ReadMe – Project 2021-22 – Assignment 3

Georgios Georgiou - Γεώργιος Γεωργίου - sdi1800220 - 1115201800220

Panagiotis Mavrommatis - Παναγιώτης Μαυρομμάτης - sdi1800115 – 1115201800115

GitHub Link: https://github.com/blackeye18/LSTM-Project-3-DIT

Στην συγκεκριμένη εργασία χρησιμοποιήσαμε σε μεγάλο βαθμό το google colab σε συνδυασμό με το google drive, με αποτέλεσμα να μην έχουμε πολλά commits στο GitHub.

Τίτλος: Νευρωνικά δίκτυα LSTM – Forecast – Anomaly Detection – Encode – Decode

Τα μοντέλα που παραδίδουμε σε κάθε υποερώτημα έχουν εκπαιδευτεί με όλα τα training sets του dataset, δηλαδή με όλα τα training sets των 360 μετοχών. Οι υπερπαράμετροι με τους όποιους κάναμε train τα τελικά μας μοντέλα είναι οι τιμές που βρίσκονται σε κάθε train_.py αρχείο αντίστοιχα.

Περιεχόμενα φακέλων και τρόπος εκτέλεσης:

A_Forecast:

Best_model_forecast.h5-αρχείο μοντέλου για το A

Forecast.py-αρχείο πρόβλεψης

```
εντολή:$python forecast.py –d <dataset> -n <number of time series selected>
```

Train_forecast.py-αρχείο παραγωγής μοντέλου

```
εντολή: $python train_forecast.py –d <dataset> -n <number of time series
selected> <OPTIONAL: name of model to save>
```

Nasdaq2007_17.csv-αρχείο dataset απο το eclass

B_Detect:

model_detect_best.h5-αρχείο μοντέλου για το B

detect.py-αρχείο πρόβλεψης

```
εντολή:$python detect.py –d <dataset> -n <number of time series selected> -mae
<error value as double>
```

train_detect.py-αρχείο παραγωγής μοντέλου

```
εντολή:$python train_detect.py –d <dataset> -n <number of time series selected>
<OPTIONAL: name of model to save>
```

Nasdaq2007_17.csv-αρχείο dataset απο το eclass

G_Reduce:

encoder1.h5-αρχείο μοντέλου για το Γ

reduce.py-αρχείο εκτύπωσης σε csv αρχεία

```
εντολή:$python reduce.py –d <dataset> -q <queryset> -od <output_dataset_file> -
oq <output_query_file>
```

train_reduce.py-αρχείο παραγωγής μοντέλου

```
εντολή: #$python train_reduce.py –d <dataset> -n <number of time series
selected> <OPTIONAL: name of model to save>
```

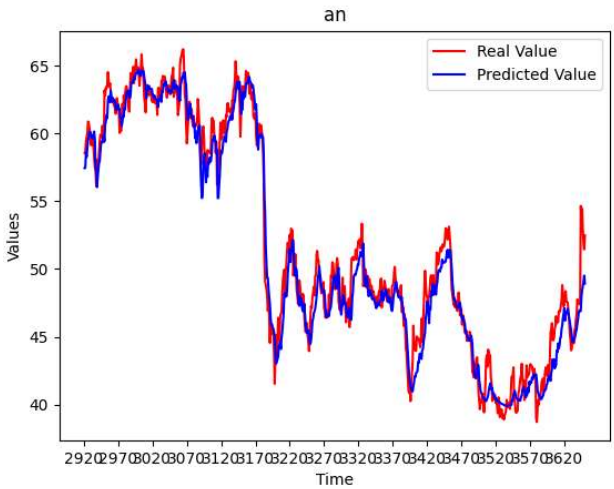
Nasdaq2007_17.csv,nasdaq2007_17_fretchet.csv-αρχεία για πειράματα

Output_dataset_file.csv,output_dataset_file_frechet.csv,output_query_file.csv,output_query_file_frechet.csv-αρχεία που παράγονται από τα αρχεία πειράματος.

Περιγραφή:

A Forecast: Σε αυτό το υποερώτημα αναπτύξαμε αναδρομικό νευρωνικό δίκτυο LSTM πολλαπλών στρωμάτων. Σύμφωνα με τα πειράματα μας, για να έχουμε βέλτιστα αποτελέσματα, μειώσαμε τα epoch στα 50, από 100, και τα units μειώνονται σε κάθε layer(60, 50, 50, 40). Επιπλέον αυξήσαμε κατά 0.1 το dropout στο τελευταίο layer. Καλώντας το forecast.py γίνεται load το μοντέλο και κάνουμε predict χρησιμοποιώντας το τελευταίο 20% των μετοχών. Τα αποτελέσματα εμφανίζονται στο terminal. Επιπλέον εμφανίζονται δειγματοληπτικά, το πολύ 10% του n, γραφικές παραστάσεις. Στο τέλος, ο χρήστης έχει την επιλογή να κάνει train και test κάθε μετοχή σε ξεχωριστό μοντέλο.

