# AML Project

December 2018

Jade Cock | jadec | 971005- 6665

Youness El Idrissi | yei |979818-T152

Merlin Sewina | sewina | 952610-T358

Amelie Grall | grall | 961007-T343

Group Cock-Grall-ElIdrissi-Sewina P

**Abstract**

Labelling instances is a tedious task and requires a consequent amount of resources. With the amount of data growing constantly, supervised methods are becoming difficult to apply with the available data being rarely labelled. This report presents state of the art semi-supervised learning algorithm which incorporates both unlabelled and labelled data as prior knowledge to the machine learning algorithm. Specifically, this paper focuses on repeating and reproducing the experiments in the paper "Cluster Kernels for Semi-Supervised Learning". Specifically, it is aimed to reproduce the results of the experiments presented in [5]. Though the results for thet text dataset were similar if not slightly better than the ones obtained in the original paper, the parameters had to be slightly tuned as the ones described in the paper did not give conclusive results. Similarly to the text dataset, the parameters described in [5] for the digit dataset had to be tuned too. Due to the number of ambiguities and the lack of clarity throughout the whole paper, not all results could be reproduced. The best performances achieved were due to guesses and trials and error, which could have been avoided by a more thorough description of their main algorithm. Eventually, tests on larger datasets and datasets with different distributions in the training sets have been conducted. It was noticed then that the Cluster Kernels did not scale well for such applications.

## 1 Introduction

In many fields, when labelled data is available, supervised learning algorithms are applied. However, labelling data is a tedious task that requires resources that are not always available.

Semi-supervised learning is an approach to solve this task with less effort. Supervised and unsupervised learning methods are combined to create an algorithm which can use the knowledge from both labelled and unlabelled data to build decision boundaries separating different classes.

The paper "Cluster Kernels for Semi-Supervised Learning" provides a skeleton where a linear classifier is given a new representation of the data. This new representation takes into account both labelled and unlabelled data by using the idea that the similarity of two data points within a cluster should be higher than the similarity across two different clusters and therefore should share the same label. This idea can be referred to as the *cluster assumption*. Implementing this assumption as prior knowledge into a classifier should then increase its performance. Examples of such clusters are the *Mutual Information Kernel*[10], the *Fisher Kernel* [8] or the *Marginalized Kernel* [12]. Those, however, all share a similar trait which is to depend on generative models [5]. The random walk kernel, however, does not. It builds a matrix where each entry $k_{i,j}$ is the value of any two data points $i, j$ passed into the RBF kernel. This results in something similar to a transition matrix where points in the same clusters should output a higher probability than points in different clusters.

Another different approach to implement the cluster assumption is to transform the input before using a linear classifier [13]. For example, spectral clustering which is similar to PCA.

Likewise, the paper Cluster Kernels for Semi-Supervised Learning [5] proposes a matrix implementing the cluster assumption through eigendecompositions and transfer functions proceeding in the following way:

1. Compute the RBF matrix with all the data (labelled and unlabelled) where all elements of the diagonal are 1's, $D$ which is a diagonal matrix where the diagonal elements are the sum of the rows of the RBF matrix. Compute then $L = D^{\frac{-1}{2}} K D^{\frac{-1}{2}}$ and its eigendecomposition of $L$ such that $L = U \Lambda U$

2. Use a transfer function $\phi$ to transform the eigenvalues such that $\widetilde{\Lambda}$ becomes the new set of eigen values. The different transfer functions are the following:

   (a) Linear Kernel $\longrightarrow \phi(\lambda) = \lambda$

   (b) Step Kernel

   $$\longrightarrow \phi(\lambda) = \begin{cases} 1, & \text{if } \lambda \geq \lambda_{threshold} \\ 0, & \text{otherwise} \end{cases}$$

   (c) Linear-Step

   $$\longrightarrow \phi(\lambda) = \begin{cases} \lambda, & \text{if } \lambda \geq \lambda_{threshold} \\ 0, & \text{otherwise} \end{cases}$$

   (d) Polynomial $\longrightarrow \phi(\lambda) = \lambda^t$

   (e) Polystep

   $$\longrightarrow \phi(\lambda) = \begin{cases} \sqrt{\lambda_i}, & \text{if } i \leq n + 10 \\ 0, & \text{otherwise } i > n + 10 \end{cases}$$

