New Event Detection with Nearest Neighbor, Support Vector Machines, and Kernel Regression

Jian Zhang Yiming Yang Jaime Carbonell

March 15, 2003
CMU-CS-04-118, CMU-LTI-04-180

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Support Vector Machines have received extensive attention in machine learning community and have been successfully applied in pattern recognition and regression problems. Recently, it has also been proposed to solve novelty detection problems, whose objective is to detect novel objects from existing instances. New Event Detection (NED), which can be treated as one special application of novelty detection, has been a research topic in Topic Detection and Tracking (TDT) community for several years. However, the winning technology of NED in the TDT community has remained to be the nearest neighbor method with suitable distance metric in the document vector space. In this paper we investigated Support Vector Machines and kernel regression (as a smoothed nearest neighbor method) for the NED task, and compared them to the nearest neighbor method. We conducted a set of experiments on TDT benchmark collections, and provided analysis on the failure of SVM for not being able to capture **Misses**.

This research is sponsored in part by NSF under the grant number KDI-9873009, and in part by NSF under the grant number IIS-9982226. The views and conclusions contained in this document are those of the authors and do not necessarily reflect those of the sponsors.

Keywords: I.2.6 [Artificial Intelligence]:Learning; I.5.1 [Pattern Recognition]:Models–Statistical, I.5.2 [Pattern Recognition]:Design Methodology–Classifier design and evaluation; General Terms: Algorithms; Additional Key Words: Novelty Detection, New Event Detection, Nearest Neighbor, Support Vector Machines, Kernel Regression.

1 Introduction

New Event Detection (a.k.a. First Story Detection) is the task of online identification of the earliest report for each event [1] as soon as that report arrives in the temporal sequence of documents. Being a part of the Topic Detection and Tracking (TDT), New Event Detection (NED) has been recognized as the most difficult one (Allan et al., 2000) among all tasks in TDT. Current approaches of NED mainly focus on comparing a new document to all the documents in the past, and various clustering techniques have been applied to aid the detection.

The difficulties of the NED task lie in several aspects. First, as a special case of novelty detection, it is an online unsupervised learning task, which is more difficult than traditional supervised learning tasks or retrospective unsupervised learning. Second, unlike novelty detection for handwritten digits (Kivinen et al., 2002), text data is usually sparse and high-dimensional, and the number of possible classes in the history can be up to thousands of, which greatly increases the complexity and difficulty of the task.

Yang et al. (1998) discussed the task of event tracking and detection, and proposed the GAC-INCR algorithm which achieved the best performance in TDT evaluation in recent years. In this approach, clustering techniques plus cosine similarity metric are used on top of TFIDF term weighting. Besides, time decay is also applied which gives recent stories more weights than older ones.

Tax and Ruin (1999) proposed to use a variant of Support Vector Machines called Support Vector Domain Description (SVDD), for the task of novelty detection. Schölkopf et al. (1999) proposed to use ν-SVM to estimate the support region of a high-dimensional distribution, and showed their method is equivalent to SVDD under certain conditions.

One-class document classification, which shares many properties with NED, has also been studied in the literature. In particular, Manevitz and Yousef (2001) compared several methods for one-class classification, and their results illustrated that although SVM has good performance in this task, it is very sensitive to the parameter tuning and number of features. Since it has been observed in text categorization that SVM is robust and suitable even with many relevant features (Joachims, 1998), it would be interesting to analyze what makes the difference.

Another recently proposed method, called Topic-Conditioned Novelty Detection (Yang et al., 2002), treats NED as a two-step process. Documents are first classified into corresponding topics (a group of similar events), then event detection is performed at each topic node. One advantage of this approach is that topic-informative features and event-informative features can be used at different steps, thus the confusability between stories within the same topic but

[1] "Event" and "topic" have been used interchangeably in the TDT literature for historical reasons.

not the same event can be reduced. However, our investigation here is orthogonal to that approach, i.e., we focus only on event-level detection where all the methods in this paper can be applied to.

In this paper, we compare the nearest neighbor method with Support Vector Machines and kernel regression method in the NED task. The variant of SVM we used, Support Vector Domain Description, is trying to find a hyper-ball in the feature space that can enclose all the data points and thus can be used as an estimation of the support region of the underlying density. Kernel regression is a natural generalization of nearest neighbor method in the sense that it will consider all surrounding data points with appropriate weights (decaying based on distance) rather than the nearest neighbor alone. By tuning its bandwidth we are able to investigate the sensitivity of the NED task. Since it has been reported that SVM is very robust with many features in text categorization tasks, by conducting this comparison, we would also be able to explore that whether SVM is suitable for this unsupervised task with high-dimensional and sparse text data. We would also like to investigate how the estimated support region in such a high-dimensional case (Scholkopf et al., 2000) help our novelty detection task.