Παράδειγμα εκτέλεσης με n=10:

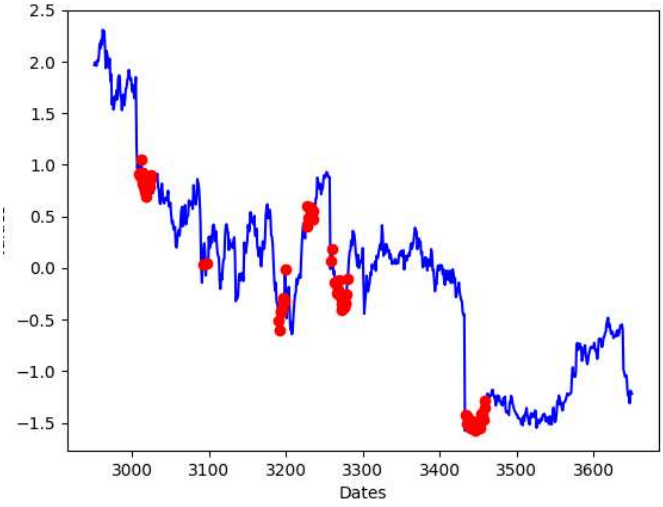



```
(venv) I:\forecast_project3>python forecast.py -d nasdaq2007_17.csv -n 10

Number of rows and columns: (359, 3651)
Now Loading model that has been trained with the whole dataset
2022-01-21 13:58:11.056185: I tensorflow/core/platform/cpu_feature_guard.c
erations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriat
2022-01-21 13:58:11.591555: I tensorflow/core/common_runtime/gpu/gpu_devic
:00.0, compute capability: 5.2
2022-01-21 13:58:14.250325: I tensorflow/stream_executor/cuda/cuda_dnn.cc:
iteration  0 with mean squared error  2.117953637150321
iteration  1 with mean squared error  12.61343336028214
iteration  2 with mean squared error  1.8749319722425444
iteration  3 with mean squared error  2.2356691177159362
iteration  4 with mean squared error  1.4112226068419076
iteration  5 with mean squared error  0.5060431745817476
iteration  6 with mean squared error  0.6528843917798123
iteration  7 with mean squared error  2.082934437536968
iteration  8 with mean squared error  8.831965409677368
iteration  9 with mean squared error  0.4077932773331134
Type 1 to train and test each stock separately or 0 to exit!
0
Finished! :)
```

B Detect:

Σε αυτό το υποερώτημα αναπτύξαμε αναδρομικό νευρωνικό δίκτυο LSTM αυτοκωδικοποιησης χρονοσειρών με στρώματα κωδικοποίησης και αποκωδικοποίησης. Σύμφωνα με τα πειράματα μας για να έχουμε βέλτιστα αποτελέσματα χρησιμοποιήσαμε περισσότερα epoch (25) και τα units μειώνονται σε καθε layer, όπως και στο ερώτημα Α. Τα αποτελέσματα εμφανίζονται στο terminal. Επιπλέον εμφανίζονται δειγματοληπτικά, το πολύ 10% του n, γραφικές παραστάσεις.




```
Iteration:  8
          loss  threshold  anomaly    values  dates
2963  0.412485        0.4     True  0.013084   2963
2966  0.427733        0.4     True  0.096933   2966
2967  0.410175        0.4     True -0.012746   2967
3108  0.402865        0.4     True -1.644430   3108
3115  0.443096        0.4     True -1.919026   3115
Iteration:  9
          loss  threshold  anomaly    values  dates
2968  0.409373        0.4     True -0.666524   2968
2969  0.413227        0.4     True -0.699391   2969
2970  0.406824        0.4     True -0.717244   2970
3084  0.402536        0.4     True -0.391009   3084
3085  0.422638        0.4     True -0.397095   3085
Finished :) No of plots:  1
```

Γ Reduce:

Σε αυτό το υποερώτημα αναπτύξαμε νευρωνικό μοντέλο συμπίεσης και αποσυμπίεσης διαστάσεων (train_reduce.py). Για να το πετύχουμε αυτό χωρίζουμε το σύνολο των τιμών μας σε 10δες στην μέση του μοντέλου(encoder), οι 10δες γίνονται 3αδες και ακολούθως κάνουμε αποσυμπίεση των διαστάσεων, δηλαδή γίνονται και παλι 10δες. Αυτό το κάνουμε έτσι ώστε να μπορούμε να συγκρίνουμε τα δεδομένα στο τέλος με τα αρχικά και να υπολογίσουμε την απόκλιση τους. Όσο μικρότερη απόκλιση τόσο πιο ιδανική συμπίεση γίνεται. Στο τέλος αποθηκεύουμε μόνο τον encoder μιας και αυτός μας χρειάζεται για την συμπίεση. Επίσης, προσθέσαμε ένα debug flag στο train_reduce.py το οποίο όταν είναι ενεργοποιημένο, δηλαδή έχει τιμή 0, εμφανίζονται κάποιες γραφικές παραστάσεις που δείχνουν την αρχική γραφική παράσταση και την γραφική παράσταση μετά από το encode – decode. Τις συγκεκριμένες γραφικές παραστάσεις τις χρησιμοποιήσαμε για την επιλογή των καλύτερων υπερπαραμέτρων.