3. You now have $\widetilde{L} = U \widetilde{\Lambda} U^T$ with which you can build $\widetilde{D}$, the diagonal matrix where $\widetilde{D}_{ii} = 1/\widetilde{L}_{ii}$ and $\widetilde{K} = \widetilde{D}^{\frac{1}{2}} \widetilde{L} \widetilde{D}^{\frac{1}{2}}$

The final kernel matrix $\widetilde{K}$ is then used as input data in a linear classifier where the labelled points $x^n$ equivalent to the $n^{th}$ row in $\widetilde{K}$ are used as training instances. The unlabelled instances are then classified through the trained model, where each unlabelled instances $x^i$ in the original dataset corresponds to the i$^{ith}$ row in $\widetilde{K}$.

In the paper, the effects of the different transfer functions used as a kernel in their algorithm are compared among each other, and against different state of the art classifiers. The purpose of this report is to validate the experiments conducted by the authors of "Cluster Kernels for Semi-Supervised Learning". This implies repeating them, comparing our results to those described in the paper, as well as evaluating their research methodology. Additionally, we further extend their research through testing the scalability of the proposed algorithm on large datasets and its performance when there are imbalances in the training set.

## 2 methodology

In this section, we explore the methodology of the authors through three main arguments: the lack of clarity, the lack of details, and the lack of justification for certain methods.

### 2.1 Lack of Clarity

Throughout the paper [5], it is unclear whether they attempt to replace the kernel matrix in the SVM, or use it as new inputs, as in 2.2 it is said "how a modified version of point can be used as a kernel"[5] and in 2.3. "we suggest to train a linear classifier on the mapped points"[5]. We decided the latter version would be used. As there is not enough details to implement the former.

Another problem was which dataset they used. Indeed, the 20 newsgroup dataset has two targets which containing the word "windows". None of them matched the dimensions they described. Moreover, the link to the dataset being dead, it was impossible to reproduce the same text processing.

Additionally, it is unclear how they obtain their results. They used 100 random selection, but was it per fixed number of labelled instances or 100 overall? The lack of standard deviation and correct labels on the axes add up to the confusion.

### 2.2 Lack of details

Lack of details arises in the experiments. It is unclear which linear classifier should be used to implement cluster kernel, and with which parameters.

Additionally, the span estimate is usually applied to choose SVMs' parameters automatically in a one leave out situation. How to apply this to our polystep function with a keep one in approach is not documented enough to reproduce it. Indeed, it relies heavily on the support vectors set being very similar from fold to fold, which is not the case here.

Furthermore for the Text-Classification dataset in 4.1 they mention that "... 2 to 128 (points) were randomly selected to be labelled..." which implies that it is not important that at least 1 sample for each class has to be given, but in fact for proper training at least one instance per class has to be present in the training set. This was never mentioned in the paper, which reduces the reproducible aspect of their experiment.

In the experiment regarding the digit dataset again, it was unclear whether they used the same test sets for all algorithms or not.

On a slightly different register, the graphs they display in this paper are never properly annotated. The axis are not labelled, there is no title to the graph, which is especially useful when two of them are set in the same figure. Additionally, on figure 2 of [5] the scaled chosen for the x-axis does not depict fairly how the performance improves as more instances are given to the training set.

## 2.3 Lack of motivation

What adds to the general confusion is that the lack of details implies the lack of motivation/justification.

The linear step transfer function is described in the main skeleton, but polystep is used in all experiments. Why they made this decision is never motivated.

Another lack of motivation is to be found in the use of SVM as a classifier in (supposedly) both experiments. They mention that their $\widetilde{K}$ matrix can be used on any linear classifier, but they only train it on SVMs. Are those proven to work better to train a few instances? Or is it just convention?

Moreover, the way they split the training and testing folds and the classes for the digit dataset seems arbitrary, just like the choice of replacing linear step for polystep.

## 2.4 Conclusion

There is a lot of ambiguity in this paper that might prevent people who are not mastering semi-supervised learning at a similar level as the authors from repeating and reproducing the experiments or applying those methods to their problems.

The methods presented are fascinating and could help in many situations, but unfortunately, not enough background information is given to use cluster kernel to its full potential.

## 3 Experiments

## 3.1 Text Dataset

In this set of experiments, a subset of the 20newsgroups dataset was used. The one as close as possible as in [5]. We tried to remain as loyal to the original paper given the information available (2.1). The following section describes the types of experiments executed and displays the resulting performances.