2 METHODS

2.1 NEAREST NEIGHBOR FOR NOVELTY DETECTION

Nearest Neighbor method is one of the simplest methods in machine learning. Given a metric, it simply compute the distances between the test data $\vec{x}$ and all the training examples $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N$ and use the highest similarity score (lowest distance score) plus a threshold (a radius around that example) to make the novelty prediction. In this paper, we will use the Cosine similarity between two vectors, which is the same as the inner product between two normalized vectors:

$$\cos(\vec{x}, \vec{y}) = \frac{\vec{x}}{||\vec{x}||} \cdot \frac{\vec{y}}{||\vec{y}||}$$

where $|| \cdot ||$ denotes the Euclidean norm (2-norm).

Although simple, nearest neighbor with cosine similarity has been one of the most successful approaches in the NED task and many other Information Retrieval tasks like adaptive filtering.

2.2 SVM FOR NOVELTY DETECTION

Support Vector Domain Description (SVDD) is proposed by Tax and Duin (1999) for novelty detection. The basic idea is that it tries to find a hyper-ball in the feature space which can enclose all the training data (already seen examples in our case) with the minimum volume. Mathematically, it is aimed to minimize the following objective

$$F(R, \vec{c}, \xi_i) \quad = \quad R^2 + C\sum_{i=1}^{N} \xi_i$$

subject to the constraints

$$0 \leq \xi_i$$
$$(\vec{x}_i - \vec{c})^T(\vec{x}_i - \vec{c}) \leq R^2 + \xi_i$$

where $\vec{c}$ is the center of the hyper-ball and ξ_i's are slack variables measuring how far the data point is away from the surface of the hyper-ball, and as in the traditional SVM for classification, the coefficient C controls the balance between minimum volume and the tolerance errors. By incorporating those constraints into the objective function, we get the Lagrangian:

$$\begin{aligned} L(\vec{\alpha}, \vec{c}, R, \vec{\xi}) &= R^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \gamma_i \xi_i \\ &- \sum_{i=1}^{N} \alpha_i (R^2 + \xi_i - (\vec{x}_i - \vec{c})^T(\vec{x}_i - \vec{c})) \end{aligned}$$

with Lagrange multipliers $\alpha_i, \gamma_i \geq 0$. By setting the derivatives with respect to the primal variables $\vec{c}$, $\vec{\xi}$ and R to zero, we get: $\vec{c} = \sum_{i=1}^{N} \alpha_i \vec{x}_i$, $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{N} \alpha_i = 1$.

Substituting the above formulas into the Lagrangian we obtain the dual problem

$$\max_{\vec{\alpha}} \sum_{i=1}^{N} \alpha_i (\vec{x}_i^T \vec{x}_i) - \sum_{i,j=1}^{N} \alpha_i \alpha_j (\vec{x}_i^T \vec{x}_j) \tag{1}$$

subject to: $\sum_{i=1}^{N} \alpha_i = 1$ and $0 \leq \alpha_i \leq C$ and the center of the hyper-ball can be computed as $\vec{c} = \sum_{i=1}^{N} \alpha_i \vec{x}_i$. Like in SVM, we can replace the inner product $\vec{x}^T \vec{y}$ by any positive definite kernel $K(\vec{x}, \vec{y})$ in the above formula. As a result, the resulting hyper-ball will enclose data examples that are mapped into a high dimensional feature space. To test whether a new data point $\vec{z}$ is within the region we need to evaluate

$$F(\vec{z}) = R^2 - (\vec{z} - \vec{c})^T(\vec{z} - \vec{c}).$$

A test example $\vec{z}$ will be predicted as novel if $F(\vec{z})$ is less than the threshold θ.

2.2.1 Equivalence to One-Class ν-SVM

It has been shown in previous work that the SVDD method is equivalent to the one-class ν-SVM when certain kind of kernels are used. The following text follows (Scholkopf et al., 1999) to introduce the ν-SVM and summarize the equivalence derivation. The ν-SVM tries to separate the data from the origin with maximum margin. To solve the problem, we need to solve the following optimization problem:

$$\begin{aligned} \min_{\vec{w},\vec{\xi},\rho} \quad & \frac{1}{2}||\vec{w}||^2 + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i - \rho \\ \text{subject to:} \quad & \vec{w}^T\vec{x}_i \geq \rho - \xi_i,\ \xi_i \geq 0 \end{aligned}$$

By applying the Lagrange, we can maximize its dual problem:

$$\begin{aligned} \max_{\vec{\alpha}} \quad & -\frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j K(\vec{x}_i,\vec{x}_j) \\ \text{subject to:} \quad & 0 \leq \alpha_i \leq \frac{1}{\nu N},\ \sum_{i=1}^{N}\alpha_i = 1 \end{aligned} \tag{2}$$

Based on KKT conditions, ρ can be computed as $\rho = \sum_i \alpha_i K(\vec{x}_i,\vec{x}_k)$ where $\vec{x}_k$ is any training example whose $0 < \alpha_i < \frac{1}{\nu N}$.

