UCLouvain

LINMA2472 - Algorithms in Data Science

Embeddings - K-PCA and Random Projections and Random Fourier Features (RFF)

Team Member Names:
Sarra Laksaci - sarra.laksaci@student.uclouvain.be
Mojgan Giahi - mojgan.giahi@student.uclouvain.be
Sixto Castro Redrobán - sixto.castro@student.uclouvain.be

Teachers:
Jean-Charles Delvenne
Gautier Krings
Course Assistants:
Rémi Delogne
Loïc van Hoorebeeck
Nizar Bousselmi

A report submitted in partial fulfilment of the requirements of
the UCLouvain for the degree of
Master of Science in *Computer Science* and *Data Science*

November 17, 2021

Abstract

In this Embedding project we could perform EDA. We computed the outlier ratio, plotted the true points and the outliers. Next, we performed outlier detection using PCA. Then we performed K-PCA and selected the best models among PCA and K-PCA. In the second part, we trained a linear SVM, trained a Gaussian Kernel SVM, and implemented the RFF. Finally, we computed the influence of parameter D on different metrics.

Part 1: K-PCA and Random Projections

Q1. EDA (Exploratory Data Analysis)

1.1 Outliers ratio

In the following equation we denote N_o as the number of outliers in the data and N_t as the number of true data points

$$r_0 = \frac{N_o}{N_t + N_o}$$

The given data regroups a total on 500 data points of which 400 are true points and a 100 are outliers according to the corresponding y value

$$r_0 = 0.2$$

1.2 Data visualization

The figure 1 is a visualization of the data points on which we can see the true points as green dots and outliers as red dots. We can first observe that the shape formed by the true data resembles a dismenteled square as the green point are spread across 4 clusters. Outliers on the other hand are scattered across the plan. Majority of the red points are located inside the boundaries of the visible square and the smaller part is spreaded outside the green lines. We can also notice that some of the red points (8 points) overlap with 2 of the 4 green clusters formed by true data.

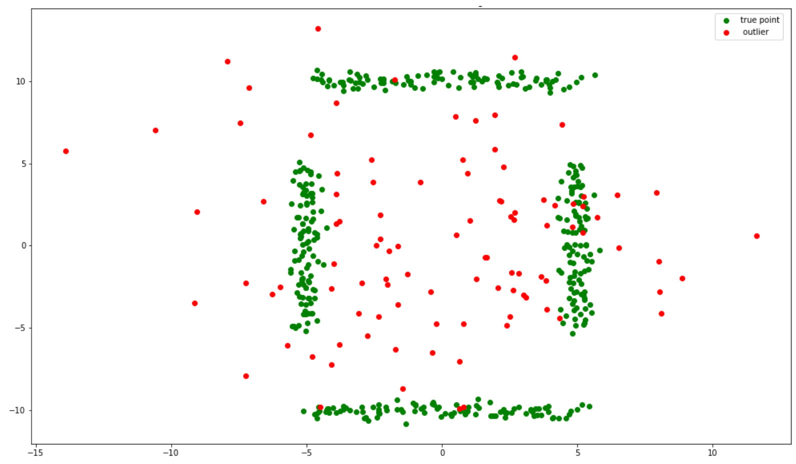


Figure 1: True data vs outliers plot

Given the poor class separability observed, we can assume that the task of separating both true data and outliers will not be an easy task. Specially as the the true point do not form a solid shape but rather small disjuncts with overlapping spars outlier points.