### 3.1.1 Different algorithms

**Standard SVM** Using the scikit SVM implementation, we trained the classifier with 16 labelled points, 8 from each class. We obtain a test-error of 27.04% with a standard deviation of 7.84%. This result is similar to the one in the paper where they have an error of 27.5% with a standard deviation of 7%.

**Transductive SVM** For the transductive SVM we used an existing implementation by Fabian Giesecke [1] instead of implementing the entire algorithm ourselves, due to its complexity. A total of 16 labelled data points, 8 for each class, was used in the training process. The test error was computed through 100 fold cross-validation. The mean obtained was 24.69% with a standard deviation of 0% compared to the results of the paper with 15.6% with a standard deviation of 2.5%. The absence of variance was unexpected. We assumed that this might be due to the implementation [1]. Unfortunately, the code had to be modified before we were able to get it working. Therefore the implementation (for sparse vector data) might not be entirely accurate.

**Cluster kernel** The lowest error obtained using the cluster kernel for 16 labelled data points (averaged over 100 trials) was 7.1%, 4.9% lower than the result on [2].
Regarding the use of the transfer functions, if we take a look at "Figure 2'" the best overall performance was obtained using the step and poly-step function. However, the poly-step function used in [2] was merely raising all the eigenvalues to the power of 2. This resulted in all eigenvalues being extremely small, too small to be trained on an SVM. The classifier was consistently unable to find suitable support vectors. Moreover, it seemed that this way,

the poly-step function was not used to the maximum of its potential. Besides, in practice, it was not producing good results. We decided to use this one instead:

$$\phi(\lambda_i) = \begin{cases} \sqrt{\lambda_i} & i \leq \lambda_{44} \\ \lambda^2 & i > \lambda_{44} \end{cases}$$

where $\lambda_{44}$ equals to approximately 0.003 in most of the cases. Finding these results took a considerable amount of time and effort because the parameters stated in [5] were not outputting the reported results.

**Random walks** Another machine learning technique which can be compared with cluster kernels is the partially labelled classification with Markov random walks, which was described in [5]. This algorithm has been implemented with expectation maximisation estimation. It was tested on non linear data (figure 1). There were 200 points separated into 2 classes, and 5 points of each class were labelled. The parameters chosen are t = 8 and $\sigma$ = 0.6. 7% test-error has been reached with these parameters.

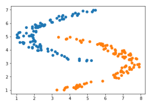

Figure 1: Data to test random walk algorithm.

This algorithm has been tested on the text dataset with these parameters and by changing $\sigma$ and t to search for the optimal solution. However, accuracy did not exceed 48%. It must be because of the curse of dimensionality. Indeed, this has been tested with a proportion of labelled data between 5% and 100%. Increasing the number of labelled data did not show any improvement in the results. Thus, the problem does not spring from the fact that the data is partially labelled. It has been checked that the labelled data came from both classes, and the proportion of both classes was balanced. Thus, the algorithm was trained to recognise well and differentiate the classes. This dataset has the same normalisation than the data from figure 1: labels are the same and dimensions are processed similarly. The text dataset is much more complex. There are 1919 samples, and the dimension of each sample is about 350. In the matrix of the t-step transition probabilities, all the columns are the same, resulting in the same posterior probability to belong to one class or the other one for each point. It means that the problem is not when the EM estimation is done but when the distance between the text data is executed. It must be because there are so many dimensions that distances are meaningless.

