Advanced Machine Learning Project
Cluster Kernels for Semi-Supervised Learning
Project Group 82

Adrián Campoy Rodríguez - adriancr@kth.se - (940329 - 3336)
Doumitrou-Daniil Nimara - nimara@kth.se - (960908 - T154)
Nikolaos Chrysanthidis - nchr@kth.se - (940218 - 2431)
Theodoros Panagiotakopoulos - thepan@kth.se - (941017 - 2879)

January 15^{th}, 2020

Abstract

Semi-Supervised Learning is a sub-field of Machine Learning in which the training set consists of both labeled and unlabeled examples. In such a setting, one seeks to utilize the extra information provided by the unlabeled data to bolster his training on the labeled data set. Semi-Supervised algorithms usually assume that the underlying mapping is continuous or that the data form clusters. As such, distance plays a pivotal role in such algorithms. In this paper, we will examine the algorithm proposed in "Cluster Kernels for Semi-Supervised Learning" (Olivier Chapelle et. al. [7]), which utilizes and reshapes the traditional Radius Basis Function Kernel (RBF) in order to formulate an adequate metric to measure such distances. This new induced distance extracts information from the unlabeled data points and can be used to improve the accuracy of the traditional SVM. We will reproduce the results of the original paper, analyze the kernel's properties, propose improvements and apply the implementations to new data sets.

Keywords: *Semi-Supervised Learning, Clustering, Kernels, Support Vector Machine, Transductive learning.*

1 Introduction

In the paper "Cluster Kernels for Semi-Supervised Learning" [7] Olivier Chapelle *et. al.* introduce a method that increases the accuracy of an SVM classifier by exploiting unlabeled points. Their approach is identified as semi-supervised, since it leverages the unlabeled data in order to refine an affinity matrix $\tilde{K}$, which is then used for supervised classification.

Semi-Supervised Learning is a vast and ever growing field where one can opt into different approaches of utilizing both labeled and unlabeled data. For instance, one can use their labeled data in order to better tune some initial parameters of unsupervised techniques (e.g. utilize the mean of the labeled data points to initialize the centroids of an EM classifier) [13]. Semi-supervised learning can be very useful when the unlabeled data are far more than the labeled. This is very common in situations where obtaining data points is fairly cheap, while obtaining the labels can cost a lot of time or money [8]. This is usually the case in medical fields, where diagnosing (labeling) a patient (sample) often requires an expensive procedure (e.g. MRI, surgery).

An equally interesting perspective is that of Active (Machine) Learning, where the user trains a classifier on a small labeled data set and utilizes it to label the unlabeled data. The user intervenes (by manually labeling it instead of the algorithm) only in instances where the Algorithm showcases low confidence. This technique is useful in tasks where labeling is time consuming, such as object recognition in pictures [12].

As stated in the publication [7], in order to effectively use the unlabeled data one needs to formulate assumptions. The assumption that is being used by the authors is the so called "cluster assumption", which states that two points are likely to have the same class label if there is a path connecting them passing through regions of high density [7] (i.e. they are connected via regions containing many data points).

This article aims at showing how to design kernels which implement the aforementioned assumption (i.e. kernels that produce large distance for points in different clusters and small distances for points within the same cluster). For that purpose, Chapelle et al. present a set of kernels implementing the cluster assumption:

Kernels from mixture models: If we assume that the data points were generated by a **Generative Model**, then we can utilize the data and traditional unsupervised algorithms (e.g. Expectation Maximization, Variational Inference) to learn the underlying model and use it to induce a metric which places low distances to points with a high probability of belonging to the same mixture. For the Guassian Mixture models $\{N(\mu_k, \Sigma_k)\}_{k=1}^{q}$, a marginalized kernel based on Fisher's kernel is proposed in [6], where

$$K(x, y) = \sum_{k=1}^{q} P(k|x) \cdot P(k|y) \cdot x^T \Sigma y$$

Combining such a kernel with traditional SVM exhibits the intuitive behavior seen in figure 1, where the decision boundary **only passes through a cluster (mixture) if necessary**.

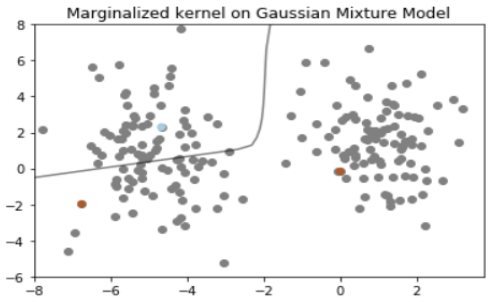


Figure 1: Marginalized Kernel SVM. A Gaussian Mixture Model was considered with means $\mu_1 = (1, 1)$, $\mu_2 = (-5, 1)$, and equal diagonal covariances Σ with diagonals $\sigma_1 = 1$, $\sigma_2 = 5$. The decision function has the expected behavior: it only passes through a cluster if necessary.