1.3 Minimal score

$$Score = \frac{TP + TN}{N + P}$$

The worst acceptable outlier detector is the random classifier which has a 0.5 probability of classifying a TN and 0.5 probability of classifying a TP Thus we assume that the lowerbound for the score criterion is 0.5

Q2. Train/Test/Validation

We splitted the data into training, validation and test data with a 90/10 split between 2 first categories and the test category and a 70/30 split between train set and validation set. Before splitting the data according to the given ratio we performed a shuffle of the data points. Indeed shuffling the data is necessary for the given data because the original data file regroups the outliers at the end. If we do not shuffle, the training data will cover true data only whereas test data will only contain outlier. The previous scenario would lead to a poor classification

Q3. PCA (Principal Components Analysis)

- **score:** Our PCA classifier with k=1 returns an acceptable score as it exceeds the lowerbound of 50%

$$score(X\_val \ \ , \ y\_val) = 0.84 \tag{1}$$

$$score(X\_test \ \ , \ y\_test) = 0.74 \tag{2}$$

- **parameters/ hyperparameters:** The first hyperparameter of our model is K the size of the eigen subspace. As the original data X is of shape (nb_sample , 2) this means that $k \in \{1, 2\}$. An OutlierDetectorPCA of 2 eigen vector returns a score of 0.81 on the validation data which is worse than our original model of k=1. As for the parameters we have the threshhold τ which is learned throught the fitting of the model.
  The reconstruction matrix is calculated through the euclidean distance between Xpca and XpcaHat (the projection of Xpca to the 1st component). Once this matrix is built, its maximum and minimum values are got, and then τ is calculated as follows:
  max(abs(min(recons_error)),abs(max(recons_error))

Q4. k-PCA (Kernel PCA)

- **fit function :** The fit function of the OutlierDetectorKPCA aims to fit the model to the true data. To do so it takes as input the data X and its true labels y and returns the classifier. To explain the inner working of this function we first remind you that the goal here is to perform PCA in the feature space while using kernel tricks to avoid explicit computations of mapping function ϕ. This is why we first apply the kernel transformation specified by the param attribute (in our case Gaussian kernal with $\gamma = 0.01$) then feed this *new data*,which is really just our input lifted to the high dimensional feature space, to the OutlierDetectorPCA 's fit function.
- **value of k:** In the OutlierDetectorKPCA classifier the value k range between 1 and the number of features of the data. Originally the data has 2 features, however in KPCA we worked in a higher dimensional feature space which in our case regroups a number of 84 features. $K \in [1..84]$

- **Hyperparameters :** As we work with a KPCA model we first and formost have the K hyper-parameter which represents the number of eigen vectors. We also have the Kernel parameters which we can finetune: gamma value and kernal passed through the param dictionary.
- **parameters :** The parameters of this model are the threshhold τ

Q5. Model selection

- **parameters and optimal combination:** As mentioned in the previous section, the OutlierDetectorKPCA has hyperparameters K, kernel and γ. The first one k is the number of eigen vectors, experimentation shows that closest model to the perfect classifier is of value k=5. The second parameter kernel type between Radial Basis kernel function and polynomial kernel defines the computation applied to the input data, experimentation hold better results with kernel='rbf. Lastly the gamma parameter assigned to the kernel defines how far the influence of a single training example reaches. the best experimentation model was set to $\gamma = 0.001$. Best OutlierDetectorKPCA was defined by K=5, 'rbf' kernel and $\gamma = 0.001$.
- **ROC curve for PCA and best KPCA:** The plot of PCA and best KPCA roc shows a major difference of results as the pca classifier roughly approaches the random classifier and even goes bellow it. The best OutlierDetectorKPCA model grows towards the top_left corner near the perfect classifier point way above the random classifier dashline. [figures available in notebook pdf]