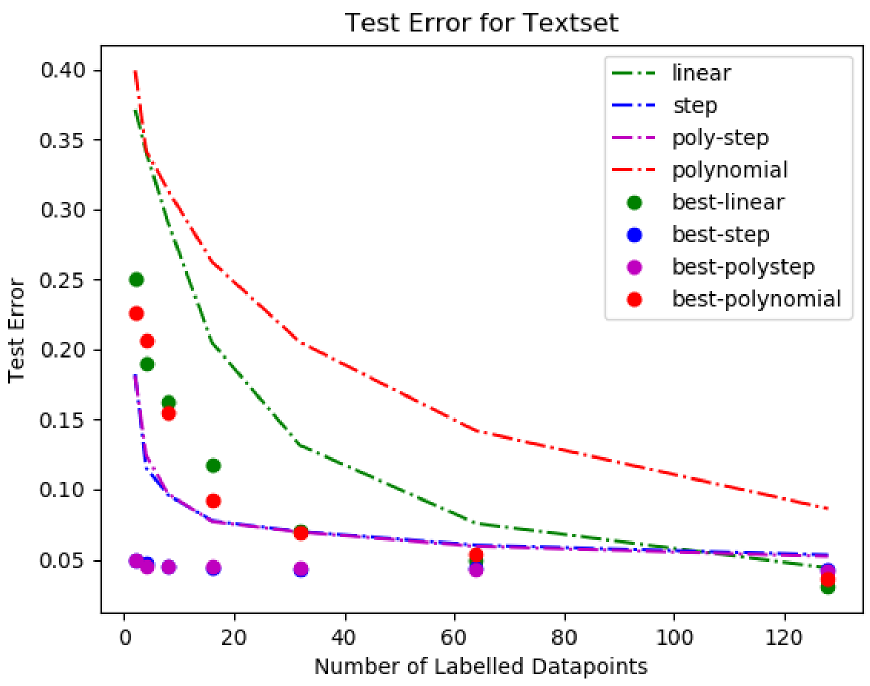

Figure 2: Performances with different transfer-functions with different amount of labelled data-points

$DIGIT = zero|one|two|three|four|five|six|seven|eight|nine;$
$CMD$ =DIGIT $DIGIT$DIGIT $DIGIT$;
( SENT-START $CMDSENT - END$)

### 3.2 Digit Dataset

All experiments below have been conducted on the digit dataset, split, trained and tested in the same fashion as in [5] on our SVM implementation. That is, with digits from 0-4 and digits from 5-9 being the two classes, and cross-validation using fifty subsets mutually exclusive of 40 instances. All algorithms are tested on the test mean error. Notice that all instances have been scaled as it gave dummy predictions (all -1) when not. For the different version of the algorithm, exploratory research has been conducted to find the best suitable parameters (manual hyperparameter optimisation). That is, we tried first a big range of parameters with a big step size, and decreased the range and step size where the performance varied. We noticed there was always a ”stable zone” where no matter what parameter values we implemented in that range, the performance value would always be the same, and a small variation zone. The mean optimal value across 50 folds for all tests are given.

#### 3.2.1 Different algorithms

**Standard SVM** The mean obtained with our implementation was 27.12% with a standard deviation of 5.6 %. Scikit SVM performed 33.88% with a standard deviation of 5.67.

**Transductive SVM** As earlier mentioned for the transductive SVM we used an existing implementation by Fabian Giesecke [1]. The mean obtained was 32.55% with a standard deviation of 1.73%

**Cluster Kernel with Different Cutoffs**

**With SVM** Trained on an SVM with a polynomial kernel of degree 2, with different values for the lambda threshold in the polystep transfer function. The minimum value was found when the lambda threshold was 0 and had a mean of 27.40% and a standard deviation of 4.28%

**With logistic regression** Trained with logistic regression the minimum test error was 35.34%, and a standard deviation of 5.12% which is why further exploration has not been conducted.

**K Highest Eigen Values** Similarly to the algorithm described in section 2.3 of [5], we tested Cluster Kernel with k highest eigenvalues. With different values from -20 to 20 and a step size of two, the performance was constantly 49.75%. Therefore, no further exploration has been conducted.

**Cluster Kernel with SVM and different degrees for polynomial kernel** In an exploratory research for the best degree for the polynomial kernel, the best minimum value was 25.69% for degree 1, which equivalent to the linear kernel.

**Cluster Kernel with SVM and different gammas for RBF kernel** Exploration of the best gamma value for the RBF kernel. The minimum average performance for all gammas is 46.63% when gamma equals 0.06.

**Cluster Kernel as Kernel representation within the SVM** Using the cluster kernel matrix within the SVM is a task that could not be achieved, as too many and not enough possibilities to do so existed, and no information in the paper helped us.

**Cluster Kernel with submatrices** Alternative implementation: only use the columns of the instances it has been trained on. Best performance with optimal SVM parameters and exploratory research on the lambda cut was 27.45%, which is less than the best performance by using all columns.

## 3.3 Conclusion

The performances achieved by our implementation of their algorithm for the text dataset were aligned with those in [5]. However, the effort required to reach those results were uselessely complicated. The information given in the paper was never enough for us to understand what exactly needed to be done. This opening for interpretation makes it very complicate to use their algorithm at its full potential.

Then, the discrepancy between the results from theit paper and ours for the digit dataset vary from 10 to 30 per cent for the digit dataset. Again, the reasons for this is that there is a lack of background information and that our data might have slightly differed in the way it is processed.