Now, by comparing equation (1) and (2), it is clear that if $K(\vec{x},\vec{x})$ is constant, then those two methods are equivalent. It is obvious that the RBF kernel automatically satisfies this condition since $K(\vec{x},\vec{x}) = \exp(-||\vec{x}-\vec{x}||^2/\sigma^2) = 1$, and for linear kernel, since we are using normalized document vector, $K(\vec{x},\vec{x}) = ||\vec{x}||^2 = 1$. So in our problem where normalized document vector is used, those two formulations are equivalent, and we will briefly refer it as SVM in the rest of this paper.

2.2.2 Hard-Margin versus Soft-Margin

When SVM is used for classification, people usually use the soft-margin SVM mainly for the reason that it is a generalized version of hard-margin SVM, and that it handles linearly non-separable dataset. However, for one-class SVM the second issue does not exist, since for every finite training set we can find a hyper-ball that encloses all the data examples in the feature space.

Here the hard-margin SVM is the same as the soft-margin one except that all the upper bounds of α's are removed in equation (1) or (2). For our new event detection task, suppose we meet one novel story in the input stream and the system correctly identifies it as a novel story and output a score which is above the novelty threshold. Then if the next document is exactly the same (or almost the same), its prediction is still very likely to be above the novelty threshold (due to the soft margin). However, by the definition of the NED task, the latter document should clearly be a non-novel story. Though it can still be corrected by varying the novelty threshold, we believe that there is no big difference between those two formalism[2]. Plus, we can also gain efficiency since the hard-margin SVM is simplfied due to less involved constraints. Hence, we only investigate the hard-margin SVM for the NED task in our experiments.

2.2.3 Online Learning Algorithm

New Event Detection requires online predictions. That is, we need to an online version of SVM optimization algorithm. At first glance, it might seem extremely expensive to use SVM for the NED task, which usually contains tens of thousands of documents to be processed, while the number of features can also be over ten thousands. However, notice that when processing the documents, the model does not need to be re-trained when an incoming document is classified as not-novel. Retraining only happens when the input document is predicted as novel. Furthermore, when we retrain the model, since all the α's except the new one are already optimized in the last run, so similar to the idea used by the chunking algorithm, it should be very efficient if we start with the old $\vec{\alpha}$'s value to retrain the model.

We use the SMO algorithm (Platt, 1998) as the training algorithm for SVM, and only slight modifications are needed to convert it into the online algorithm as we just mentioned. Notice that although there are online approximate algorithm available for one-class SVM (Kivinen et al., 2002), it does not compute exact solution. Furthermore, it requires one-pass of the entire dataset before getting stable results, which severely violates the TDT requirement.

2.3 Kernel Regression for Novelty Detection

Kernel regression is a natural generalization of k-Nearest Neighbor regression method. Compared with the wiggy, discontinuous regression curve generated by k-Nearest Neighbor, its curve is smooth and more intuitive. And theoretically it has faster convergence rate than k-Nearest Neighbor. A popular version is the Nadaraya-Watson kernel estimator defined as follows:

Definition: The **Nadaraya-Watson** kernel estimator is defined by $\hat{r}(\vec{x}) = \sum_{i=1}^{N} w_i(\vec{x}) y_i$ where K is a kernel and the weights $w_i(\vec{x})$ are given by $w_i(\vec{x}) = \frac{K(\frac{\vec{x}-\vec{x}_i}{h})}{\sum_{j=1}^{N} K(\frac{\vec{x}-\vec{x}_j}{h})}$.

[2]Though not reported in this paper, our separate results for hard-margin SVM also support this argument.

In our experiments we still choose the Gaussian kernel [3] $K(\frac{\vec{x}-\vec{y}}{h}) = \exp(-\gamma||\vec{x}-\vec{y}||^2)$ where both h and the original σ^2 are absorbed into one parameter γ.

We use Figure 1 to further illustrate how kernel regression works as γ changes. From the graph we can see that when $\gamma \to 0$, kernel regression is simply average of all the y_i's, which is also called "oversmoothing"; and when $\gamma \to +\infty$ (means we do not limit the number of features in the document vector), kernel regression behaves more and more like the one Nearest Neighbor (1-NN) method, which is also corresponding to the "undersmoothing" case [4].

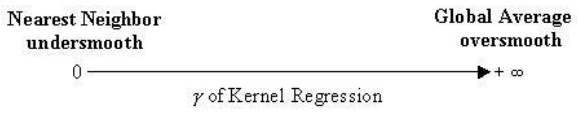

Figure 1: γ of Kernel Regression

By tuning the parameter γ of kernel regression, we are able to investigate whether using a distribution of scores will help to improve the novelty detection performance. This will help us to justify why 1-NN method works so well in the NED task.

