Node Prediction using Graph-Based KPCA and LDA

1 INTRODUCTION

In this work, we present a methodology in which we create a pipeline to predict node labels of graphs. First, we make dimensionality reduction using graph-based Kernel Principal component analysis (KPCA). Then we use the transformed dataset to feed an Linear discriminant analysis (LDA) classifier. For each graph we want to predict all its nodes, so we use a multi-output LDA. We make experiments with two datasets. First, we use the parking violation data of Thessaloniki's controlled parking system and we predict the parking violation rate of sectors. Then, we used the chickenpox dataset and tried to predict disease outbreaks. The problems we are going to solve concern regression tasks. However, we wanted to compare LDA with our previous works and that is why we turned our problem into classification. The range of target values helped us to convert the targets into classes however the metrics we use are used in the regression task. Finally, we make additional experiments for both datasets using common KPCA methods as well as the K-Nearest Neighbor algorithm.

2 DATA

We use parking violation and chickenpox datasets, However, the datasets and the processing process are the same as our previous work. We also kept the same train-test splits to have a fair comparison.

3 GRAPH KERNEL PCA

We use kernels to calculated the metric space distances between the graphs of the dataset and produced an adjacency matrix with shape of (N_{graphs}, N_{graphs}). Then we applied dimensionality reduction to this adjacency matrix with precomputed KPCA. We tried various kernels to measure the distances between graphs such as Shortest-Path [1], but we found that PropagationAttr [4] is more efficient for our problem. In fact, with this kernel, we do not measure the distances between the graphs, but the similarities. We will not describe the kernel in detail as we have done so in our previous work.

4 MULTI-OUTPUT LDA CLASSIFIER

Since we performed dimensionality reduction we entered the transformed data into an LDA classifier. For each graph we predict all its nodes using a multi-output LDA. The given input to the LDA mode has the shape of $(N_{graphs}, N_{components})$ while the output has the shape of $(N_{graphs}, N_{sectors})$.

5 EXPERIMENTS ON PARKING DATASET

In this chapter, we perform several experiments for Parking dataset to optimize the choice of hyper-parameters for each model of our pipeline. Then we compare our model with other approaches.

5.1 Graph KPCA and LDA Tuning

First, we looked for the best parameters for the KPCA model which is the first part of our Pipeline. The parameters of PropagationAttr were found in our previous work. Since we use precomputed kernel we will focus more on finding the best N-components of KPCA. First, we found the number of non-zero components. Then we tried all N-components starting from 1 up to the number of non-zero components. For each of these, we measured our performance on the test set. Where we use Mean Absolute Error (MAE) and Mean Squared Error (MSE) metrics. It is worth noting that during these experiments the LDA was trained with the default parameters. Fig 1 shows the change of the metrics in the range of N-components.

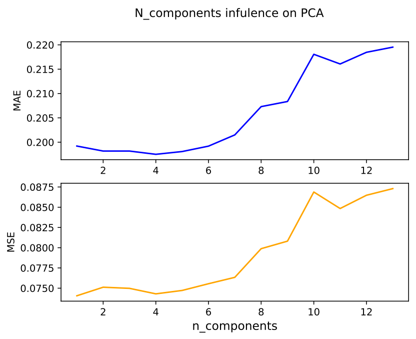

Figure 1: Influence of n-components

As is obvious, the most effective option is 4-components. Knowing the best parameters for the first model of the pipeline, we did several experiments to optimize LDA model, we use different solvers and hyper-parameters while evaluate on test set. Table 1 describes the results of our experiments.

Table 1: Experiments using different hyper-parameters of LDA.

Solver	Store-Covar.	Tol	Shrinkage	MAE	MSE
Svd	**False**	**0.5**	-	**0.19737**	**0.07420**
Svd	True	0.5	-	0.19737	0.07420
Svd	False	0.05	-	0.19751	0.07430
Svd	True	0.05	-	0.19751	0.07430
Lsqr	-	-	None	0.19759	0.07436
Eigen	-	-	None	0.19759	0.07436
Lsqr	-	-	Auto	0.19761	0.07434
Eigen	-	-	Auto	0.19761	0.07434
Lsqr	-	-	0.2	0.19771	0.07455
Eigen	-	-	0.2	0.19771	0.07455
Lsqr	-	-	0.4	0.19803	0.07477
Eigen	-	-	0.4	0.19803	0.07477

Looking at the results we notice that some parameter choices do not affect the performance of our model, however, the best LDA model has fitted with Solver = Svd, Store-Covariance = False, and Tol = 0.001. Metrics calculated while predicting the test set is MAE = 0.19751 and MSE = 0.07430.

5.2 Comparison with usual PCA Kernels

To validate the use of graph kernels we compare our pipeline with a pipeline that contains non-graph-based PCA kernel methods. For this reason, we solved the problem as a simple classification task, without using the graph information (edges and edge weight). For the following procedure, it was impossible to use the entire data set as it consists of 500000 samples that did not fit in the computer's memory. For this reason, we used a small subset of 25000 samples. However, we experimented with the same subset using the graph-based KPCA pipeline to have a fair comparison. The methodology we followed is as follows. First, we find the best n-components using explained variance metric with a 95% cut-off. Fig 2 shows the plot we use to find the best n-components.