Στην συνέχεια (reduce.py), κάνουμε load το μοντέλο συμπίεσης και γράφουμε τα συμπιεσμένα δεδομένα σε νέα csv αρχεία, τα οποία χρησιμοποιούμε για το ερώτημα Δ όπως ζητείται.

Δ Σύγκριση Αποτελεσμάτων:

Σχετικά με τον χρόνο εκτέλεσης:

Παρατηρούμε ότι σε όλες τις περιπτώσεις υπάρχει μεγάλη μείωση του χρόνου εκτέλεσης με τα αρχεία που προκύπτουν από το reduce.py σε σχέση με τα αρχικά αρχεία. Πιο συγκεκριμένα, κατά μέσο όρο υπάρχει περίπου υποδεκαπλασιασμός του χρόνου εκτέλεσης. Στην περίπτωση του Search με τον αλγόριθμο LSH_Vector, η μείωση των συντεταγμένων είχε ως αποτέλεσμα να κάνει την πιο συμφέρον τον brute force αλγόριθμο, καθώς επηρεάστηκε περισσότερο από ότι η μέθοδος LSH. Αυτό θεωρούμε πως συμβαίνει λόγω του χρόνου που απαιτείται για την δημιουργία και αρχικοποίηση των δομών του LSH.

Σχετικά με τα αποτελέσματα:

Παρατηρούμε ότι τα αποτελέσματα δεν είναι ακριβώς ίδια και διαφέρουν, συνήθως κατά 10%, είτε προς το καλύτερο είτε προς το χειρότερο. Θεωρούμε πως αυτή η διαφορά προκύπτει από την τυχαιότητα που έχουν οι αλγόριθμοι μας στην $2^{η}$ εργασία, LSH kmeans++ , και δεν έχουν σχέση με την μείωση των συντεταγμένων. Δεδομένου λοιπόν την τυχαιότητα των αλγορίθμων θεώρουμε πως τα αποτελέσματα είναι «ίδια».

Εν κατακλείδι: Λαμβάνοντας υπόψιν τις παραπάνω παρατηρήσεις, η μείωση των συντεταγμένων είχε ως αποτέλεσμα την μείωση του χρόνου εκτέλεσης των προγραμμάτων χωρίς να χάνουμε σημαντική, αν όχι καθόλου, ακρίβεια στα αποτελέσματα.

Σημείωση: Στο clustering με assignment Classic και update MeanCurve καθώς και στο search με algorithm Frechet και metric discrete και continuous, χρησιμοποιήθηκε ένα μικρότερο αρχείο που περιέχει 20 μετοχές. Και σε αυτές τις περιπτώσεις παρατηρήθηκαν ίδιας τάξης μειώσεις.

Clustering: LSH Frechet		
	Before Reduce	After Reduce
Clustering Time:	957.089	37.1045
Total Silhouette:	0.098619	0.118806

Clustering: LSH Vector		
	Before Reduce	After Reduce
Clustering Time:	0.402157	0.10854
Total Silhouette:	0.124351	0.100955

Clustering: Hypercube		
	Before Reduce	After Reduce
Clustering Time:	0.137885	0.032506
Total Silhouette:	0.00605375	0.0604349

Clustering: Classic Vector		
	Before Reduce	After Reduce
Clustering Time:	0.021035	0.003601
Total Silhouette:	[0.99938,-nan,-nan]	0.0244257

Πριν το reduce το πρόγραμμα μας με τον αλγόριθμο Lloyds και update MeanVector έβαζε όλα τα vector στο ίδιο cluster με αποτέλεσμα να μην μπορεί να υπολογιστεί το συνολικό silhouette. Αυτό μάλλον συμβαίνει λόγω του πολύ μεγάλου πλήθους σημείων που έχει κάθε vector στο συγκεκριμένο αρχείο καθώς με κανένα από τα προηγούμενα αρχεία που μας είχαν δοθεί δεν αντιμετωπίσαμε κάποιο παρόμοια πρόβλημα.

Clustering: Classic Frechet		
	Before Reduce	After Reduce
Clustering Time:	15.8839	1.81491
Total Silhouette:	0.552282	0.296283

Search: LSH Vector		
	Before Reduce	After Reduce
tApproximateAverage:	0.00109351	0.000848975
tTrueAverage:	0.00230392	0.000652911

Search: LSH Discrete		
	Before Reduce	After Reduce
tApproximateAverage:	1.04401	0.107782
tTrueAverage:	5.2343	0.439613

Search: LSH Continuous		
	Before Reduce	After Reduce
tApproximateAverage:	1.83276	0.349294
tTrueAverage:	226.35	66.4738

Search: Hypercube Vector		
	Before Reduce	After Reduce
tApproximateAverage:	0.000355827	0.000184423
tTrueAverage:	0.00228811	0.0014192