2.4 Summary of Methods

Due to similarities between the methods we mentioned above, we summarize them in this section. Suppose at time t we have N history documents $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N$, and we are going to predict whether the current document $\vec{x}$ is novel. We also assume that the similarities between the current document $\vec{x}$ and history documents are already calculated as $s_1, s_2, \ldots, s_N$. Then, the prediction of all methods can be summarized as

$$S(\vec{x}) = \sum_{i=1}^{N} w_i(\vec{x}) s_i$$

where $w_i(\vec{x})$ is the normalized weight of document $\vec{x}_i$ in the history. To be more specific,

- Nearest Neighbor (1-NN): The $w_i(\vec{x})$ for the nearest neighbor is 1.0, while all the other w's are 0.0. It can be treated as an extreme case of the above formula

[3]Notice that the definition of "kernel" is slightly different in statistics from that in SVM. Here we use the definition in statistics. And it has been observed that the choice of the kernel, unlike the bandwidth h, is usually numerically indistinguishable in kernel regression.

[4]The radius of 1-NN method is another smoothing parameter (Baillo et al., 2000). However, since it can be adjusted by our threshold, we simply ignore it in this paper.

- SVM: The w_i is the same as α_i, the Lagrangian multiplier of the optimization problem that are learned from training data. In other words, all support vectors have positive influence on the prediction of the new document, while the non-support vectors are simply ignored in the prediction. Unlike the w's in 1-NN, the α's are fixed constant once trained and do not depend on the query point $\vec{x}$.

- Kernel Regression (KR): The normalized weight w_i is decided by the Kernel function we chose (mainly by its parameter), and as we tune the kernel parameter, it can result in a broad range of versions with different degree of smoothing, where 1-NN method is one extreme case. In other words, kernel regression will consider a whole distribution of neighbors (as opposed to 1-NN method only a single nearest neighbor will be taken into consideration).

Though it is true that in SVM the s_i's are generated by kernel $K(\vec{x}, \vec{x}_i)$, this can easily extended to Nearest Neighbor and Kernel Regression as well. However, since plugging in the kernel usually will not change the order of similarities, this change will not significantly influence the results of 1-NN and Kernel Regression.

3 EXPERIMENTAL SETUP

To compare SVM and KR with the 1-NN method in the NED task, we want to investiage how sensitive they are with respect to their parameter settings. One thing pointed out by previous research on one-class document classification (Manevitz & Yousef, 2001) is that compared with other methods, one-class SVM is more sensitive to parameter settings and thus is regarded not as robust as others like Neural Networks.

Our testbed, TDT-3 corpus [5], is made up of English stories from the last three months of 1998 as well as 120 events judged for relevance against those stories. Among those stories only a small portion are judged according to whether it discusses each of the defined events. Stories are given to the system sequentially based on their time order, and the system need to predict novelty results simultaneously[6].

In our experiments we use the standard TDT evaluation measure (TDT, 2002) to evaluation our results. The performance is characterized in terms of the probability of two types of errors: **miss** and **false-alarm** (P_{Miss} and P_{FA}). These error probabilities are then combined into a single detection cost, C_{Det},

[5]TDT-3 corpus contains stories from eight English, three Chinese, and four Arabic sources. However, only English stories are used for the New Event Detection task.

[6]The official NED task allows some deferral window. That is, the system can output decisions no later than some point after it sees the document. Though this condition makes the system more realistic, we simply ignore this information here to simplify our comparison and focus on more important factors.

by assigning costs to **miss** and **false-alarm** errors:

$$C_{Det} = C_{Miss} \cdot P_{Miss} \cdot P_{target} + C_{FA} \cdot P_{FA} \cdot P_{non-target}$$

where

- C_{Miss} and C_{FA} are the costs of a Miss and a False Alarm, respectively,
- P_{Miss} and P_{FA} are the conditional probabilities of a Miss and a False Alarm, respectively, and
- P_{target} and $P_{non-target}$ are the a priori target probabilities ($P_{target} = 1 - P_{non-target}$).

It is the normalized cost that is used in evaluating various TDT systems:

$$(C_{Det})_{Norm} = \frac{C_{Det}}{\min(C_{Miss} \cdot P_{target}, C_{FA} \cdot P_{non-target})}.$$

And when we mention cost later, we will refer to this normalized cost.

In TDT two types of evaluations are used, namely topic-weighted and story-weighted evaluations. In topic-weighted evaluation (macro-average), the cost is computed for every event, and then the average is taken. In story-weighted evaluation (micro-average) the cost is computed for all stories in the system, thus large event might have bigger impact on the overall performance. Note that in official TDT evaluation, topic-weighted cost is used as the primary evaluation measure.

In addition to the binary decision "novel" or "non-novel", each system is also required to generate a confidence score. Then its performance can be easily evaluated based on those confidence scores by simply varying the threshold. This leads to the DET curve which is extensively used in the TDT evaluation. Similar to the ROC curve, the DET curve reflects the system's performance as the threshold varies, in terms of the tradeoff between **miss** and **false-alarm**.

To sum up, the input of the system is a sequence of documents, and the system need to output online decision for each document in the stream. In this paper we will evaluate our methods with either the minimum normalized cost (as we vary the threshold) or the DET curve.