Random walk kernel: The method utilizes labeled and unlabeled data to create clusters. As proposed in [4], first, a symmetrized K nearest neighbour graph G is constructed and weights $W_{ij} = \exp\left(-d\left(\mathbf{x}_i, \mathbf{x}_k\right)/\sigma\right)$ are assigned to each edge connecting the data points. Then, the probability of transitioning from point i to point k is obtained from the weights as $p_{ik} = W_{ik}/\sum_j W_{ij}$ (note that $p_{ik} = 0$ for any non-neighbour k). In order to classify unlabeled points, Random Walk assumes that each data point has a label i.e. a distribution $P(y \mid i)$ (probability that a certain points i has the label y). These distributions are unknown, and constitute the main parameters to be estimated. Given a point k (labeled or unlabeled), the posterior probability of its label is given by: $P_{\text{post}}\left(y|k\right) = \sum_i P(y|i)P_{0|t}(i|k)$. To classify each point, we choose the class that maximizes the posterior: $c_k = \text{argmax}_c\, P_{\text{post}}\left(y = c|k\right)$, where $P_{0|t}(i|k)$ is the probability of transitioning from point i to point k as mentioned before, whereas the estimation of the unknown parameters $P(y \mid i)$ is done applying EM.

The project is organized as follows: In the following section (Methods), we will briefly describe the main methods used in this paper alongside with other implemented algorithms used for comparison. Afterwards, the Results section will include the reproduction of the experiments conducted in the paper on the *20news* and *USPS dataset*. This will be followed by a Analysis and Extensions section, where we will further analyze the method, propose improvements and apply the algorithms to other real data sets to observe their performance. Finally, we will conclude by summarizing the key notions of this method and highlighting its most pivotal aspects.

2 Methods

2.1 Support Vector Machine (SVM) and Transductive SVM (TSVM)

As previously mentioned, semi-supervised learning, which leverages the information provided by unlabeled data points, lies between supervised and unsupervised learning [9]. The aim of SVMs as an inductive discriminative classifier is to induce a general decision function for a learning task while the purpose of TSVMs as an transductive classifier is to predict the targets of the unlabeled data [1]. TSVMs inherit most of the SVMs properties [1] yet, they additionally perform minimization of the misclassification when it comes to the unlabeled data which are taken into account. Training a TSVM is time consuming since the iterative process of predicting the labels of a particular data set is rather computationally expensive. The implementation of TSVM is based on the algorithm presented at figure 4 of [1]. Starting by training a regular SVM, the algorithm then improves the error rate by switching the labels of the unlabeled data in order to maximize the margin on both labeled and unlabeled data points [1].

2.2 Cluster Kernel

Another approach for semi-supervised classification is to consider all the data points (both labeled and unlabeled) and use tools of spectral clustering (see [2]) to create a kernel, which can then be used in a supervised classifier such as Support Vector Machines. The kernel representation of all data has to be constructed in such a way that points in the same cluster are grouped together.

Based on this concept, the analyzed paper proposes an extension, where by performing eigendecomposition and configuring the eigenvalues, it is possible to create a robust kernel that incorporates the cluster assumption. First an RBF kernel is being used to create an affinity matrix K. Then, another matrix D is defined, which is a diagonal matrix consisting of elements that are the sum of the rows of K. Another matrix L is computed, such as $L = D^{-1/2}KD^{-1/2}$ and its eigendecomposition $L = U\Lambda U$. At this point it is possible to compute new kernels $\tilde{K}$ by applying different transfer functions φ to the eigenvalues $\tilde{\lambda}_i = \varphi(\lambda_i)$ of L and construct $\tilde{L} = U\tilde{\Lambda}U^\top$. Then $\tilde{K}$ can be calculated using $\tilde{K} = \tilde{D}^{1/2}\tilde{L}\tilde{D}^{1/2}$, where $\tilde{D}$, a diagonal matrix with $\tilde{D}_{ii} = 1/\tilde{L}_{ii}$.

The transformations proposed in the paper for the eigenvalues are the following: *Linear*, $\varphi(\lambda) = \lambda$, no transformation is being used; *Step*, $\varphi(\lambda) = 1$ if $\lambda \geq \lambda_{cut}$ and 0 otherwise; *Linear-step*, $\varphi(\lambda) = \lambda$ if $\lambda \geq \lambda_{cut}$ and 0 otherwise; *Polynomial*, $\varphi(\lambda) = \lambda^t$; *Poly-step*, $\varphi(\lambda) = \lambda^p$ if $\lambda \geq \lambda_{cut}$ and $\varphi(\lambda) = \lambda^q$ otherwise.

