UCLouvain

epl

LINMA2472 – Algorithms in Data Science

Homework 2 - PART 1 - Graph Kernel

Groupe 40

21/10/2022

Pierre Remacle
Pierre Merveille
Salma Ismail

Teatchers :
Jean-Charles Delvenne
Rémi Delogne
Estelle Massart
Bastien Massion
Gautier Krings
Brieuc Pinon

2022-2023

Contents

1 Computing the kernels

1.1 Weisfeiler-Lehman Subtree Complexity

The first step of the Weisfeiler-Lehman subtree kernel is to create the multisets for each nodes. The multisets are the sets of the neighboring labels. The maximum amount of neighbours is E (the number of edge). Creating a set of neighbouring labels for a single node takes O(E) complexity. While it was not explicitly mentioned in the assignment, it remains quite important to know that the set must be ordered in order to prevent redundancies in the relabeling stage. This sorting can be made clear through 0(E log E) with the Quick sort. Based on the fact that O(E) >= O(E log E), we keep O(E) as the maximum complexity. Moreover, For each node, it takes complexity O(NE). Due to the existence of two graphs, we can add a times 2, but since it is a constant, it will be eliminated from the final complexity.

Making a two-component tuple is the second stage. The first one is the node v's previous label, and the second is the node's collection of discovered neighboring labels. This step can be done in 0(N). It's one operation by node (N)

The third stage may be carried out with a constant complexity. If this tuple has never appeared before, it is mapped into a new integer label in a single operation. If so, the algorithm maps the tuple onto the label discovered in a previous iteration. It has the same O(N) complexity as the preceding step.

The fourth step is to set the new labels for the nodes. Once again in O(N).

For a single iteration, the complexity can be represented by O(NE) + 3*O(N). Without loss of generality, we can assume that the symptotic computational complexity for one iteration can be represented by O(NE).

Since this algorithm iterates H times, its overall complexity is equal to O(HNE). The asymptotic computational complexity of the Weisfeiler-Lehman subtree kernel equals to O(HNE).

1.2 Explicit Features

The kernel takes as input two graphs and applies an inner product on these two graphs having undergone a non-linear transformation. This non-linear transformation is called the embedding.

$k_{WLsubtree}(G1, G2) = \langle \phi_{WLsubtree}(G1), \phi_{WLsubtree}(G2) \rangle$.

Let's define Σ as the set of caracters that occur as node labels at least once in G or G'at the end of WL algorithm. Σ_i is the set of caracters at the end of i-th iterations of WL algorithm, $\Sigma_i \subset \Sigma$. So Σ_0 is the inital labels set. Each elements in Σ_i are different. We can write $\Sigma_i = \{\alpha_{ij} | \forall j : 0 \leqslant j \leqslant |\Sigma_i|\}$.

$\phi_{WLsubtree}(G1) = (c(G1, \alpha_{01}), ..., c(G1, \alpha_{0|\Sigma_0|}), ..., c(G1, \alpha_{H1}), ..., c(G1, \alpha_{H|\Sigma_H|}))$
$\phi_{WLsubtree}(G2) = (c(G2, \alpha_{01}), ..., c(G2, \alpha_{0|\Sigma_0|}), ..., c(G2, \alpha_{H1}), ..., c(G2, \alpha_{H|\Sigma_H|}))$.

where $c(G, \alpha_{ij})$ corresponds to the number of occurrences of the label α_{ij} in the graph G.

1.3 Explicit Embedding versus Kernel

After computing the rank of the WL subtree kernel matrix we get the following values. The rank of the kernel gives us a lower bound on the number of dimensions needed to represent the entire dataset without losing any information. Consequently, we have a lower bound on the implicit embedding space.

data set	rank of the WL subtree	number of lines
MUTAG	175	188
ENZYMES	595	600
NCI1	4002	4110

2 Visualization

2.1 Kernel Centralization

The formula for the centralization is $K - 1^N K - K1^N + 1^N k 1^N$ with 1^N as a matrix of size (N, N) with all entries set to 1/N.