4 EXPERIMENTAL RESULTS

In this section we will describe our experiments as well as show the results. Up to now, one thing we have not mentioned is how to create document vectors from the dataset. We use the following popular TFIDF term weighting method, which is also called "LTC" term weighting in the literature:

$$TW(w_j, \vec{x}) \quad = \quad TF(w_j, \vec{x}) \cdot \log \frac{N}{DF(w_j)}$$

where w_j the jth word, $TF(w_j, \vec{x})$ is the Term Frequency (TF) of the word w_j in document $\vec{x}$, and $DF(w_j)$ is the Document Frequency (DF) of word w_j in the corpus. To make a stable estimation of the DF, we load an initial corpus before processing any test document.

We will use the normalized cost as the evaluation measure, as people did in TDT. However, the normalized cost is related to the choice of threshold, which makes the comparison tedious. So we decide to first get the minimum normalized cost by varying the threshold and then compare this single number. We will also compare some of the DET curves when necessary, which can give further information about how the normalized cost changes as we vary the threshold.

As observed in (Manevitz & Yousef, 2001), SVM for one-class classification task is very sensitive to the number of features used. Since our task is unsupervised, there is no explicit criteria to select features. Instead, we truncate our document vector length based on the significance of term weighting (which is LTC term weighting). After this truncation, each document vector can contain at most *vlen* number of features. However, this is quite different from feature selection since those remaining features for each document vector will be different across documents.

4.1 RESULTS WITH NEAREST NEIGHBOR

One thing that slightly helps the performance of nearest neighbor method is adaptive IDF. That is, the IDF will be online updated as new document comes in, which will more accurately reflect the corpus statistics.

4.2 RESULTS WITH LINEAR-SVM

One thing to notice is that in both SVM algorithms (linear-kernel and RBF-kernel), the adaptive IDF option is turned off, otherwise the SVM model needs to be re-trained for each incoming document, and the computation becomes inapplicable for our dataset (around 40k documents) [7].

4.3 RESULTS WITH RBF-SVM

Compared with linear-kernel SVM, one important factor for RBF-kernel SVM is the kernel parameter $\gamma = 1/\sigma^2$. Previous studies (Manevitz & Yousef, 2001) show that the choice of the kernel parameter is very sensitive to the results. We start from an initial value of γ: $\gamma = \frac{1}{\max_{i,j} ||\vec{x}_i - \vec{x}_j||^2} = 0.5$ which has been used as the starting point in the Support Vector Clustering framework (Ben-Hur et al., 2001), and the right equation comes from the fact that in our experiments

[7]Though it is not fair for SVM to use static instead of adaptive IDF, we also conducted separate experiments for nearest neighbor by turning off the adaptive IDF option, and results are consistent with our current ones.

all document vectors are normalized and have only positive components:

$$\max_{i,j} ||\vec{x}_i - \vec{x}_j||^2 = \max_{i,j} \left\{ ||\vec{x}_i||^2 + ||\vec{x}_j||^2 - 2\vec{x}_i^T \vec{x}_j \right\} = 2.0$$

Although we did all experiments by tuning the γ value (among 0.5, 0.7, 1.0, 3.0) for each possible *vlen*'s value, we only report the best performance for each *vlen* here.