2.3 Projection

A key problem of the above approach is that it is necessary to include the test data in the definition of the kernel. This implies that when new points need to be classified, a new kernel has to be recreated and the model has to be retrained, something very inconvenient when working on large data sets. A solution to this problem, which is described in the paper and implemented during the assignment, is to calculate the projection of the new points to the space induced by the labeled data. Utilizing this procedure, we construct a new kernel matrix $\tilde{K}$ with dimensions $N \times L$, where N and L the length of the test and training set respectively. Note that this matrix is different from the initial Kernel matrix and constitutes only the projections between the new data and the rest of the data. The new matrix is essentially the projection of each test input to the training sample and can be calculated using: $\tilde{K}(\mathbf{x}, \mathbf{x}_i) \equiv \left(\tilde{K}\boldsymbol{\alpha}^0\right)_i = \left(\tilde{K}K^{-1}\mathbf{v}\right)_i$ where, $v_i = K(\mathbf{x}, \mathbf{x}_i)$[1], $K(\mathbf{x}, \mathbf{x}') = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}'))$ the initial kernel (RBF in the paper) and $\mathbf{x}$ the new test point.

3 Results

We tested our implementation of regular SVM, TSVM, Random Walk and Cluster Kernel on the USPS [16] and 20news [15] datasets (same as in [7]). In the USPS dataset, which consists of 16×16 pixelated black and white images, we managed to reproduce their results (taking into consideration the standard deviation):

	SVM	TSVM	Cluster Kernel	Random Walk
Our Errors	19.0%(±3.2)	20.1%(±3.2)	17.3%(±2.2)	11.4%(±2.5)
Paper Errors	17.8%(±3.5)	17.6%(±3.6)	14.9%(±3.3)	Not estimated

Table 1: USPS results reproduction. We used kernel width of $\sigma = 5$, $p = \frac{1}{2}$, $q = 2$ (as proposed in the paper). The results were averaged over $2000 = 40 \cdot 50$ training samples (40 unlabeled and 1960 unlabeled).

During our testing, an interesting property of the SVM algorithms was observed. Unsurprisingly, SVM is highly dependent on the proportionality of labels in the training set, as it tends to underestimate the boundary of the minority class. Initially, we randomly sampled 40 examples as labeled training data, ignoring the proportions of 1 $(5-9)$ and -1 $(0-4)$ labels. When doing this, all of our algorithms' accuracies dropped by $\sim 10\%$. This class proportionality effect will be further examined in the next section. Interestingly, Random Walk showcases exceptional accuracy [4] in both datasets, while the Cluster Kernel had the second best results.

The 20 news dataset consists of texts that span a wide variety of topics. We focused on the comp-sys-mac-hardware and comp-windows-x categories (mac and windows), as suggested by the paper. For preprocessing, we

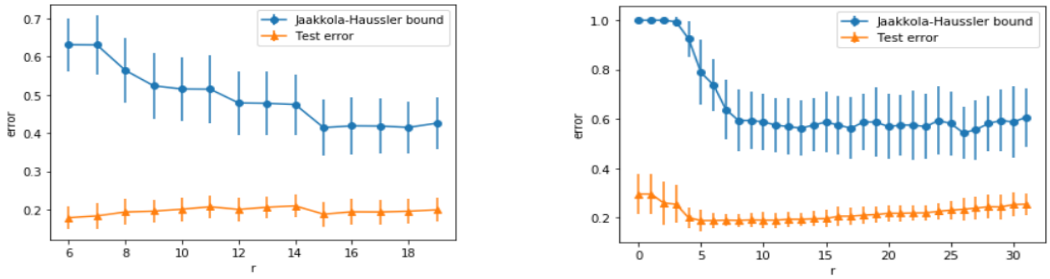
removed duplicate documents, computed the tf-idf score of the vocabulary of the corpus and removed the rarest words (by keeping the 7511 words with the highest tf-idf score, as suggested in [4]). Unfortunately, due to the sparsity of the problem, our TSVM implementation did not manage to converge to the desired precision. We argue that this is due to the fact that TSVM (without an optimal heuristic) can get trapped at the multiple local optima present in the high dimensional space induced by the sparse data. We base this argument on the high Variance of the method, which showcases that its accuracy was heavily influenced by the initial training set (which directly influences the initial labelling of the unlabeled data).