$$1^N = \begin{bmatrix} 1/N & \dots & 1/N \\ \vdots & \ddots & \vdots \\ 1/N & \dots & 1/N \end{bmatrix}$$

this formula can be reduced to:

$$\tilde{K} = K - 2 * 1^N K + 1^N k 1^N$$

and we have to prove that it is equivalent to the below:

$$\tilde{K}_{i,j} = \langle \phi(G_i) - \bar{\phi}, \phi(G_j) - \bar{\phi} \rangle$$

by the linearity of the inner product we can reduce the expression

$$\langle \phi(G_i) - \bar{\phi}, \phi(G_j) - \bar{\phi} \rangle = \langle \phi(G_i), \phi(G_j) \rangle - \langle \phi(G_i), \bar{\phi} \rangle - \langle \bar{\phi}, \phi(G_j) \rangle + \langle \bar{\phi}, \bar{\phi} \rangle$$

Secondly, we apply the inner product on each individual element

– 1 –

$$\langle \phi(G_i), \phi(G_j) \rangle = K$$

– 2 –

$$\langle \bar{\phi}, \phi(G_i) \rangle = \langle \phi(G_i), \bar{\phi} \rangle = \langle \phi(G_i), \frac{1}{n} \sum \phi(G_j) \rangle$$

$$\langle \phi(G_i), \frac{1}{n} \sum \phi(G_j) \rangle = \frac{1}{n} \sum \langle \phi(G_i), \phi(G_j) \rangle$$

$$\frac{1}{n} \sum \langle \phi(G_i), \phi(G_j) \rangle = 1^N * k$$

– 3 –

$$\langle \bar{\phi}, \bar{\phi} \rangle = \langle \frac{1}{n} \sum \phi(G_i), \frac{1}{n} \sum \phi(G_j) \rangle$$

$$\langle \frac{1}{n} \sum \phi(G_i), \frac{1}{n} \sum \phi(G_j) \rangle = \frac{1}{n} \sum \frac{1}{n} \sum \langle \phi(G_i), \phi(G_j) \rangle$$

$$\frac{1}{n} \sum \frac{1}{n} \sum \langle \phi(G_i), \phi(G_j) \rangle = 1^N * k * 1^N$$

– conclusion –

$$\langle \phi(G_i) - \bar{\phi}, \phi(G_j) - \bar{\phi} \rangle = K - 2 * K1^N + 1^N k 1^N$$

2.2 Pairwise Distances

We understand this to be true: The Euclidean distance between two points can be written as the norm of there difference

$d(u, v) = \|u - v\|$

We can then take the square of the norm of $\phi(G_1) - \phi(G_2)$

$$\|\phi(G_1) - \phi(G_2))\|^2 = \langle\phi(G_1), \phi(G_1)\rangle - \langle\phi(G_1), \phi(G_2)\rangle - \langle\phi(G_2), \phi(G_1)\rangle + \langle\phi(G_1), \phi(G_2)\rangle$$

knowing that $\langle\phi(G_1), \phi(G_2)\rangle = \langle\phi(G_2), \phi(G_1)\rangle$ and that $k(G_x, G_y) = \langle\phi(G_x), \phi(G_y)\rangle$ we get

$$d(\phi(G1), \phi(G2)) = \sqrt{k(G_1, G_1) + k(G_2, G_2) - 2k(G_1, G_2)}$$

2.3 Comparison between KPCA and tSNE

KPCA and tSNE are two non-linear dimensionality reduction techniques. PCA is the algorithm that reduces the dimension of a dataset that can be linearly seperatable. tSNE can also be used for linear data.

The KPCA algorithm works in the same way as the PCA. However, the algorithm reduces the dimension of the kernel and not the base dataset. The idea is to use the eigenvectors of the Kernel. The observations will be projected on the N vectors normalized being linked with the N most important eigenvalues. N is defined as the new dimension. These N vectors are named Kernel Principal components and represent most of the total variance in the data. This method would theoretically allow a better linear separation of the initial data using the depth of dimension and the Kernel trick.

