
Next time:

\[ y = \max \left( 0, \ -0.23 \times x_1 + 0.31 \times x_2 + 0.65 \times x_3 \right) \]

\[ y = \max \left( 0, \ -0.21 \times x_1 + 0.3 \times x_2 + 0.7 \times x_3 \right) \]

Weights

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
Optimizer:
Stochastic Gradient Descent (SGD)

\( J(w) \)

Initial weight
Global cost minimum
Gradient
This shows a function of 2 variables: real neural nets are functions of hundreds of millions of variables!
Neural Network and Deep Learning

Source: 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning,
https://www.youtube.com/watch?v=aircAruvnKk
Gradient Descent
how neural networks learn

Average cost of all training data...

\[
\begin{align*}
(0.18 - 0.00)^2 + \\
(0.29 - 0.00)^2 + \\
(0.58 - 0.00)^2 + \\
(0.77 - 0.00)^2 + \\
(0.20 - 0.00)^2 + \\
(0.36 - 0.00)^2 + \\
(0.93 - 0.00)^2 + \\
(1.00 - 0.00)^2 + \\
(0.95 - 1.00)^2 + \\
(0.35 - 0.00)^2
\end{align*}
\]

What’s the “cost” of this difference?

Source: 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning,
https://www.youtube.com/watch?v=iHZwWFHWa-w
Backpropagation

Source: 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, https://www.youtube.com/watch?v=Ilg3gGewQ5U
Learning Algorithm

While not done:

Pick a random training example “(input, label)”
Run neural network on “input”
Adjust weights on edges to make output closer to “label”
Finance Big Data
Python in Google Colab

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

```python
# !pip install pandas_datareader
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline

# Read Stock Data from Yahoo Finance
end = dt.datetime.now()
start = dt.datetime(2016, 1, 1)
df = web.DataReader('AAPL', 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.tail()

df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
plt.plot(df.index, df['Adj Close'], color='blue')
plt.plot(df['Volume'])
plt.set_xlabel('Volume')
plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')

# Simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean()  # 5 days
df['MA20'] = df['Adj Close'].rolling(20).mean()  # 20 days
df['MA60'] = df['Adj Close'].rolling(60).mean()  # 60 days

df = pd.DataFrame({'Adj Close': df['Adj Close'], 'MA05': df['MA05'], 'MA20': df['MA20'], 'MA60': df['MA60']})

df.plot(figsize=(12, 9), legend=True, title='AAPL')
df.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
```
pandas

Python Data Analysis Library

providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Source: http://pandas.pydata.org/
pandas Ecosystem

• **Statistics and Machine Learning**
  – Statsmodels
  – sklearn-pandas

• **Visualization**
  – Bokeh
  – yhat/ggplot
  – Seaborn
  – Vincent
  – IPython Vega
  – Plotly
  – Pandas-Qt

• **IDE**
  – IPython
  – quantopian/qgrid
  – Spyder

• **API**
  – pandas-datareader
  – quandl/Python
  – pydatastream
  – pandaSDMX
  – fredapi

• **Domain Specific**
  – Geopandas
  – xarray

• **Out-of-core**
  – Dask
  – Blaze
  – Odo

pandas-datareader

Up to date remote data access for pandas, works for multiple versions of pandas.

**Warning**

As of v0.6.0 Yahoo!, Google Options, Google Quotes and EDGAR have been immediately deprecated due to large changes in their API and no stable replacement.

**Note**

As of v0.6.0 Google finance is still functioning for historical price data, although there are frequent reports of failures. Failure is frequently encountered when bulk downloading historical price data.

**Usage**

Starting in 0.19.0, pandas no longer supports `pandas.io.data` or `pandas.io.wb`, so you must replace your imports from `pandas.io` with those from `pandas_datareader`:

```python
from pandas.io import data, wb
# becomes
from pandas_datareader import data, wb
```

Many functions from the data module have been included in the top level API.

Get Financial Data Directly into Python

Get millions of financial and economic datasets from hundreds of publishers directly into Python.

Load Quandl Data Directly Into Python

All the Data You Want
Quandl unifies financial and economic datasets from hundreds of publishers on a single user-friendly platform.