	SVM	TSVM	Cluster Kernel	Random Walk
Our Errors	31.8%(±4.4)	36.5%(±9.1)	18.8%(±3.2)	14.8(±1.6)%
Paper Errors	27.5%(±7)	15.6%(±2.5)	12.6%(±5.3)	15.5%

Table 2: 20news results reproduction. We used $\sigma = 0.55$, as proposed in the paper, while the training set consisted of 16 labeled examples as proposed in the paper.

Recall that the polynomial step transfer function ϕ contains three crucial hyperparameters, p, q, r. One way of finding the optimal value of r after fixing p and q is by estimating an upper bound of the **Generalized Test Error**. An unbiased estimate of this error is the *Jaakkola-Haussler Bound* [5], which is given by $T = \frac{1}{N}\sum_p I(a_p \cdot K(x_p, x_p) - 1 > 0)$ where a_p are the dual coefficients of the SVM.

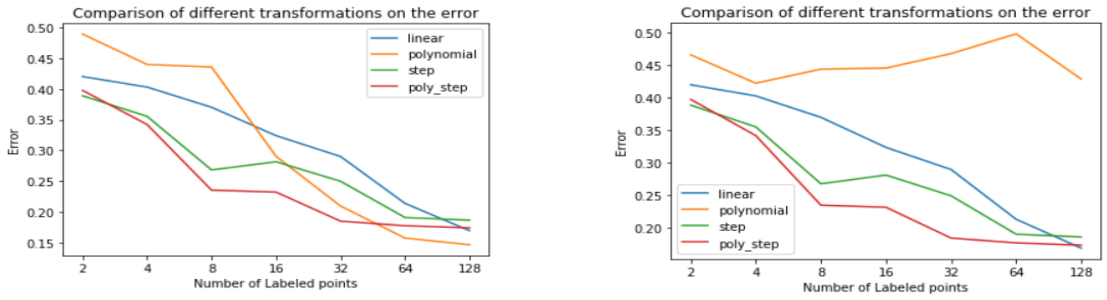
Note that this upper bound is not as tight as the one proposed in the paper (Span bound) and as such, we expect it to give higher error estimates. It nevertheless manages to capture the position of the minimum error in the USPS dataset ($r = 15$):



(a) USPS dataset **(b)** 20news dataset

Figure 2: Jaakola-Hayssler Bound Estimate. As expected, the estimation is indeed an upper bound minimized (though not as sharply) at the desired local minima

Lastly, one can also compare the effect that the transformation function ϕ has on the accuracy of the classifier. We examined a linear transformation (SVM), a polynomial of degree 3 and 5, a step and polynomial step function with a cut off index $n + 10$, n : number of labeled data samples. Increasing the labeled training examples augments the accuracy of our classifier in the following way:



(a) Polynomial of degree 3 **(b)** Polynomial of degree 5

Figure 3: Test error on 20news dataset for training size varying from 2 to 128 labeled examples. The different kernels correspond to different kinds of transfer functions

We did not manage to emulate their results for a polynomial transfer function of degree 5. The main reason for this is that the eigenvalues λ_i are fairly small, and as such contain little information when raised to a large exponent. This is why we see that the accuracy fluctuates and remains relatively the same, as the added λ_i^5 provide little new insight. On the contrary, the step functions manage to utilize the new data points, as they increase their effect by augmenting the most relevant (largest) eigenvalues ($1 \geq \sqrt{\lambda_i} \geq \lambda_i$).

Overall, our reproduction illustrates that Cluster Kernel constitutes a valid alternative for the SVM in a semi supervised manor.

4 Analysis and Extensions

4.1 Projection versus Utilizing the whole Test Set

Arguably, the most glaring issue of this method is how it handles test points. Because the kernel $\tilde{K}$ is not analytic (lack of direct formula), in order to utilize this kernel for predicting the test data, one must either use the test data during the fit stage or approximate the distance of the new test point x with the train samples, by projecting it to the space induced by the training data. Both approaches suffer major drawbacks. For instance, it is not always possible to incorporate the test data in your kernel, as it might not be available during the training phase (online learning). Furthermore, it scales poorly, as the increased size of the kernel will have a huge toll in speed and memory complexity ($O(N^2)$ space). Also, our experiments have shown that increasing the size of the kernel leads to numerical instabilities, as the condition number (an index of invertability [10]) of the RBF kernel increases significantly.

