National and Kapodistrian University of Athens

Department of Informatics and Telecommunications

software development for algorithmic problems

Assignment 3 - Supervised Learning with Neural Networks

Project implemented by team № **59**:

CHARALAMPOS MARAZIARIS - 1115201800105
SPYRIDON CHALKIAS - 1115201800209

Contents

1. Usage

You can have access to project's **parent directory** by typing :

```
$ cd ~/team59_project3/
```

1.1 Prerequisites

In order to run the project, you need to install to your computer the following:

- Python 3
- pip
- Pandas
- numPy
- matplotlib
- seaborn
- Tensorflow
- sklearn
- tqdm

1.2 Project's Structure

The filesystem structure is as follows:

- **/team59_project3/**: Project's main directory.
- **/README.pdf**: Main project's documentation.
- **/src/**: Project's source code.
- **/data/**: Input data used by the project.
- **/out_files/**: Files generated by the compression of time series.
- **/saved_models/**: Saved models, in order to quickly demonstrate their usage without training them.
- **/reports/**: Experiment reports for questions A, B and C individually.
- **/reports/d_comparison_results/ex4_results/**: Comparison of the original dataset versus the compressed dataset using 2nd assignment's executables.
- **/src/preprocess.py**: File containing all the utility functions used by the project's main files.
- **/src/forecast.py**: File used for *time series forecasting.*
- **/src/detect.py**: File used for *time series anomaly detection with LSTM autoencoders.*
- **/src/reduce.py**: File used for *compression of stock market time series using autoencoders.*

- **/src/time_series.ipynb**: The python notebook used in order to train the models and tune the data!
- **/data/nasdaq2007_17.csv**: Data file used in questions A and B.
- **/data/input.csv**: Input file used in question C.
- **/data/query.csv**: Query file used in question C.
- **/out_files/output_dataset_file.csv**: Compressed time series file used as an input file in question D.
- **/out_files/output_query_file.csv**: Compressed time series file used as query file in question D.
- **/saved_models/*_model_trained_on_all_stocks.h5**: Saved forecast, detect and reduce models, trained on all data.
- **/saved_models/*_model_trained_on_AAPL.h5**: Saved forecast and detect models, trained on $AAPL stock.

1.3 Build & Run

While being in the project's parent directory, simply type the following in order to execute each question's corresponding file.

1.3.1 Run A - Time Series forecasting

```
python3 ./src/forecast.py -d <dataset> -n <number of time series selected>
```

1.3.2 Run B - Time Series Anomaly Detection with LSTM Autoencoders

```
python3 ./src/detect.py -d <dataset> -n <number of time series
        selected> -mae <error value as double>
```

1.3.3 Run C - Autoencoders for the compression of stock market time series

```
python3 ./src/reduce.py -d <dataset> -q <queryset> -od
        <output_dataset_file> -oq <output_query_file>
```

2. General Notes

1. The project was written in Python 3, using **Tensorflow** and specifically **Keras API**.
2. The assignment's code was inspired by the three (3) articles provided in the lectures and displayed in the **Resources** section.
3. In order to prevent overfitting, Early Stopping has been added to every model.
4. Each model is being compiled with:

- **Mean Squared Error (MSE)** as a loss function.
- **Adam** as an optimizer.
- **Mean Absolute Error (MAE)** as an evaluation metric.

5. MinMax scaler is used in order to properly scale the data.
6. In *Anomaly Detection*, if the anomaly threshold is not provided by the user, then it is being **automatically computed** by taking the maximum value, when computing the training set's Mean Absolute Error (MAE).

3. Fine-tuning

Fine-tuning reports showcasing our experiments for exercises **A**, **B** and **Γ** can be found in the additional PDFs provided in the submitted directory.