Directly Into Python

https://www.quandl.com/tools/python
PyDatastream

PyDatastream 0.5.1

pip install PyDatastream

Python interface to the Thomson Reuters Dataworks Enterprise (Datastream) API

Project description

PyDatastream is a Python interface to the Thomson Dataworks Enterprise (DWE) SOAP API (non free), with some convenience functions for retrieving Datastream data specifically. This package requires valid credentials for this API.

For the documentation please refer to README.md inside the package or on the GitHub (https://github.com/vfilimonov/pydatastream/blob/master/README.md).

https://pypi.org/project/PyDatastream/
pandaSDMX: Statistical Data and Metadata eXchange in Python

pandaSDMX is an Apache 2.0-licensed Python client to retrieve and acquire statistical data and metadata disseminated in SDMX 2.1, an ISO-standard widely used by institutions such as statistics offices, central banks, and international organisations. pandaSDMX exposes datasets and related structural metadata including dataflows, codelists, and datastructure definitions as pandas Series or multi-indexed DataFrames. Many other output formats and storage backends are available thanks to Odo.

Supported data providers

pandaSDMX ships with built-in support for the following agencies (others may be configured by the user):

- Australian Bureau of Statistics (ABS)
- European Central Bank (ECB)
- Eurostat
- French National Institute for Statistics (INSEE)
- Instituto Nacional de la Estadística y Geografía - INEGI (Mexico)
- International Monetary Fund (IMF) - SDMX Central only
- International Labour Organization (ILO)
- Italian statistics Office (ISTAT)
- Norges Bank (Norway)
- Organisation for Economic Cooperation and Development (OECD)
- United Nations Statistics Division (UNSD)
- UNESCO (free registration required)
- World Bank - World Integrated Trade Solution (WITS)
Fred API

FRED® API

General Documentation | API | Toolkits

The FRED® API is a web service that allows developers to write programs and build applications that retrieve economic data from the FRED® and ALFRED® websites hosted by the Economic Research Division of the Federal Reserve Bank of St. Louis. Requests can be customized according to data source, release, category, series, and other preferences.

General Documentation

- Overview
- What is FRED®?
- What is ALFRED®?
- FRED® versus ALFRED®
- Real-Time Periods
- Errors

API

Categories

- fred/category – Get a category.
- fred/category/children – Get the child categories for a specified parent category.
- fred/category/related – Get the related categories for a category.
- fred/category/series – Get the series in a category.
- fred/category/tags – Get the tags for a category.
- fred/category/related_tags – Get the related tags for a category.

https://research.stlouisfed.org/docs/api/fred/
Python Pandas for Finance

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/
! pip install pandas_datareader

Collecting pandas_datareader
  Downloading https://files.pythonhosted.org/packages/cc/5c/ea5b6dcfd0f55c5fb1e37fb45335ec01ccea199b8a79339137f5ed269e0/pandas_datareader-0.7.0.tar.gz (112kB)
100% |████████████████████████████████| 112kB 2.7MB/s

Collecting lxml (from pandas_datareader)
  Downloading https://files.pythonhosted.org/packages/03/a4/9eea8035fc7c7670e5eab97f34ff2ef0ddd78a491bf96df5acc6db0e63f5/lxml-4.2.5-cp38-cp38-manylinux1_x86_64.whl (5.8MB)
100% |██████████████████████████████| 5.8MB 7.5MB/s

Requirement already satisfied: pandas>=0.19.2 in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (0.22.0)
Requirement already satisfied: requests>=2.3.0 in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (2.18.4)
Requirement already satisfied: wrapt in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (1.10.11)
Requirement already satisfied: python-dateutil>=2 in /usr/local/lib/python3.6/dist-packages (from pandas-datareader) (2.8.0)
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from pandas-datareader) (1.14.6)
Requirement already satisfied: idna<=2.7,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas-datareader) (2.6)
Requirement already satisfied: charsetdecode>=3.1,<=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas-datareader) (3.1.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas-datareader) (2018.11.27)
Requirement already satisfied: urllib3<=1.23,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas-datareader) (1.22.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2->pandas-datareader) (1.13.0)