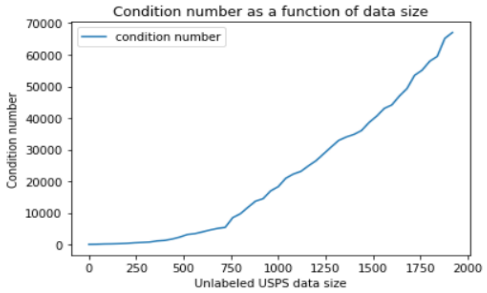
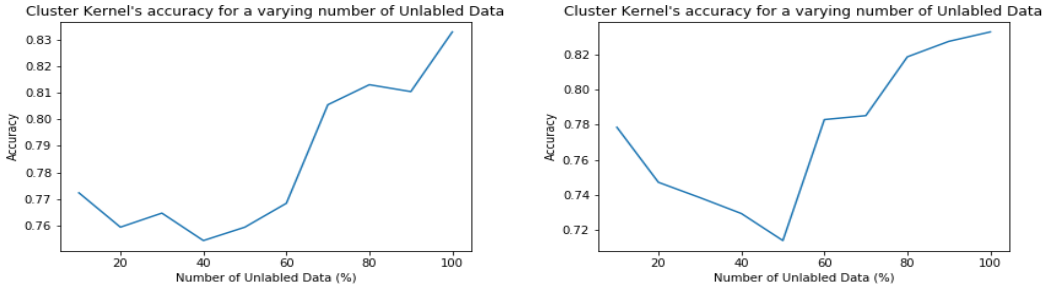


Figure 4: $k(K) = \frac{|\lambda_{max}|}{|\lambda_{min}|}$ condition number as we incorporate more and more unlabeled data in our matrix.

Even though projecting is still favored for online training, as its prediction time is substantially lower than that showcased when fitting the whole kernel (figure 7b and 7d), figure 4 illustrates why we might be unable to utilize it. When presented with a large amount of data, it might be numerically unstable to invert K^{-1}, in order to compute the projection $\tilde{K}(x, x_i) = (\tilde{K}K^{-1}v)_i$.

4.2 Cluster Kernel and imbalanced Data Sets

Traditional SVM is sensitive to imbalanced Data Sets [11], because it tends to underestimate the boundary of the minority class. A traditional approach in tackling this problem is to assign higher weights to the misrepresented class in order to balance this discrepancy. However, such an approach can not be utilized in a semi supervised scenario, where the imbalance can lie in the unlabeled data. In fact, our experiments show that aimlessly considering unlabeled data might lead to **reduced accuracy**:



(a) Increasing unlabeled data evenly

(b) Increasing unlabeled data unevenly

Figure 5: Cluster Kernel's performance for a different proportion of unlabeled data in the USPS dataset. In figure 5a, unlabeled data are added from both classes evenly and in 5b, classes are added sequentially (imbalanced). In 5b , all the unlabeled data from one single class is added (up to 50% of "Number of Unlabeled Data(%)") and then unlabeled data from the second class is added afterwards until finally reaching 100% of "Number of Unlabeled Data(%)". 40 samples were used for the training and 20 to 1960 data points were used as unlabeled. The transformation function selected is polynomial step.

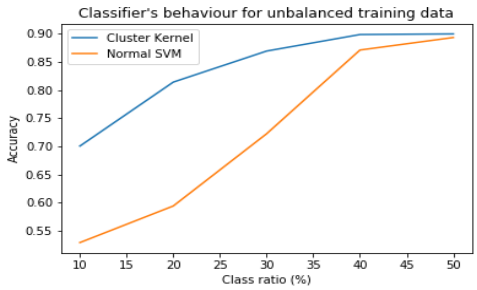


Figure 6: Effect of imbalanced training labels on SVM and Cluster Kernel. Even though, as expected, both classifiers perform better with balanced labels, the Cluster kernel is proficient even in the 10-90 % scenario. The whole training set consisted of 200 examples, and 1800 unlabeled points that are evenly distributed across both classes.

As showed in figure 5, the kernel cluster method requires many unlabeled data in order to produce good results (at least 60%). Furthermore, when the data points are added unevenly, cluster kernel produces worse results until samples from the other class are also being taken into account (figure 5b, 50 to 60 %). In figure 6, we compare normal SVM with the one utilizing the cluster kernel in imbalanced data. Interestingly the cluster kernel produces better overall results and manages to outperform SVM by a big margin in very imbalanced scenarios, due to the added information of unlabeled data used in the kernel.