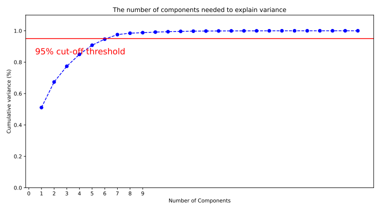

Figure 2: The number of components needed to explain variance (Parking Dataset)

Then we applied Grid Search 4-fold cross-validation (CV) with 3 different kernels (RBF, Sigmoid, Poly) and various combinations of KPCA hyper-parameters however all CV experiments have been done using 5 components (LDA was trained using the default parameters). Table 2 shows the results of 4-fold cross-validation while negative mean absolute error is used as a metric. The table is also sorted from the best model to the least efficient.

Table 2: Experiments using different hyper-parameters of KPCA (Subset of Parking Data).

N-Comp	Kernel	Gamma	Degree	Coef0	Neg. MAE
5	**Sigmoid**	**0.3**	-	**2**	**-11.01430**
5	Sigmoid	0.5	-	1	-11.18880
5	Sigmoid	0.3	-	1	-11.19091
5	Sigmoid	0.4	-	2	-11.19654
5	Sigmoid	0.4	-	1	-11.21419
5	Sigmoid	0.5	-	2	-11.24626
5	Rbf	0.3	-	-	-11.76463
5	Rbf	0.4	-	-	-12.13923
5	Rbf	0.5	-	-	-12.13923
5	Poly	0.5	3	-	-12.85231
5	Poly	0.3	5	-	-12.87014
5	Poly	0.4	3	-	-12.87014
5	Poly	0.5	5	-	-12.88829
5	Poly	0.4	5	-	-12.89434
5	Poly	0.3	3	-	-12.89434

Analyzing the results, we understand that the Sigmoid kernel solves our problem better compared to the rest, so the best model is the one with 5 components, kernel = Sigmoid, Gamma = 0.3, and Coef0 = 1. Knowing the best parameters for the first model of the pipeline, we kept them and did several experiments to optimize the LDA model, we use different solvers and hyper-parameters while evaluating on the test set. Table 3 describes the results of our experiments.

Table 3: Experiments using different hyper-parameters of LDA (Subset of Parking Data).

Solver	Store-Covar.	Tol	Shrinkage	MAE	MSE
Lsqr	-	-	**0.4**	**0.17291**	**0.04733**
Eigen	-	-	0.4	0.17291	0.04733
Lsqr	-	-	0.2	0.17417	0.04938
Eigen	-	-	0.2	0.17417	0.04938
Svd	False	0.5	-	0.17427	0.04782
Svd	True	0.5	-	0.17427	0.04782
Svd	False	0.05	-	0.18588	0.05645
Svd	True	0.05	-	0.18588	0.05645
Lsqr	-	-	Auto	0.18618	0.05658
Eigen	-	-	Auto	0.18618	0.05658
Lsqr	-	-	None	0.18722	0.05728
Eigen	-	-	None	0.18722	0.05728

Looking at the results we notice that some parameter choices do not affect the performance of our model, however, the best LDA model has fitted with Solver = Lsqr, and Shrinkage = 0.4. We use the subset and trained this pipeline (KPCA + LDA) while testing on the raw test set to compare it with the pipeline we propose in the present work (PropagationAttr PCA + LDA), this comparison is shown in Table 4. In addition, the results obtained using K-Nearest Neighbor (KNN) algorithm are also shown.

Table 4: Experimental comparison between model pipelines on subset of Parking Data (training time is described in seconds).

Pipeline	MAE	MSE	T. Time
PropagationAttr PCA + LDA	**0.16913**	**0.04621**	**63**
KPCA + LDA	0.17291	0.04733	9
101-Nearest Neighbor	0.17538	0.04723	2
Evaluation on Training set			
PropagationAttr PCA + LDA	0.9432	0.01852	63
KPCA + LDA	0.10295	0.0223	9
101-Nearest Neighbor	**0.07839**	**0.01468**	53

6 EXPERIMENTS ON CHICKENPOX DATASET

In this chapter, we perform several experiments for the Chickenpox dataset to optimize the choice of hyper-parameters for each model of our pipeline. Then we compare our models with other approaches. We do not show all the results for the hyper-parameter optimization experiments in order not to make the report long. However, the results are visible in the code.

6.1 Graph KPCA and LDA Tuning

We applied the same methodology that we used above for the parking dataset. We do various experiments to optimize the hyper-parameters of used models. Finally, our best pipeline has a PropagationAttr KPCA with 2 components, and a Multi-output LDA with solver = SVD, store-covariance = False, and Tol = 0.5. The calculated metrics on the test set were MAE = 0.12731 and MSE = 0.02792.