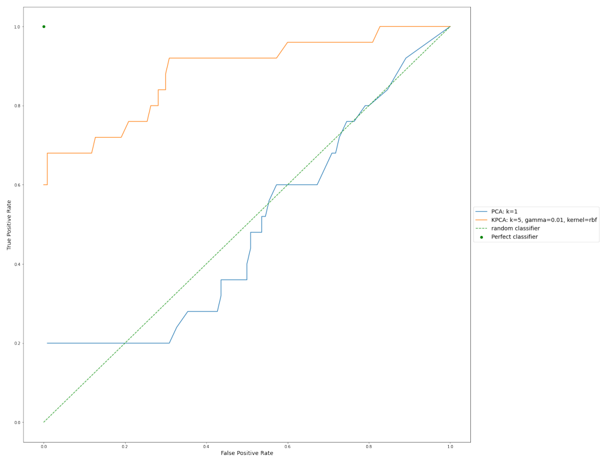


Figure 2: PCA vs best KPCA

- **Outlier detection comparision:** The below plot shows the outlier detection capacities of two models: pca model on the left and kpca on the right. We can see that the pca model approaches the general shape of the true data (hourglass purple shape) however this dense shape covers a large number of outliers, the outliers of the middle. Based on this description we can say that the left model fails to properly detect outliers. The right model is the kpca model which reaches a better classification as the purple shape of true data forms a donut-like object that perfectly overlaps with true data and leaves out majority of the outliers on the green space.

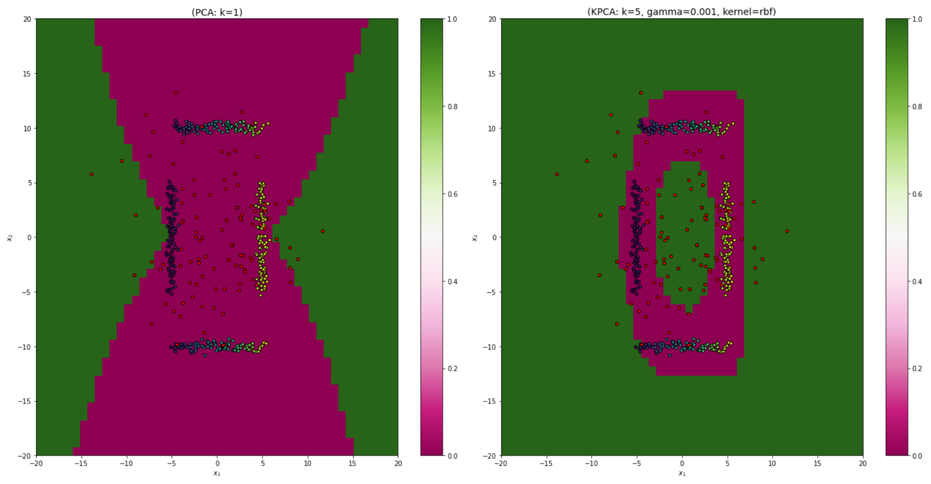


Figure 3: plot_comparision_outlier_detection for pca and best kpca

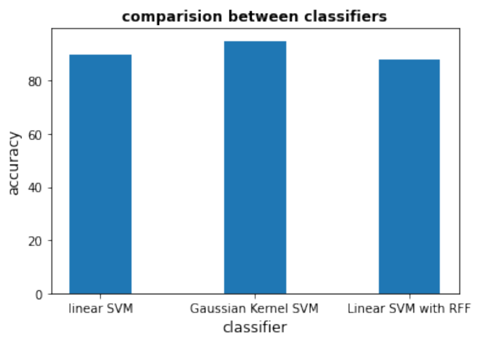
Q6. Analysis

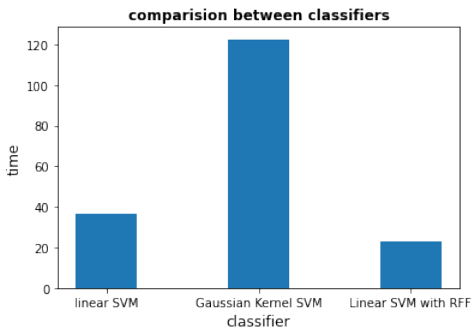
- **generalization error of best model on test data:** Plotting the outlier detection performance of the best model on test data shows a good generalisation as the purple color includes all true data, however red point with a short distance to true data region overlap with the purple color instead of the green one this shows a flaw in the model's performance.
- **very far data point:** if we take into consideration the reconstruction error on the pca model we can see that the linearity of this method fails to detect outliers that are very far from the bulk of data for example a point that would reach infinity following the principal component it stays in the purple zone thus labeled as true data with 0 reconstrucion error. For our best model however, the 0 reconstruction error zone only follows the shape of the true data on which we fitted the classifier (aka the donut shape) so the very far data point are given a high reconstruction error.