Installing collected packages: lxml, pandas-datareader
Successfully installed lxml-4.2.5 pandas-datareader-0.7.0
conda install pandas-datareader

```
[IMyday-MacBook-Pro:~ imyday$ conda install pandas-datareader
Fetching package metadata .......... 
Solving package specifications: . 

Package plan for installation in environment /Users/imyday/anaconda:

The following NEW packages will be INSTALLED:

   pandas-datareader: 0.2.1-py36_0 
   requests-file:     1.4.1-py36_0 

Proceed ([y]/n)? y

requests-file- 100% |####################################################################################| Time: 0:00:00  1.55 MB/s
pandas-datareader 100% |####################################################################################| Time: 0:00:00  409.66 kB/s

[IMyday-MacBook-Pro:~ imyday$ conda list
# packages in environment at /Users/imyday/anaconda:
#
_license            1.1           py36_1
alabaster           0.7.9         py36_0
anaconda            4.3.1         np111py36_0
anaconda-client     1.6.0         py36_0
anaconda-navigator  1.5.0         py36_0
anaconda-project    0.4.1         py36_0
```
# !pip install pandas_datareader
import pandas_datareader.data as web
import datetime as dt

# Read Stock Data from Yahoo Finance
end = dt.datetime(2017, 12, 31)
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
# !pip install pandas_datareader
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline

#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0),
rowspan=10, colspan=9)
bottom = plt.subplot2grid((12,9), (10,0),
rowspan=2, colspan=9)
top.plot(df.index, df['Adj Close'],
color='blue') #df.index gives the dates
bottom.bar(df.index, df['Volume'])
# set the labels
```
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_ylabel('Volume')
```

```
plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(),
            bins=50, color='purple')
```
# simple moving averages

def['MA05'] = def['Adj Close'].rolling(5).mean()  # 5 days

def['MA20'] = def['Adj Close'].rolling(20).mean()  # 20 days

def['MA60'] = def['Adj Close'].rolling(60).mean()  # 60 days

def2 = pd.DataFrame({'Adj Close': def['Adj Close'], 'MA05': def['MA05'], 'MA20': def['MA20'], 'MA60': def['MA60']})

def2.plot(figsize=(12, 9), legend=True, title='AAPL')

def2.to_csv('AAPL_MA.csv')

fig = plt.gcf()

fig.set_size_inches(12, 9)

fig.savefig('AAPL_plot.png', dpi=300)

plt.show()
# !pip install pandas_datareader
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline

#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()

df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0), colspan=9)
bottom = plt.subplot2grid((12,9), (10, 0), colspan=9)
top.plot(df.index, df['Adj Close'], color='blue') #df.index gives the dates
bottom.bar(df.index, df['Volume'])

# set the labels
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_ylabel('Volume')

plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')

# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
df2 = pd.DataFrame({
'Adj Close': df['Adj Close'],
'MA05': df['MA05'],
'MA20': df['MA20'],
'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True, title='AAPL')
df2.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plt.show()
```python
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt

# Read Stock Data from Yahoo Finance
end = dt.datetime.now()
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()

df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
top.plot(df.index, df['Adj Close'], color='blue') # df.index gives the dates
bottom.bar(df.index, df['Volume'])

# set the labels
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_xlabel('Volume')

plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')

# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean() # 5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() # 20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() # 60 days
df2 = pd.DataFrame({'[Adj Close]': df['Adj Close'], 'MA05': df['MA05'], 'MA20': df['MA20'], 'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True, title='AAPL')
df2.to_csv('AAPL_MA.csv')

fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plt.show()
```
# ! pip install quandl
import quandl
# quandl.ApiConfig.api_key = "YOURAPIKEY"
df = quandl.get("WIKI/AAPL", start_date="2016-01-01", end_date="2017-12-31")
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
Deep Learning with TensorFlow
TensorFlow

An open source machine learning framework for everyone

GET STARTED

Get started with TensorFlow

There are new tutorials to get started with TensorFlow using tf.keras and eager execution. Run the Colab notebooks directly in the browser.

TensorFlow 1.12 is here!

TensorFlow 1.12 is available, see the release notes for the latest updates.