On the other hand, the tSNE algorithm works differently. Indeed, this algorithm is based on the initial representation between the different observations such that data that is close in the original space has a high probability of having close representations in the new space and vice versa. The first step is to calculate the probabilities for the observations to have a similar representation. In our case, this probability is produced thanks to the distance matrix calculated on the basis of the Kernel. Then the algorithm reduces the dimension of the input space. The inputs are represented randomly in this space. And finally, through an iterative process, the algorithm groups observations with a high probability of having a similar representation in said novel space. And at a later stage discards the observations with a low probability. It is also important to note that the TSNE algorithm has several hyper parameters. Here we are working with a perplexity of 50 (representing the number of neighbours we are considering) and a number of iterations that is 1000.

2.3.1 MUTAG

The KPCA algorithm used on the Kernel to reduce the dimension of this same Kernel to a dimension of 2, allows us to visualise a separation between the two groups of observations. However, this separation is not clear. Indeed, if we apply a linear classification algorithm on this new representation, we could not obtain a lossless classification. Some observations will not be correctly represented. This visualisation is still quite informative.

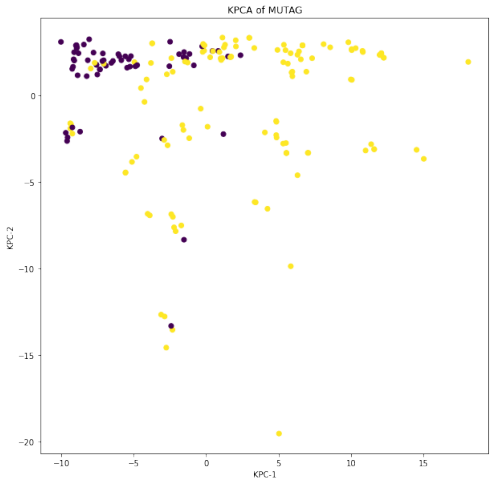

Figure 1: KPCA of MUTAG (dim: 2)

The TSNE algorithm on the same dataset returns a less clear visualisation. Indeed, we can still notice two main clusters but a linear classification on this visualisation will give us a worse result than with the KPCA. Our hypothesis is that the initial dimension is too high to be reduced directly to 2 dimensions. An alternative would be to reduce the initial dimension with the KPCA to obtain a smaller input space but still keep variance information. And then applying the TNSE algorithm on this new space. We can also see that the points are quite far apart, which is perhaps due to a wrong choice of hyperparameters. However, after applying several combinations of perplexity and iteration numbers we did not get clearer results.

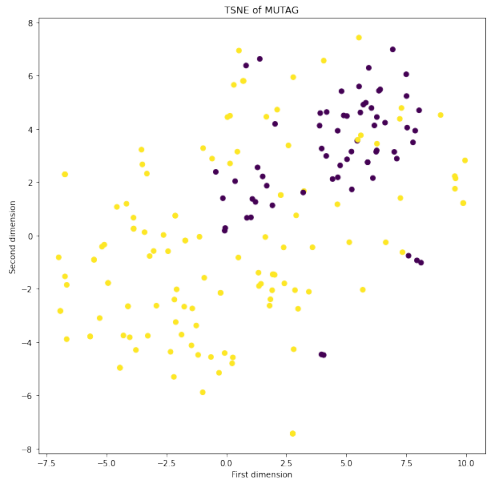

Figure 2: TSNE of MUTAG (dim: 2)

For the MUTAG dataset we can reduce the dimension of the Kernel to a dimension of 2 via KPCA in order to obtain a fairly accurate classification model. Our TSNE model no longer seems relevant.

2.3.2 ENZYMES

The KPCA dimension reduction of the ENZYMES dataset does not give us a useful 2-dimensional visualisation. Indeed, all the observations are condensed in a cluster. Consequently, a linear classification model on this representation will be poor (accuracy).