Part 2: Random Fourier features (RFF)

Q1. Testing time

- **Linear SVM** Accuracy:0.90 Total time: 36.41
- **Gaussian Kernel SVM** Accuracy: 0.95 Total time: 122.56
- **Linear SVM on RFF data** Accuracy: 0.88 Total time: 23.06



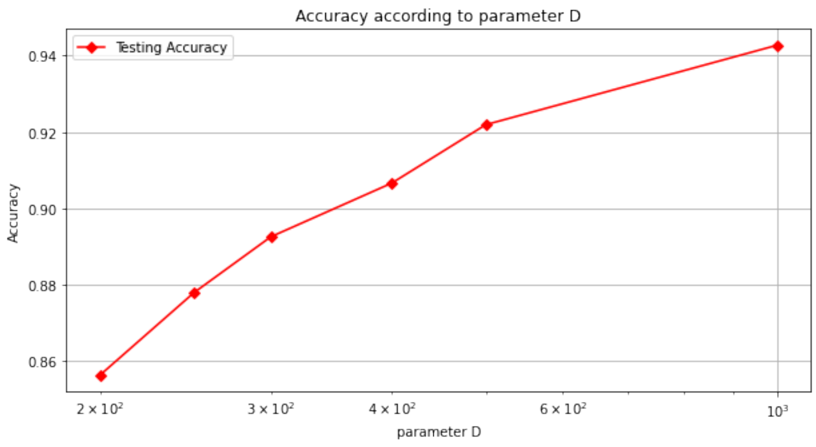


The two histograms above represent the accuracy achieved by the three classifiers and the prediction time registered by these same classifiers. We notice that the accuracy is good for all three methods: Linear SVM, Gaussian Kernel SVM, Linear SVM with RFF as they have over 80% accuracy. More specifically the 2nd classifiers hits the highest value of 95% whereas the linear SVM has a 90% value and the Linear SVM a 88% acc (Note that these were results based on the given parameters). Althought the accuracies are roughly similar for all three classifiers, the prediction time is very different. We can spot that the longest prediction time is attributed to the Gaussian kernal SVM with a value of over 120secondes as the Linear SVM with RFF has a prediction time of a little over 20secondes. This $t(Classifier2) >>> t(Classifier3)$ observation is a proof that the intended goal of Random Fourier Feature has been reached. Indeed we remind you that the purpose of RFF is to explicitly approximate the feature space hoping to reduce the time complexity of the procedure.

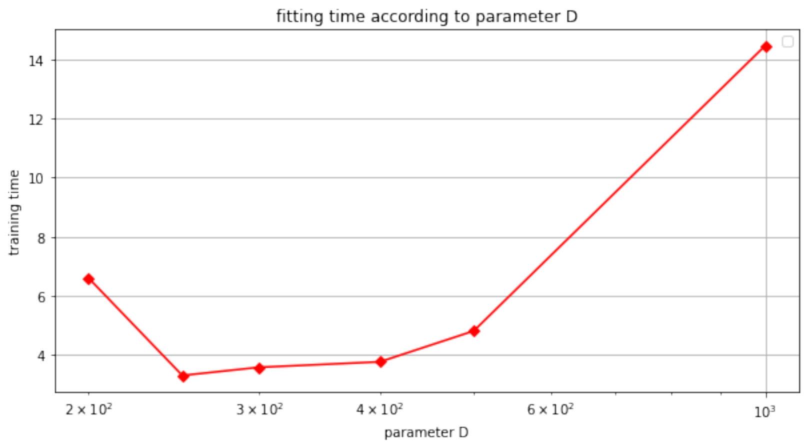
Q2. Training time and parameters

To further analyse the influence of the parameter D on the Linear SVM with RFF method we conducted a number of experiments which consisted in generating ω and b based on and increasing value of D. The generated parameters were used to train and infer the model.