Announcing TensorFlow.js

Learn about our JavaScript library for machine learning in the browser.

https://www.tensorflow.org/
Deep Learning Software

• TensorFlow
  – TensorFlow™ is an open source software library for high performance numerical computation.

• Keras
  – Deep Learning library for TensorFlow, CNTK

• PyTorch
  – An open source deep learning platform that provides a seamless path from research prototyping to production deployment.

• CNTK
  – Computational Network Toolkit by Microsoft Research

Source: http://deeplearning.net/software_links/
tf.keras

Keras: High-level API for TensorFlow
Keras: The Python Deep Learning library

You have just found Keras.

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Use Keras if you need a deep learning library that:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Read the documentation at Keras.io.

Keras is compatible with: Python 2.7-3.6.

http://keras.io/
FROM RESEARCH TO PRODUCTION

An open source deep learning platform that provides a seamless path from research prototyping to production deployment.
Keras

• Keras is a high-level neural networks API
• Written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
• It was developed with a focus on enabling fast experimentation.
• Being able to go from idea to result with the least possible delay is key to doing good research.

Source: https://keras.io/
Google TensorFlow

TensorFlow is an Open Source Software Library for Machine Intelligence

About TensorFlow

TensorFlow™ is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API.

https://www.tensorflow.org/
TensorFlow Playground

Tinker With a Neural Network Right Here in Your Browser. Don’t Worry, You Can’t Break It. We Promise.

http://playground.tensorflow.org/
TensorFlow is an Open Source Software Library for Machine Intelligence

https://www.tensorflow.org/
numerical computation using data flow graphs

https://www.tensorflow.org/
• 3
  – # a rank 0 tensor; this is a scalar with shape []
• [1., 2., 3.]
  – # a rank 1 tensor; this is a vector with shape [3]
• [[[1., 2., 3.], [4., 5., 6.]]
  – # a rank 2 tensor; a matrix with shape [2, 3]
• [[[1., 2., 3.], [[7., 8., 9.]]]
  – # a rank 3 tensor with shape [2, 1, 3]
Nodes:
mathematical operations

edges:
multidimensional data arrays (tensors)
communicated between nodes
Computation is a Dataflow Graph

Graph of Nodes, also called Operations or ops.

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
Computation is a Dataflow Graph

Edges are N-dimensional arrays: Tensors

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
Logistic Regression as Dataflow Graph

Nodes

Operations

Ops

Edges are N-dimensional arrays: **Tensors**

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
Computation is a Dataflow Graph with state

‘Biases’ is a variable

Some ops compute gradients

-= updates biases

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
Data Flow Graph

Source: https://www.tensorflow.org/
Data Flow Graph

Source: https://www.tensorflow.org/
Data Flow Graph

Data Flow Graph
TensorFlow
TensorBoard

Main Graph

Auxiliary nodes

https://www.tensorflow.org/tensorboard/index.html#graphs
Getting Started With TensorFlow

This guide gets you started programming in TensorFlow. Before using this guide, install TensorFlow. To get the most out of this guide, you should know the following:

- How to program in Python.
- At least a little bit about arrays.
- Ideally, something about machine learning. However, if you know little or nothing about machine learning, then this is still the first guide you should read.

TensorFlow provides multiple APIs. The lowest level API—TensorFlow Core—provides you with complete programming control. We recommend TensorFlow Core for machine learning researchers and others who require fine levels of control over their models. The higher level APIs are built on top of TensorFlow Core. These higher level APIs are typically easier to learn and use than TensorFlow Core. In addition, the higher level APIs make repetitive tasks easier and more consistent between different users. A high-level API like tf.estimator helps you manage data sets, estimators, training and inference.