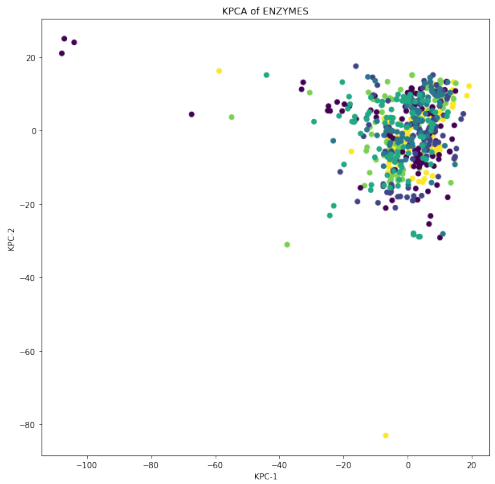

Figure 3: KPCA of ENZYMES (dim: 2)

These observations are probably due to the fact that the first two Kernel Principal components do not represent a large enough percentage of the variance of the feature space (mapping of the input space). The graph below allows us to visualise the percentage of variance represented by the different KPCs. We can see that the first 2 KPCs represent more or less 35% of the variance of the feature space. 2 dimensions do not seem to be enough to represent this dataset.

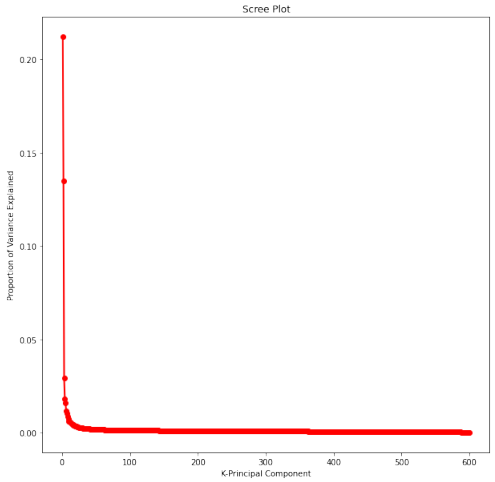

Figure 4: scree plot KPCA

The application of the TSNE algorithm gives us a visualisation as poor as the KPCA for 2 dimensions. The reasons are probably the same as those stated for the MUTAG dataset.

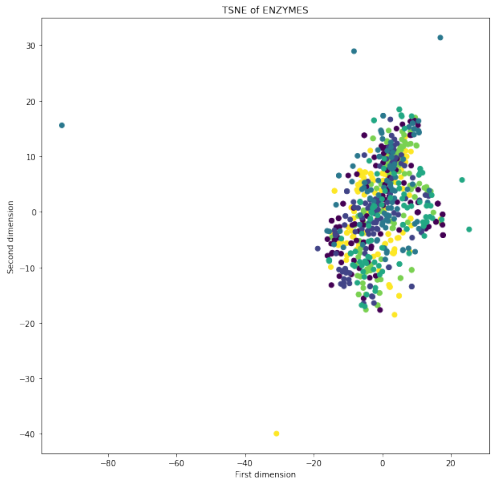

Figure 5: TSNE of ENZYMES (dim: 2)

Thanks to these observations, we can deduce that the reduction to 2 dimensions of the Kernel of the ENZYMES dataset will probably not allow the correct classification of new entries.

2.3.3 NCI1

The same observations as the ENZYMES dataset can be made on the NCI1 dataset. Indeed, the reduction to a dimension of 2 does not allow us to separate our two classes of points. The crown of points equidistant that we can see on the TSNE visualization probably comes from the fact that we used a maximum number of iterations equals to 1000 and there are much more observations to classify. This is due to a lack of computationnal power. We were able to identify two areas for improvement. The first, for the KPCA, is to use a higher dimension to capture more variance. The second, for the TNSE, would be to reduce the initial dimensions with the KPCA and then reapply the algorithm.