4.3 Experimenting with different kernel functions for the affinity matrix calculation

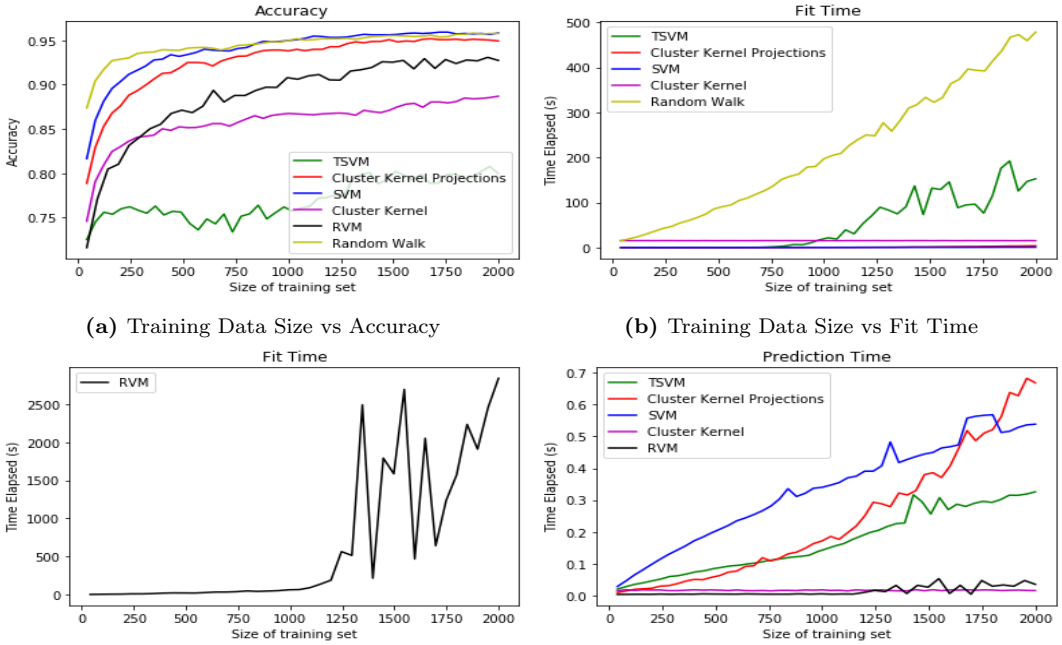
The cluster kernel approach uses the RBF kernel to compute the affinity matrix K which constitutes the core of the method. The Radial basis function is a wise choice in this case, since the produced K matrix will be positive definite and thus invertible, a property which is especially needed in projection. In table 3 we experiment with different kernels in the same dataset. As expected the RBF kernel produces the best results, while the Polynomial Kernel produced decent results. Interestingly, as we increase the polynomial degree, we achieve better results. Furthermore combining kernels via addition produced good results that lay in between the methods combined. The linear kernel had very poor results, due to the fact that produced K_{linear} matrix is ill conditioned ($k(K_{linear}) \to \infty$) and thus the projection was not accurate. Lastly, the KNN kernel is an RBF kernel matrix where for each point only the nearest k data points were taken into account, creating a symmetrical distance matrix that is utilized in random walk. As suggested in the paper [7], the transformation function used for the cluster kernel was polynomial with a power of $d = 2$. As illustrated in the table, KNN's kernel produced the best overall (including table 1) results in the USPS dataset.

Kernels	RBF	Linear	Poly $d = 2$	Poly $d = 6$	Poly $d = 9$	Poly $d = 6$ + RBF	KNN
Errors	17.3%(±2.2)	50.3%(±8.8)	26.0%(±3.7)	23.0%(±3.7)	21.1(±3.4)%	18.6%(±3.2)	10.9%(±1.6)

Table 3: Errors of Cluster kernel, using different kernel functions for the calculation of the affinity matrix. The experimentation has been done in the USPS dataset, using poly step for all our testing except KNN, were a polynomial function was used

4.4 Comparison of algorithms on different input data sizes

A comparison between different parameters as a function of the size of the training data set was made. In particular, accuracy, fit time and prediction time were compared with increasing sizes of training data set. This was made for the methods proposed in [7] (SVM, TSVM, Random Walk, Cluster Kernel and Cluster Kernel with Projections) and, as an extension, Relevance Vector Machine (RVM) [3] was applied to the data set and its results were also used for comparison.



(a) Training Data Size vs Accuracy

(b) Training Data Size vs Fit Time

(c) Training Data Size vs Fit Time (for RVM)

(d) Training Data Size vs Prediction Time

Figure 7: Comparison of different parameters as a function of the number of data points used as training in USPS data set. Figure 7a contains the performance of the accuracy. Figures 7b and 7c show the fit time. Figure 7d shows the prediction time with respect to the number of training data points.

The accuracy, in general, increases proportional to the size of the training data set, even in the TSVM algorithm which is the one with slower increase.