This guide begins with a tutorial on TensorFlow Core. Later, we demonstrate how to implement the same model in tf.estimator. Knowing TensorFlow Core principles will give you a great mental model of how things are working internally when you use the more compact higher level API.
Try your first TensorFlow

$ python

```python
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> sess.run(hello)
'Hello, TensorFlow!'
>>> a = tf.constant(10)
>>> b = tf.constant(32)
>>> sess.run(a+b)
42
```
Hello TensorFlow

```python
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
sess.run(hello)

b'Hello, TensorFlow!'
```

https://github.com/tensorflow/tensorflow
tf.Session()  
sess.run()

```python
import tensorflow as tf
sess = tf.Session()
a = tf.constant(10)
b = tf.constant(32)
sess.run(a+b)
```

42

https://github.com/tensorflow/tensorflow
Linear Regression Model

```python
import tensorflow as tf

# Model parameters
W = tf.Variable([.3], dtype=tf.float32)
b = tf.Variable([-3], dtype=tf.float32)
# Model input and output
x = tf.placeholder(tf.float32)
linear_model = W*x + b
y = tf.placeholder(tf.float32)

# loss
loss = tf.reduce_sum(tf.square(linear_model - y)) # sum of the squares
# optimizer
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)

# training data
x_train = [1, 2, 3, 4]
y_train = [0, -1, -2, -3]
# training loop
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init) # reset values to wrong
for i in range(1000):
    sess.run(train, {x: x_train, y: y_train})

# evaluate training accuracy
curr_W, curr_b, curr_loss = sess.run([W, b, loss], {x: x_train, y: y_train})
print("W: \$s b: \$s loss: \$s"%(curr_W, curr_b, curr_loss))

W: [-0.9999969] b: [ 0.99999082] loss: 5.69997e-11
```

Source: https://www.tensorflow.org/get_started/get_started
import numpy as np
import tensorflow as tf

feature_columns = [tf.feature_column.numeric_column("x", shape=[1])]
estimator = tf.estimator.LinearRegressor(feature_columns=feature_columns)

x_train = np.array([1., 2., 3., 4.])
y_train = np.array([0., -1., -2., -3.])
x_eval = np.array([2., 5., 8., 1.])
y_eval = np.array([-1.01, -4.1, -7, 0.])
input_fn = tf.estimator.inputs.numpy_input_fn(
    {'x': x_train}, y_train, batch_size=4, num_epochs=None, shuffle=True)
train_input_fn = tf.estimator.inputs.numpy_input_fn(
    {'x': x_train}, y_train, batch_size=4, num_epochs=1000, shuffle=False)
eval_input_fn = tf.estimator.inputs.numpy_input_fn(
    {'x': x_eval}, y_eval, batch_size=4, num_epochs=1000, shuffle=False)
estimator.train(input_fn=input_fn, steps=1000)

train_metrics = estimator.evaluate(input_fn=train_input_fn)
eval_metrics = estimator.evaluate(input_fn=eval_input_fn)
print("train metrics: %r" % train_metrics)
print("eval metrics: %r" % eval_metrics)
Deep Learning for Financial Market Prediction
Deep Learning for Financial Market Prediction
Stock Market Prediction
Stock Price Prediction
Time Series Prediction
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/
Long Short Term Memory (LSTM) for Time Series Forecasting

\[ h_{t-2} \rightarrow LSTM \rightarrow LSTM \rightarrow LSTM \rightarrow LSTM \rightarrow LSTM \rightarrow h_{t+2} \]

\[ X_{t-2} \rightarrow LSTM \rightarrow LSTM \rightarrow LSTM \rightarrow LSTM \rightarrow LSTM \rightarrow X_{t+2} \]
### Time Series Data

\[
[10, 20, 30, 40, 50, 60, 70, 80, 90]
\]

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>[10 20 30]</td>
<td>40</td>
</tr>
<tr>
<td>[20 30 40]</td>
<td>50</td>
</tr>
<tr>
<td>[30 40 50]</td>
<td>60</td>
</tr>
<tr>
<td>[40 50 60]</td>
<td>70</td>
</tr>
<tr>
<td>[50 60 70]</td>
<td>80</td>
</tr>
<tr>
<td>[60 70 80]</td>
<td>90</td>
</tr>
</tbody>
</table>
Deep Learning for Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM

Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
Deep Learning for Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM