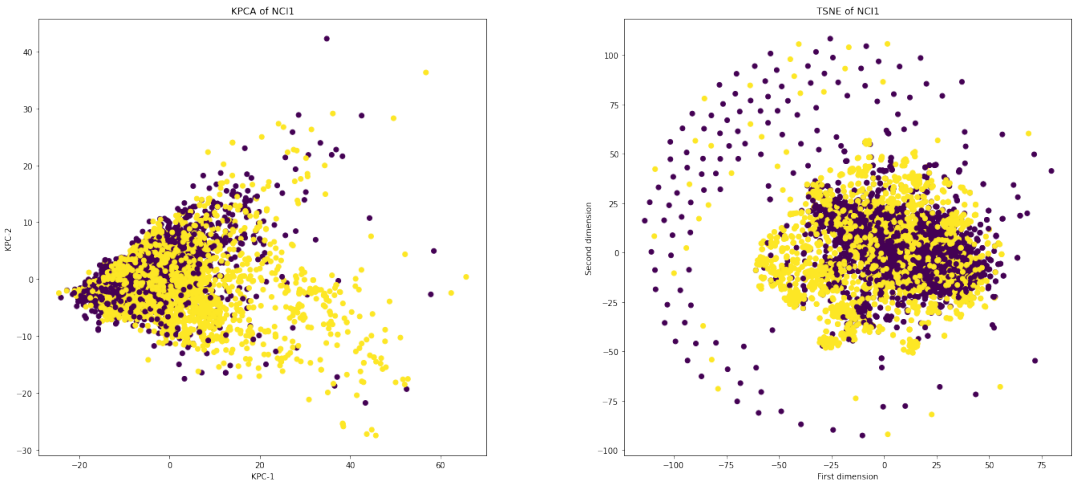

Figure 6: KPCA and TSNE of dataset NCI1 (dimension: 2)

3 Classification

3.1 A simple baseline

To compute the constant model of each datasets, we took the most represented constant in each of them.
we end up with these results. We find a very low accuracy for all models. A constant model can't be used in this particular situation.

data set	accuracy of constant model	number of possible values
MUTAG	0.665	2
ENZYMES	0.166	6
NCI1	0.5	2

Figure 7: constant model accuracy

3.2 Support Vector Machines (SVM)

We have trained a Support Vector classifier on the pre-computed WL Kernel.
We have found these results. They are a lot better than with a constant model. For the data set MUTAG and NCI1, the model found interesting result. Based on these result we can assume that we can classify the data linearly based on their representation in the feature space (Kernel).

data set	SVC accuracy
MUTAG	0.895
ENZYMES	0.508
NCI1	0.82

Figure 8: SVC accuracy with hyperparameters C = 100 and H = 3

3.3 Select Hyperparameters

To select the best combinations of hyperparameters for a SVC based on our datasets, we have to execute a 10-fold cross validation on our training set. This cross validation allows us to split our training set in 10 subsets and train our model on each combinations of 9 subsets. The last subset is used to validate our model. We took as the measure of performance, the mean of the 10-fold cross validation. Indeed, this cross validation returns 10 values. One for each combination. Thanks to that, we define the best combination of hyperparameters. It is important to notice that the split of our datasets is done randomly (seed: 42). It is however, possible to find another training set that find a better combination.

3.3.1 MUTAG

The best validation scores found for this dataset is 85%. And this validation score has been found by using C = 0.01 and H = 5. Based on these hyperparameters, the model accuracy is equal to 92,1% on the testing set. Using a SVC allows us to have a model that classifies new entries fairly correctly. However, depending on the case, 92% accuracy can be seen as not accurate enough.

3.3.2 ENZYMES

The best validation scores found for this dataset is 48%. And this validation score has been found by using C = 0.01 and H = 4. Based on these hyperparameters, the model accuracy is equal to 45% on the testing set. Using an SVC on this dataset does not seem to be recommended at all.

3.3.3 NCI1

Due to a lack of power computation, we only focused on the first 7 C. We have found 2 combinantions of hyperparameters that maximize the validation score (85,2%). We have chosen C = 10 and H = 9. Based on these hyperparameters, the model accuracy is equal to 86,1% on the testing set. The use of SVC for this dataset allows for a fairly high percentage performance for new entries. A linear classification in the feature space seems to be a good idea.

4 Final Observation

We can observe a relationship between the visualization done and the classifying accuracy of our models for the two first datasets. Indeed, our initial observations based on the visualization tend to be assessed by the SVC model. The MUTAG dataset, which we found interesting KPCA visualization, derived a linear classification on the feature space pretty accurate. On the other hand, the dataset ENZYMES with the poor visualization tend to be derived in a poor classification model. However, for the third dataset, we note that it is possible to train a linear classification model on the feature space with a fairly high success rate. This reinforces our hypotheses made when visualising it. Indeed, perhaps we would have seen distinct divisions if we had used more components for the KPCA or if we had used the TSNE on a space that was already reduced via this same KPCA.