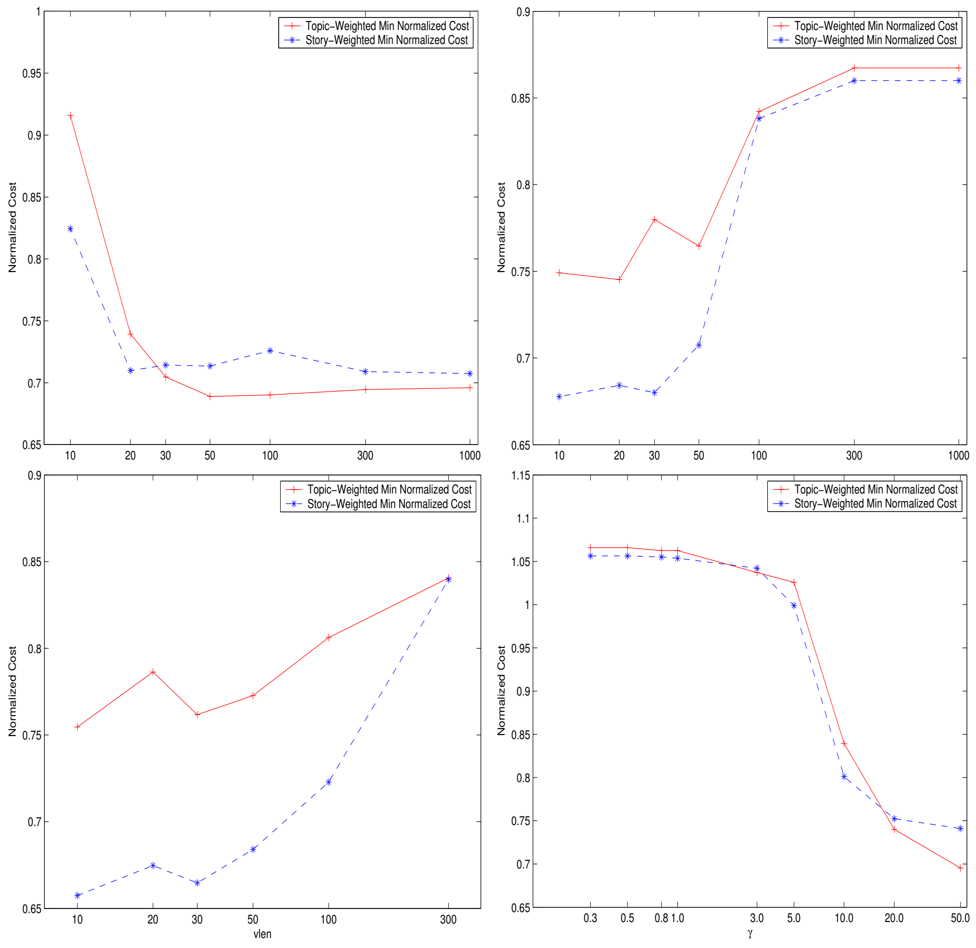

Figure 2: Results for 1-NN, LINEAR-SVM, RBF-SVM and Kernel Regression (from left to right, top to bottom)

4.4 RESULTS WITH KERNEL REGRESSION

Based on observations of nearest neighbor method, here we fix the variable $vlen = 100$ and only vary the kernel paremeter $\gamma = 1/\sigma^2$. Results are shown in Figure 2, from which we can see that when γ is as large as 50.0 the performance of Kernel Regression coincides with the nearest neighbor method, as we expected.

4.5 DET CURVE COMPARISON

For each method, we choose the best result among all parameter settings from nearest neighbor, linear SVM and RBF SVM and plot them in Figure 3 (Topic-Weighted Results) and Figure 4 (Story-Weighted Results). We did not include kernel regression since it reduces to nearest neighbor for its best condition. From the graphs we can easily see that both SVM and Kernel Regression are worse than nearest neighbor method in terms of the global trend. SVM curves, in particular, has good performance at the high-miss, low-false-alarm region and bad performance at the low-miss, high-false-alarm region.

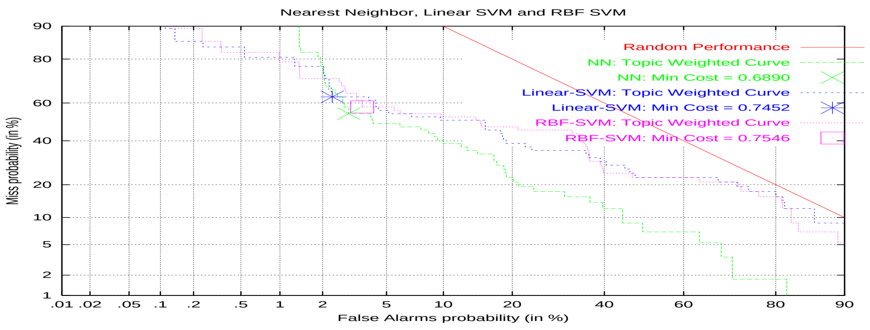

Figure 3: Topic-Weighted DET Curves

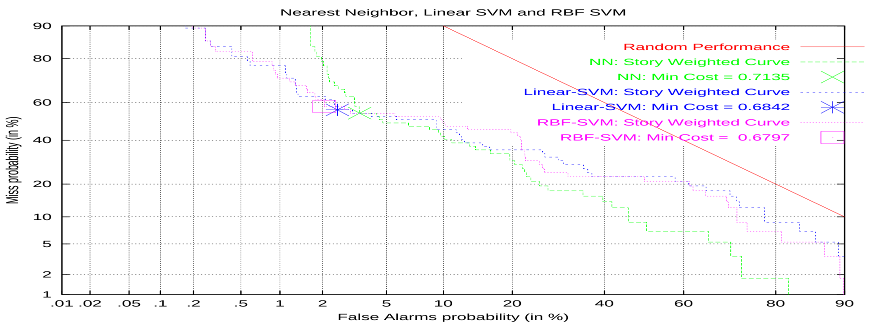

Figure 4: Story-Weighted DET Curves

4.6 DISCUSSION

Quite different from what people observed in supervised learning where SVM can deal with many features without degrading the performance, in our NED task SVM performs badly when using many features. Manevitz and Yousef also observed that for one-class document classification, SVM works better with fewer features than many features. The reason is that for unsupervised learning with SVM, since there is no target information, it is very hard for algorithms to tell which features are more informative than others by just looking at hyber-balls surrounding the data points. However, using good heuristics such as TFIDF term weighting can greatly cut off non-informative features, which will make it much easier for algorithms to learn models in a less-noisy situation.

Another important thing we found is that, as we also showed in the DET curve comparison, SVM results tend to have more misses than 1-NN. Although theoretical justifications for one-class SVM have been given in (Scholkopf et al., 1999), they only analyzed w.r.t. controling the first type of error – **false-alarm** and simply ignored the second type of error – **miss**. This can also be seen from the fact that it simply uses a hyperball to enclose all the data points without looking at how data point are distributed in the interior region. However, in most novelty detection applications reducing both kinds of errors are important to the suceess of the system. Though the RBF kernel in the extreme case will give the same result as 1-NN method (as $\gamma \to +\infty$), we believe that finding a right kernel to for a **tight** estimation of the support region is hard for this unsupervised learning task.