LSTM for Time Series Forecasting

```python
# univariate lstm example
from numpy import array
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
import matplotlib.pyplot as plt
#%matplotlib inline

# define dataset
X = array([[100, 110, 120, 110, 120, 130, 120, 130, 140, 130, 140, 150, 140, 150, 160]])
y = array([130, 140, 150, 160, 170])
# reshape from [samples, timesteps] into [samples, timesteps, features]
X = X.reshape((X.shape[0], X.shape[1], 1))
# define model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(3, 1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
# fit model
history = model.fit(X, y, epochs=2000, verbose=0)
# demonstrate prediction
x_input = array([150, 160, 170])
x_input = x_input.reshape((1, 3, 1))
yhat = model.predict(x_input, verbose=0)
print('yhat', yhat)
print(model.summary())
# list all data in history
print(history.history.keys())
# summarize history for loss
print('loss:', history.history['loss'][-1])
print('val_loss:', history.history['val_loss'][-1])
plt.plot(history.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.show()

yhat [[181.34615]]
```

Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
Deep Learning for Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM

Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
Deep Learning for Financial Time Series Forecasting

Using TensorFlow backend.

```
[[102.31296]]
yhat [[102.31296]]
```

<table>
<thead>
<tr>
<th>Layer (type)</th>
<th>Output Shape</th>
<th>Param #</th>
</tr>
</thead>
<tbody>
<tr>
<td>lstm_1 (LSTM)</td>
<td>(None, 50)</td>
<td>10400</td>
</tr>
<tr>
<td>dense_1 (Dense)</td>
<td>(None, 1)</td>
<td>51</td>
</tr>
</tbody>
</table>

Total params: 10,451
Trainable params: 10,451
Non-trainable params: 0
```
None
dict_keys(['loss'])
loss: 0.000000
loss: 1.2578432517784677e-07
```

Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
Deep Learning for Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM

Source: https://github.com/yash-1337/AAPL_LSTM_Stock_Predictor/blob/master/AAPL_daily_LSTM_stock_predictor.ipynb
The Quant Finance PyData Stack

- PyThalesians
- Zipline
- DX Analytics
- PyAlgoTrade
- QuantLib

- Quantopian
- PyTables
- NetworkX
- scikits-image
- PyMC

- StatsModels
- matplotlib
- pandas
- SymPy

- SciPy
- NumPy
- Python

Source: http://nbviewer.jupyter.org/format/slides/github/quantopian/pyfolio/blob/master/pyfolio/examples/overview_slides.ipynb#5
AI + VDI

POC
AI + VDI POS

TensorFlow Models

• M1: Basic Classification (Image Classification) (65 Seconds)

• M2: Basic Text Classification (Text Classification) (46 Seconds)

• M3: Basic Regression (Predict House Prices) (43 Seconds)

• M4: Pix2Pix Eager (Option) (7-8 Hours)

• M5. NMT with Attention (Option) (20-30 minutes)
Basic Classification

Fashion MNIST Image Classification

https://colab.research.google.com/drive/19PJOJi1vn1kjcUItlzNHjRSLbeVl4kd5z

Train your first neural network: basic classification

This guide trains a neural network model to classify images of clothing, like sneakers and shirts. It’s okay if you don’t understand all the details, this is a fast-paced overview of a complete TensorFlow program with the details explained as we go.

This guide uses `tf.keras`, a high-level API to build and train models in TensorFlow.

```python
# memory footprint support libraries/code
!ln -s /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
!pip install psutil
!pip install humanize
import psutil
import humanize
import os
import GPUtil as GPU
GPUs = GPU.getGPUs()
gpu = GPUs[0]
def printmem():
    process = psutil.Process(os.getpid())
    print("Gen RAM Free: " + humanize.naturalsize( psutil.virtual_memory().available ), " | Pro
    print("GPU RAM Free: {0:.0f}MB | Used: {1:.0f}MB | Util {2:.0f}% | Total {3:.0f}MB".format
    printmem()
```
Text Classification
IMDb Movie Reviews

https://colab.research.google.com/drive/1x16h1GhHsLrLYtPCvCHaoO1W-i_gror

Copyright 2018 The TensorFlow Authors.
Licensed under the Apache License, Version 2.0 (the "License");
MIT License

Text classification with movie reviews
Download the IMDb dataset
Explore the data
Convert the integers back to words
Prepare the data
Build the model
Hidden units
Loss function and optimizer
Create a validation set
Train the model
Evaluate the model

```
1 # memory footprint support libraries/code
2 !ln -s /opt/bin/nvidia-smi /usr/bin/nvidia-smi
3 !pip install gputil
4 !pip install psutil
5 !pip install humanize
6 import psutil
7 import humanize
8 import os
9 import GPUtil as GPU
10 GPUs = GPU.getGPUs()
11 gpu = GPUs[0]
12 def printg():
13    print(Colors['red'] + 'Please run `./tasks.sh`')
```
Basic Regression
Predict House Prices

https://colab.research.google.com/drive/1v4c8ZHTnRtg2_25K_AURjR6SCVBRdIj

Copyright 2018 The TensorFlow Authors.