6.2 Comparison with usual PCA Kernels

After we finished optimizing the hyper-parameters we wanted to compare our work with some of the usual non-graph-based PCA kernel methods. We tested various PCA kernels through a grid search. Finally, our best pipeline has a KPCA model that was fitted with kernel = Poly, degree = 5, gamma = 0.4, and n-components = 3. Then pipeline has a Multi-output LDA with solver = SVD, store-covariance = False , and tol = 0.5. The calculated metrics on the test set were MAE = 0.11590 and MSE = 0.02538. Table 5 summarizes the results of various pipelines tested for the Chickenpox dataset. In addition, the results obtained using the K-Nearest Neighbor (KNN) algorithm are also shown.

Table 5: Experimental comparison between model pipelines on Chickenpox Data (training time is described in seconds).

Pipeline	MAE	MSE	T. Time
PropagationAttr PCA + LDA	0.12731	0.02792	20
KPCA + LDA	**0.11590**	**0.02538**	**10**
101-Nearest Neighbor	0.12231	0.02651	2
Evaluation on Training set			
PropagationAttr PCA + LDA	0.07223	0.01306	20
KPCA + LDA	0.08639	0.01548	10
101-Nearest Neighbor	**0.06336**	**0.00960**	**2**

7 SUMMARY OF RESULTS

In this chapter, we summarize the results we measured by testing various methods on the 2 datasets we used. As mentioned above, we were unable to apply one of our methods to the parking dataset due to the small system memory we have. For this reason, we used a subset of the data and again performed all experiments with it so that we could compare all methods.

7.1 Results on Full Parking Dataset

Table 6 contains the results we measured across all our methods for the full Parking dataset. In all cases, we evaluated the methods using only the raw test set. The measurements for the Sigmoid KPCA + LDA method were not possible to perform as more computing memory is required than is available. At this point, I would like to mention that the Residual DNN method has been developed in our previous work and published in [2] and [3]. The TGCN + LSTM method has also been developed by us in our previous work which is based on graph neural networks and we hope it will be published soon in Expert Systems with Applications journal.

7.2 Results on Subset of Parking Dataset

As mentioned above, we used a small subset (5% of the data) and reran all experiments with it so that we could compare all methods. Table 7 describes the results for this subset.

7.3 Results on Chickenpox Dataset

Table 8 contains the performance results of all methods on the chickenpox dataset.

REFERENCES

[1] K.M. Borgwardt and H.P. Kriegel. 2005. Shortest-path kernels on graphs. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*. 8 pp.–. https://doi.org/10.1109/ICDM.2005.132

[2] Nikolaos Karantaglis, Nikolaos Passalis, and Anastasios Tefas. 2022. Deep Learning for On-Street Parking Violation Prediction. (2022), 1–5. https://doi.org/10.1109/IVMSP54334.2022.9816222

[3] Nikolaos Karantaglis, Nikolaos Passalis, and Anastasios Tefas. 2022. Predicting on-street parking violation rate using deep residual neural networks. *Pattern Recognition Letters* 163 (2022), 82–91. https://doi.org/10.1016/j.patrec.2022.09.023

[4] Giannis Siglidis, Giannis Nikolentzos, Stratis Limnios, Christos Giatsidis, Konstantinos Skianis, and Michalis Vazirgiannis. 2020. GraKeL: A Graph Kernel Library in Python. *Journal of Machine Learning Research* 21, 54 (2020), 1–5.

Table 6: Summary of results for full Parking dataset (training time is described in minutes).

Method	MAE	MSE	T. Time
TGCN + LSTM	**0.16620**	**0.0468**	**49**
SVM PropagationAttr Conv Kernel	0.18913	0.069602	23
SVM PropagationAttr kernel	0.18996	0.069753	16
PropagationAttr PCA + LDA	0.19751	0.07430	33
Residual DNN	0.19910	0.06810	39
101-Nearest Neighbor	0.20062	0.07706	3
SVM RBF Kernel	0.20653	0.08223	80
Sigmoid KPCA + LDA	-	-	-

Table 7: Summary of results for subset of Parking dataset (training time is described in minutes).

Method	MAE	MSE	T. Time
TGCN + LSTM	**0.13013**	**0.03555**	**4**
SVM PropagationAttr Conv Kernel	0.15813	0.04320	1
SVM PropagationAttr kernel	0.15895	0.04343	1
PropagationAttr PCA + LDA	0.16913	0.04621	2
Residual DNN	0.17152	0.04686	3
Sigmoid KPCA	0.17291	0.04733	1
101-Nearest Neighbor	0.17538	0.04723	1
SVM RBF Kernel	0.17612	0.04812	3

Table 8: Summary of results for Chickenpox (training time is described in seconds).

Method	MAE	MSE	T. Time
SVM RBF Kernel	**0.11455**	**0.02227**	**4**
Poly KPCA + LDA	0.11590	0.02538	10
101-Nearest Neighbor	0.12231	0.02651	2
SVM PropagationAttr Conv kernel	0.12324	0.02632	24
SVM PropagationAttr Kernel	0.12336	0.02647	19
PropagationAttr PCA + LDA	0.12731	0.02792	20