4. Exercise Δ - Comparison with Assignment 2

4.1 Search

- MAF : Maximum Approximation Factor
- AAT : Average Approximation 1-NN Time taken

We observe that our search algorithms run around $\times 100$ faster on the compressed dataset, which is expected. We also notice that our Approximation algorithms run quite well, obtaining scores of perfect **MAF** $= 1$ on the reduced datasets and less than 4 on the original dataset.

Table 1: Original input and query files

Stats	LSH-Euclidean	LSH-Discrete-Frechet	LSH-Continuous-Frechet
MAF	3.43	3.84	2.67
AAT	40.61 sec	3.66 sec	105.39 sec

Table 2: Reduced input and query files

Stats	LSH-Euclidean	LSH-Discrete-Frechet	LSH-Continuous-Frechet
MAF	1	1	1
AAT	0.03 sec	0.01 sec	0.03 sec

4.2 Clustering

4.2.1 Clustering - Mean Vector

We observe that our clustering algorithms run faster on the reduced datasets, as expected, at a factor of at least **20**. We also obtain very good Silhouette scores (> 0.8 on average) in both the clustering of the Original and the Reduced datasets. Thus we could argue that most of the information used to cluster our timeseries is preserved even after their compression, leading to equally good clustering.

Stats	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Overall
Cluster Size	1	21	1	326	349
Silhouette	1	0.24274	1	0.75713	0.72757
Clustering Time : 0.002 sec					

Table 3: Reduced clustering: Lloyd's assignment

Stats	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Overall
Cluster Size	1	21	1	326	349
Silhouette	1	0.44962	1	0.97536	0.97249
Clustering Time : 0.092 sec					

Table 4: Original clustering: Lloyd's assignment

Stats	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Overall
Cluster Size	2	1	1	345	349
Silhouette	0.81551	1	1	0.9105	0.91047
Clustering Time : 0.003 sec					

Table 5: Reduced clustering: LSH

Stats	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Overall
Cluster Size	1	1	1	346	349
Silhouette	1	1	1	0.9769	0.97718
Clustering Time : 0.053 sec					

Table 6: Original clustering: LSH

Stats	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Overall
Cluster Size	1	154	1	193	349
Silhouette	1	0.12298	1	0.71973	0.45801
Clustering Time : 0.002 sec					

Table 7: Reduced clustering: Hypercube

Stats	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Overall
Cluster Size	2	1	1	345	349
Silhouette	0.44962	1	1	0.97536	0.97249
Clustering Time : 0.095 sec					

Table 8: Original clustering: Hypercube

4.2.2 Clustering - Mean Frechet

We observe that our clustering algorithms run very faster on the reduced datasets, at a factor of around **2500**. We also obtain high Silhouette scores in the Reduced datasets, indicating a good clustering.

Stats	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Overall
Cluster Size	264	1	1	83	349
Silhouette	0.6975	1	1	0.1360	0.5657
Clustering Time : 0.054 sec					

Table 9: Reduced clustering: Lloyd's assignment

Stats	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Overall
Cluster Size	345	1	2	1	349
Silhouette	-	-	-	-	-
Clustering Time : 2551.43 sec					

Table 10: Original clustering: Lloyd's assignment

Stats	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Overall
Cluster Size	1	2	1	345	349
Silhouette	1	0.74056	1	0.90767	0.90724
Clustering Time : 0.053 sec					

Table 11: Reduced clustering: LSH

Stats	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Overall
Cluster Size	1	345	1	2	349
Silhouette	-	-	-	-	-
Clustering Time : 3555.52 sec					

Table 12: Original clustering: LSH

5. References

- Forecasting: https://towardsdatascience.com/lstm-time-series-forecasting-predicting-stock-prices-using-an-lstm-model-6223e9644a2f
- Anomaly Detection: https://curiousily.com/posts/anomaly-detection-in-time-series-with-lstms-using-keras-in-python/
- Dimensionality Reduction: https://towardsdatascience.com/autoencoders-for-the-compression-of-stock-market-data-28e8c1a2da3e