In figure 7b it can be seen how Random Walk is considerably slower in fitting compared to SVM, TSVM, Cluster Kernel and Cluster Kernel with projections. It must be taken into consideration that Random Walk includes both fitting and prediction in the same step when applying an EM algorithm in order to compute the necessary parameters (see *Random Walk Kernel* in section 1).

Fitting time of RVM increases exponentially with increasing training set. This is one of the main disadvantages of RVM as it is necessary to repeatedly compute and invert the Hessian matrix, requiring $O(N^2)$ storage and $O(N^3)$ computation (where N is the number of training data points) [3]. Also, we can observe several spikes. This can be due to the fact that RVM considers the most relevant support vectors and utilizes them for pruning the number of support vectors [3]. Thus, if the number of support vectors after pruning is small, the computation time required for fitting is substantially less, which explains the fluctuations seen in figure 7c.

It is worth noting that, in figure 7d, Cluster Kernel with projections has a cubic growth in prediction time as a function of the training data points. This is due to the fact that the projection of the unlabeled points must be calculated computing the kernel with respect to the training data points as explained in section 2.3, thus, the complexity of these computations increase as a function of the number of points used in training.

4.5 Application of algorithms to breast cancer dataset

In this subsection, we test the behaviour of regular SVM in comparison with TSVM, Cluster Kernel and Random Walk on a dataset different from those proposed in the paper [7]. As mentioned in the introduction, medical data can provide very good examples for semi-supervised learning since unlabeled data usually outnumber the labeled (diagnosed) data samples (patients). Thus, we utilize medical data whose collection might be challenging and in most cases imbalanced to one class as the majority of instances correspond to people who do not have the investigated disease (this data set was slightly imbalanced 60-40 %). More specifically, we performed binary classification on the breast cancer data set, offered by the scikit-learn library [14]. The breast cancer dataset is comprised of relevant features regarding the cell nuclei (e.g. texture, shape, concavity) and labels indicating whether or not the cell is diseased. The training set is constituted by 40 labeled samples. Furthermore, for TSVM, Cluster Kernel and Random Walk 100 unlabeled samples are also taken into consideration. The test data is comprised of the remaining samples. The results are averaged over 50 trials and are presented in the following table. 4:

Dataset	SVM	TSVM	Cluster Kernel	Random Walk
Breast Cancer Error	9.5%(±2.3)	9.4%(±2.2)	9%(±2.4)	11.5%(±2.1)

Table 4: SVM, TSVM, Cluster Kernel and Random Walk applied to breast cancer dataset

Breast Cancer Data Set: It is worth mentioning that the same parameters are set to all the tested classifiers (RBF kernel, $\gamma = 10^{-5}$, $C = 10$). As it can be seen in table 4, even though Cluster Kernel yields the highest accuracy, all the classifiers perform well and their rate of prediction is close.

Note that we also examined the confussion matrix, to better assess and evaluate the results. Lastly, the small gamma prefered indicates that a smooth decision function suffices to separate the two classes (as they are easily separable).

4.6 Cluster Kernel Ensemble Method

Based on our previous discussions, a possible solution to the ill-defined nature of large Cluster Kernel matrices for online learning is to train several classifiers on a fraction of the data and meaningfully combine their results (majority vote). In this spirit, we implemented an ensemble classifier and a boosting algorithm. Note that in ensemble, we are unable to utilize bagging (sampling *with replacement*), since having duplicates would lead to a non invertible matrix (repeating rows).

	Cluster Kernel	Ensemble method	Boosting
Errors	17.3%(±2.1)	17.5%(±0.7)	23.5%(±4.8)
time (s)	0.42(±0.05)	0.49(±0.06)	5.73(±0.5)
Condition number Magnitude	10^5	10^3	10^5

Table 5: Error comparison between regular Kernel and Ensemble Methods on the USPS dataset. Results were averaged over 50 iterations. In each iteration, one of the 50 subsets of 40 labeled examples were utilized for boosting and the signle Cluster Kernel. Each model in the ensemble method utilized $200 = 40 + 160$ training samples.

As we can see, the Ensemble method managed to acquire similar results with **considerably lower standard deviation**. This is expected, as ensemble methods are primarily used to decrease the model's variance. Boosting however, did not prove as fruitful. In fact, in order to obtain this result we had to lower the cost parameter C of the SVM, since boosting requires weak classifiers (with accuracy that is barely over 50%). Crucially, in the kernels of the Ensemble method (consisting of 10 Cluster Kernels utilizing 10% of the total examples) the condition number becomes two orders of magnitude smaller. We therefore conclude, that utilizing the Cluster Kernel in an Ensemble setting can prove a viable option for online learning.