For both datasets, the optimal parameters described in [5] did not work at all with our implementation, and we had to tune them manually in order to make the algorithm work. Whether it is because not enough information was given as to how to use those parameters, or because they needed those parameters to give some form of motivation is not clear.2.

## 3.4 Extending the Research

**Scalability - Pulsar Dataset** Cluster Kernel was tested on the pulsar or HTRU2 dataset from the UCI repository.

The first thing to notice is that memory errors were encountered, which shows that computing the kernel matrix is not scalable for datasets of a more consequent size.

A subset of 1600 instances from that dataset has been trained and tested on 100 folds with a test error of 23% for the polynomial kernel of degree 5. Standard SVM achieves 13%, therefore performing better. The conclusion here is that Cluster kernel is not scalable.

**Imbalances - Digit dataset** As it can be seen on the graph below, Cluster kernel does not perform well when the training set is imbalanced. No matter where the imbalance lies, in other words, which class is the minority one, the algorithm performs badly and is not suited for such situations. On the graph below, it can be seen that the best value is 23.7% when there is 47% of positive instances.

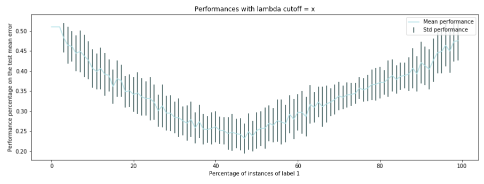

Figure 3: Performances with different distributions of instances of label 1 in the training of the SVM

## 4 Conclusion

In this report, we attempted to repeat the experiments of the paper "Cluster Kernels for Semi-Supervised Learning" and reproduce their results. The performances conducted on the text dataset aligned with those found in the paper. However, the experiments conducted on the digit dataset have a discrepancy of 10 to 30 %. Additionally to this, the parameters suggested in [5] for the step function in the text data section, and for the polystep in both experiments had to be tuned, as those reported in [5] gave inconclusive results.

Then, the scalability and the ability to deal with imbalances in training has been tested. For both cases, it has been shown that Cluster Kernel is not suited to deal with those cases.

Overall, the lack of information, motivation and high level of ambiguities in this paper made it hard to reproduce the experiments, especially for the digit dataset. Though their algorithm is backed up by the cluster assumption, most of the choices they make both in their experimental setup and their method choices, seem arbitrarily chosen.

In short, it is an exciting field which will probably grow with the amount of data generated on a daily basis and the lack of resources to label it all. Further work would involve being able to deal with larger datasets and imbalanced labelled data.

## References

[1] Quasi-newton semi-supervised support vector machines. http://www.fabiangieseke.de/index.php/code/qns3vm. Accessed: 2019-01-14.

[2] D. Adams. *The Hitchhiker's Guide to the Galaxy.* San Val, 1995.

[3] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29, June 2004.

[4] Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. Choosing multiple parameters for support vector machines. *Machine learning*, 46(1-3):131–159, 2002.

[5] Olivier Chapelle, Jason Weston, and Bernhard Schölkopf. Cluster kernels for semi-supervised learning. In *Advances in neural information processing systems*, pages 601–608, 2003.

[6] Ming Gao, Xia Hong, Sheng Chen, and Chris J. Harris. A combined smote and pso based rbf classifier for two-class imbalanced problems. *Neurocomput.*, 74(17):3456–3466, October 2011.

[7] Guo Hongyu and Viktor L. Herna. Learning from imbalanced data sets with boosting and data generation: The databoost-im approach. *SIGKDD Explor. Newsl.*, 6(1):30–39, June 2004.

[8] Tommi Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *Advances in neural information processing systems*, pages 487–493, 1999.

[9] Mahesh V. Joshi, Vipin Kumar, and Ramesh C. Agarwal. Evaluating boosting algorithms to classify rare classes: comparison and improvements. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 257–264, 2001.

[10] Matthias Seeger. Covariance kernels from bayesian generative models. In *Advances in neural information processing systems*, pages 905–912, 2002.

[11] Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *Advances in neural information processing systems*, pages 945–952, 2002.

[12] Koji Tsuda, Taishin Kin, and Kiyoshi Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 18(suppl_1):S268–S275, 2002.

[13] Yair Weiss. Segmentation using eigenvectors: a unifying view. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 975–982. IEEE, 1999.