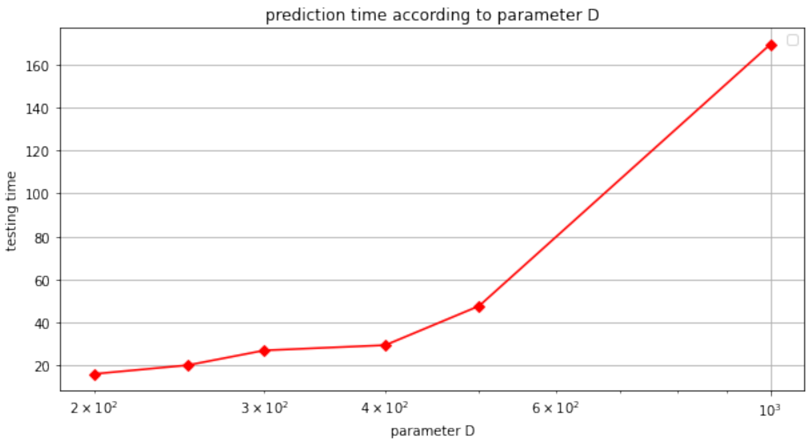
- **The influence of the parameter D on the accuracy:** The below figure shows the different accuracies reached by a Linear SVM with different numbers of Random Features. We observe an increase of the accuracy following the increase of D. The maximum accuracy of over 94% was reached with D=1000. But the curve of the plot leads us to believe that this can be further improved with a bigger D.



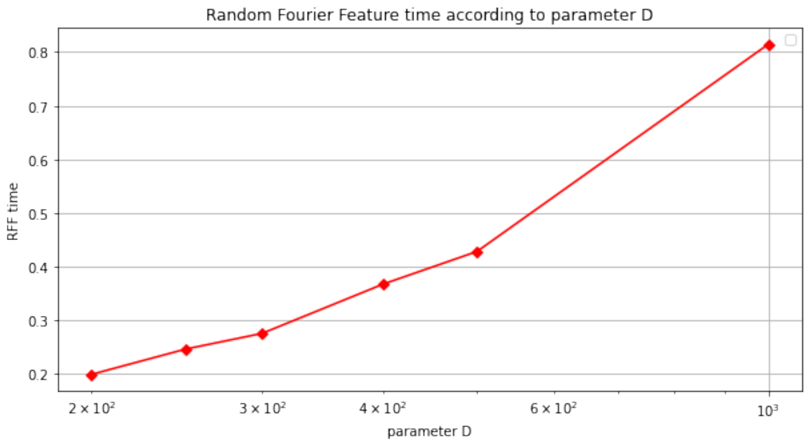
- **The influence of the parameter D on the train time:** Linear SVM with RFF: The following graph displays the growth of the training time with respect to the number of random features. We can see that with a parameter $D \in [250, 1000]$ the behavior of the training time is as we would expect i.e the higher the value D the more time is takes to fit the classifier because the size of the matrices is bigger thus the calculations are more complex. However, we observe that for D=200 the training time is bigger than of D=250, which leads to a down slope from values D¡250 up to D=250.



- **The influence of the parameter D on the prediction time** The plot of the testing time with respect to the parameter D shows that as D grows, the prediction time increases. Specifically we notice a slow growth from 20s to 50s in $D \in [200, 500]$, this is followed by a rapid increase that reaches over 160sec for the biggest value of D. We can also note that the observation made for train time down slope when D¡250 is not noticed in the prediction step.



- **The influence of the parameter D on the RFF time:** a first observation is that the rff time for each classifier is much smaller than the respective time of its training or prediction. Apart from the difference of time values, we can compare the overall plot shape of the rtt time to the prediction time. Indeed the same behavior is noted i.e a slow growth at first for $D \in [200, 500]$ and sharp increase with a maximum time value for the biggest value of D.


Random Fourier Feature time according to parameter D
RFF time
2 × 10²
3 × 10²
4 × 10²
6 × 10²
10³
parameter D