Our results about Kernel Regression show that only using the nearest neighbor score can achieve the better performance than considering scores generated by more neighbors with weight decay. On the other hand, smoothing in NED is an issue which can be seen from the threshold choosing (Baillo et al., 2000). However, for simplicity we only consider the minimum cost by varying the decision threshold in our experiments.

5 CONCLUSIONS AND FUTURE WORK

In this paper we compared Nearest Neighbor method, SVM, and Kernel Regression in our New Event Detection task. Our findings are not only the relative performance of those three methods in the NED task. By conducting a set of experiments, we are also able to conclude the following two issues:

- One interesting thing we found about SVM is that for our unsupervised task, when the number of features are large, it performs very badly, which is different from the results people got in the supervised learning task of text categorization where SVM has been shown to be very robust to a large number of features. This can be alleviated by using some unsupervised heuristics like TFIDF term weighting.

- SVM for novelty detection does not give a good balance between two types of errors (miss and false alarm) which are common in real world applications. In particular, it suffers from the miss rate compared with the nearest neighbor method.
- Through the tuning of bandwidth parameter of kernel regression, we can see that simply using the whole distribution of similarity scores actually hurts the performance.

In future work we plan to investigate the usage of mixture of SVM balls for novelty detection, which can both give a confident support estimation as well as have a good control of the miss rate. Besides, learning an appropriate metric through history datasets might be useful to further improve the system performance.

REFERENCES

J. Allan, V. Lavrenko, and H. Jin (2000). First story detection in tdt is hard. In *Proc. of 9th International Conference on Information and Knowledge Management (CIKM00).*

T. Brants, F. Chen, and A. Farahat (2003). A system for new event detection. In *Proc. of 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

A. Baillo, A. Cuevas, and A. Justel (2000). Set estimation and nonparametric detection. *The Canadian Journal of Statistics*, 28.

A. Ben-Hur, D. Horn, H. Siegelmann, and V. Vapnik (2001). Support vector clustering. *Journal of Machine Learning Research*, 2:125–137.

T. Joachims (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proc. of 10th European Conference on Machine Learning.*

J. Kivinen, A. Smola, and R. Williamson (2002). Online learning with kernels. *Advances in Neural Information Processing Systems*, 14.

L. Manevitz and M. Yousef (2001). One-class svms for document classification. *Journal of Machine Learning Research*, 2:139–154.

J. Platt (1998). Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Learning.*

B. Scholkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson (1999). Estimating the support of a high-dimensional distribution. *Technical Report MSR-TR-99-87, Microsoft Research.*

D. Tax and R. Duin (1999). Support vector domain description. *Pattern Recognition Letters*, 20(11-13):1191–1199.

TDT (2002). The 2002 topic detection & tracking task definition and evaluation plan. *http://www.nist.gov/speech/tests/tdt/tdt2002/evalplan.htm.*

Y. Yang, T. Pierce, and J. Carbonell (1998). A study on retrospective and on-line event detection. In *Proc. of 21th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Y. Yang, J. Zhang, J. Carnobell, and C. Jin (2002). Topic-conditioned novelty detection. In *Proc. of 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

APPENDIX: Detailed Results

Table 1: New Event Detection with Nearest Neighbor

	Topic-Weighted Scores	**Story-Weighted Scores**
	COST (MISS, FA)	COST (MISS, FA)
$vlen = +\infty$	0.6960 (0.5088, 0.0382)	**0.7074** (0.5088, 0.0405)
$vlen = 300$	0.6945 (0.5614, 0.0272)	0.7090 (0.5614, 0.0301)
$vlen = 100$	0.6902 (0.5439, 0.0299)	0.7260 (0.5439, 0.0372)
$vlen = 50$	**0.6890** (0.5439, 0.0296)	0.7135 (0.5439, 0.0346)
$vlen = 30$	0.7047 (0.5088, 0.0400)	0.7143 (0.5088, 0.0419)
$vlen = 20$	0.7393 (0.5263, 0.0435)	0.7098 (0.5263, 0.0374)
$vlen = 10$	0.9159 (0.5789, 0.0688)	0.8245 (0.5789, 0.0501)

Table 2: New Event Detection with Linear-kernel SVM

	Topic-Weighted Scores	**Story-Weighted Scores**
	COST (MISS, FA)	COST (MISS, FA)
$vlen = +\infty$	0.8673 (0.8596, 0.0016)	0.8600 (0.8421, 0.0037)
$vlen = 300$	0.8673 (0.8596, 0.0016)	0.8600 (0.8421, 0.0037)
$vlen = 100$	0.8421 (0.8246, 0.0036)	0.8381 (0.7719, 0.0135)
$vlen = 50$	0.7646 (0.6316, 0.0271)	0.7075 (0.6316, 0.0271)
$vlen = 30$	0.7799 (0.5614, 0.0446)	0.6800 (0.5614, 0.0242)
$vlen = 20$	**0.7452** (0.6316, 0.0232)	0.6842 (0.5614, 0.0251)
$vlen = 10$	0.7492 (0.6140, 0.0276)	**0.6777** (0.5439, 0.0273)