Predict house prices: regression

The Boston Housing Prices dataset

Examples and features

Labels

Normalize features

Create the model

Train the model

Predict

Conclusion

Predict house prices: regression

In a regression problem, we aim to predict the output of a continuous value, like a price or a probability. Contrast this with a classification problem, where we aim to predict a discrete label (for example, where a picture contains an apple or an orange).

This notebook builds a model to predict the median price of homes in a Boston suburb during the mid-1970s. To do this, we'll provide the model with some data points about the suburb, such as the crime rate and the local property tax rate.

This example uses the tf.keras API, see this guide for details.

```python
# memory footprint support libraries/code
!ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
!pip install psutil
!pip install humanize

import psutil
import humanize
import os
import GPUtil as GPU
GPUs = GPU.getGPUs()
gpu = GPUs[0]
def printm():
    process = psutil.Process(os.getpid())
    print("Gen RAM Free: ", humanize.naturalsize( psutil.virtual_memory().available ), " | Proc size: ", humanize.naturalsize( process.memory_info().rss ), " | Total ({}GB) | Used ({}GB) | Util {}%".format(gpu.memory.total / (2 ** 30), gpu.memory.used / (2 ** 30), (gpu.memory.util * 100)))
```

Source: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_regression.ipynb
AI+VDI POC
ISAC+TKU Test

• AI+VDI POC Folder (3+1 ipynb) (v3.0.20181120)
  – https://drive.google.com/open?id=1qHOemktbEmUz-ot8eFxIKbGwJvXlrjtc

• run3models.ipynb
  – https://colab.research.google.com/drive/1HQ1GrLqQUUPCct7_AVgoMwMrh0UqMm0f

• AI+VDI POC ISAC+TKU Test Report (Example)
  – https://docs.google.com/spreadsheets/d/1meMwqn15PSuTk6d5TgendDpdDX6L3OfHM4E0Slkq1Zk/edit?usp=sharing
Summary

• Deep Learning for Finance Big Data with TensorFlow
  – Deep Learning
  – Finance Big Data
  – TensorFlow
References

  https://github.com/wesm/pydata-book
• Avinash Jain (2017), Introduction To Python Programming, Udemy,
  https://www.udemy.com/pythonforbeginnersintro/
• Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly
• Michael Heydt (2015), Mastering Pandas for Finance, Packt Publishing
• Michael Heydt (2015), Learning Pandas - Python Data Discovery and Analysis Made Easy, Packt Publishing
• James Ma Weiming (2015), Mastering Python for Finance, Packt Publishing
• Fabio Nelli (2015), Python Data Analytics: Data Analysis and Science using PANDAs, matplotlib and the Python Programming Language, Apress
• Jason Brownlee (2018), How to Develop LSTM Models for Time Series Forecasting,
  https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
• Deep Learning Basics: Neural Networks Demystified,
  https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
• Deep Learning SIMPLIFIED,
  https://www.youtube.com/playlist?list=PLjJh1vIseyGvGod9wWiydumYl8hOxiNu
• 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning,
  https://www.youtube.com/watch?v=aircAruvnKk
• 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning,
  https://www.youtube.com/watch?v=lHzwWFHwa-w
• 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning,
  https://www.youtube.com/watch?v=llg3gGewQ5U
• TensorFlow: https://www.tensorflow.org/
• Keras: http://keras.io/
• Udacity, Deep Learning,
  https://www.youtube.com/playlist?list=PLAwxTw4SYaPn_OWPFT9ulXLuQrlmzHfOV
• http://p.migdal.pl/2017/04/30/teaching-deep-learning.html
• https://github.com/leriomaggio/deep-learning-keras-tensorflow