5 Conclusions

Cluster kernel offers a promising extension of traditional SVM for the semi supervised case. If used accordingly, it can offer a significant accuracy boost when the labeled training data are limited (20news). It also manages to alleviate the dependency that traditional SVM has on the training label's proportion (figure 6), yet it remains susceptible to **imbalanced unlabeled** data (figure 5). Furthermore, our proposed ensemble method seems a promising solution to one of its most prominent issue, which is the numerical instabilities of the required K^{-1}.

Our results manage to capture the relative improvements from the original papers, with Cluster Kernel having a slight and substantial improvement over SVM in USPS and 20news datasets respectively. The reproduction (with the exception of TSVM on the 20news data set) is accurate, considering the error margin. Furthermore, our experiments on the Breast Cancer dataset seems to validate the results represented on paper. Cluster Kernel showcases the least error, with TSVM and SVM giving similar slightly worse results.

Cluster kernel outperforms (especially when using projection) the other examined unsupervised algorithms time-wise. For instance, despite Random Walk's exceptional accuracy, Cluster Kernel achieves similar (or greater, when utilizing KNN cluster) results substantially faster on the USPS data set (see 7b and 7d). Lastly, a combination of the cluster kernel and the random walk produced the finest results in the USPS data set (as indicated by the performance of KNN cluster Kernel (table 3)) and showed that further experimentation with the initial kernel K is advised.

References

[1] Thorsten Joachims. "Transductive inference for text classification using support vector machines". In: *Icml*. Vol. 99. 1999, pp. 200–209.

[2] Y Weiss. "Segmentation using eigenvectors: a unifying view". eng. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. IEEE, 1999, 975–982 vol.2. ISBN: 0769501648.

[3] Michael E. Tipping. *The Relevance Vector Machine*. 2000.

[4] Martin Szummer and Tommi Jaakkola. "Partially labeled classification with Markov Random Walks". In: *Advances in neural information processing systems*. MIT Press, 2001, pp. 945–952.

[5] Olivier Chapelle et al. "Choosing Multiple Parameters for Support Vector Machines". In: *Machine Learning* 46.1 (Jan. 2002), pp. 131–159. ISSN: 1573-0565. DOI: 10.1023/A:1012450327387. URL: https://doi.org/10.1023/A:1012450327387.

[6] Koji Tsuda, Taishin Kin, and Kiyoshi Asai. "Marginalized kernels for biological sequences". In: *Bioinformatics* 18.suppl$_1$ (July 2002), S268–S275. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/18.suppl_1.S268. eprint: https://academic.oup.com/bioinformatics/article-pdf/18/suppl_1/S268/630725/18S268.pdf. URL: https://doi.org/10.1093/bioinformatics/18.suppl%5C_1.S268.

[7] Olivier Chapelle, Jason Weston, and Bernhard Schölkopf. "Cluster Kernels for Semi-Supervised Learning". In: *Advances in Neural Information Processing Systems 15*. Ed. by S. Becker, S. Thrun, and K. Obermayer. MIT Press, 2003, pp. 601–608. URL: http://papers.nips.cc/paper/2257-cluster-kernels-for-semi-supervised-learning.pdf.

[8] O. Chapelle, B. Schölkopf, and A. Zien. "Introduction to Semi-Supervised Learning". In: *Semi-Supervised Learning*. MITP, 2006, pp. 1–12. ISBN: null. URL: https://ieeexplore.ieee.org/document/6280908.

[9] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]". In: *IEEE Transactions on Neural Networks* 20.3 (2009), pp. 542–542.

[10] Clever Moler. "What is the condition number of a matrix?" In: (July 2009). URL: https://www.phys.uconn.edu/~rozman/Courses/m3511_18s/downloads/condnumber.pdf.

[11] Yuchun Tang et al. "SVMs Modeling for Highly Imbalanced Classification". In: *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 39 (Mar. 2009), pp. 281–288. DOI: 10.1109/TSMCB.2008.2002909.

[12] Neto N. Berardo S. Favero E. "Active Learning with Clustering and Unsupervised Feature Learning". In: *Advances in Artificial Intelligence* 9091 (2015). URL: %5Curl%7Bhttps://link.springer.com/chapter/10.1007/978-3-319-18356-5_25#citeas%7D.

[13] Kavan Fatehi et al. "Improving semi-supervised constrained k-means clustering method using user feedback". In: *JOURNAL OF COMPUTING AND SECURITY* 1 (Feb. 2015), pp. 273–281.

[14] *Breast Cancer Dataset*. https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html.

[15] Ken Lang. *20 Newsgroups*. https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html.

[16] US Post Office. *US Post Office Zip Code Data*. https://web.stanford.edu/~hastie/StatLearnSparsity_files/DATA/zipcode.html.