Table 3: New Event Detection with RBF-kernel SVM ("NA" entries are cases where our algorithm does not converge in reasonable amount of time)

	Topic-Weighted Scores	**Story-Weighted Scores**
	COST (MISS, FA)	COST (MISS, FA)
$vlen = 10, \gamma = 0.5$	0.7546 (0.5789, 0.0359)	0.6797 (0.5789, 0.0206)
$vlen = 10, \gamma = 0.7$	0.7676 (0.6316, 0.0278)	0.6574 (0.5263, 0.0267)
$vlen = 10, \gamma = 1.0$	0.7783 (0.6316, 0.0299)	0.6585 (0.4737, 0.0377)
$vlen = 10, \gamma = 3.0$	0.8300 (0.7544, 0.0154)	0.7423 (0.6140, 0.0262)
$vlen = 20, \gamma = 0.5$	0.7863 (0.6491, 0.0280)	0.6794 (0.5263, 0.0312)
$vlen = 20, \gamma = 0.7$	0.8047 (0.5965, 0.0425)	0.6818 (0.5439, 0.0282)
$vlen = 20, \gamma = 1.0$	0.7881 (0.6316, 0.0319)	0.6747 (0.4912, 0.0374)
$vlen = 20, \gamma = 3.0$	0.8316 (0.7544, 0.0158)	0.7492 (0.6140, 0.0276)
$vlen = 30, \gamma = 0.5$	NA	NA
$vlen = 30, \gamma = 0.7$	0.7759 (0.5614, 0.0438)	0.6814 (0.5614, 0.0245)
$vlen = 30, \gamma = 1.0$	**0.7617** (0.5614, 0.0409)	**0.6674** (0.5088, 0.0324)
$vlen = 30, \gamma = 3.0$	0.8326 (0.7544, 0.0160)	0.7242 (0.5614, 0.0332)
$vlen = 50, \gamma = 0.5$	0.7824 (0.6316, 0.0308)	0.7075 (0.6316, 0.0155)
$vlen = 50, \gamma = 0.7$	0.7727 (0.6316, 0.0288)	0.7088 (0.6316, 0.0158)
$vlen = 50, \gamma = 1.0$	0.8030 (0.7368, 0.0135)	0.6840 (0.5088, 0.0358)
$vlen = 50, \gamma = 3.0$	0.8321 (0.5614, 0.0552)	0.7094 (0.5439, 0.0338)
$vlen = 100, \gamma = 0.5$	0.8290 (0.8070, 0.0045)	0.8123 (0.7544, 0.0118)
$vlen = 100, \gamma = 0.7$	NA	NA
$vlen = 100, \gamma = 1.0$	0.8062 (0.7719, 0.0070)	0.7995 (0.7719, 0.0056)
$vlen = 100, \gamma = 3.0$	0.8372 (0.7544, 0.0169)	0.7228 (0.5614, 0.0329)
$vlen = 300, \gamma = 0.5$	0.8572 (0.8421, 0.0031)	0.8559 (0.8421, 0.0028)
$vlen = 300, \gamma = 0.7$	0.8572 (0.8421, 0.0031)	0.8559 (0.8421, 0.0028)
$vlen = 300, \gamma = 1.0$	0.8406 (0.8246, 0.0033)	0.8397 (0.8246, 0.0031)
$vlen = 300, \gamma = 3.0$	0.8592 (0.7544, 0.0214)	0.7788 (0.6491, 0.0265)

Table 4: New Event Detection with Kernel Regression

	Topic-Weighted Scores	**Story-Weighted Scores**
	COST (MISS, FA)	COST (MISS, FA)
$\gamma = 0.3$	1.0658 (0.9825, 0.0170)	1.0563 (0.9474, 0.0222)
$\gamma = 0.5$	1.0658 (0.9825, 0.0170)	1.0563 (0.9474, 0.0222)
$\gamma = 0.8$	1.0625 (0.9825, 0.0163)	1.0550 (0.9474, 0.0220)
$\gamma = 1.0$	1.0625 (0.9825, 0.0163)	1.0536 (0.9474, 0.0217)
$\gamma = 3.0$	1.0371 (0.9649, 0.0147)	1.0420 (0.9123, 0.0265)
$\gamma = 5.0$	1.0257 (0.9123, 0.0231)	0.9989 (0.7368, 0.0535)
$\gamma = 10.0$	0.8394 (0.4912, 0.0710)	0.8010 (0.4561, 0.0704)
$\gamma = 20.0$	0.7401 (0.5263, 0.0436)	0.7524 (0.5965, 0.0318)
$\gamma = 50.0$	**0.6954** (0.5439, 0.0309)	**0.7411** (0.5439, 0.0403)