Polynomial Kernel Function based Support Vectors for Data Stream Clustering

Bethapudi Naga Raju, Amarnath Gadham, M.E
M.Tech Student, QIS College of Engineering & Technology, Ongole.
Assistant Professor, QIS College of Engineering & Technology, Ongole.

Abstract:
Support vector clustering (SVC) is an important clustering algorithm based on support vector machine (SVM) and kernel methods. SVC algorithm performed better than the other traditional clustering methods, such as a global optimum, treatment of data sets of arbitrary shape, no need for specifying the number of clusters, fewer parameters, and easy treatment of high dimensional data. SV clustering consists of two phases, training based support vector machine and labeling clusters. Training phase allowing for bounded support vectors (BSVs), the existing SVStream algorithm is capable of identifying overlapping clusters. A BSV decaying mechanism is designed to automatically detect and remove outliers (noise). But outlier data doesn't optimized using linear kernel function. Proposed system will use polynomial kernel function to efficiently estimate support vectors and eliminates irrelevant data points. We represent an alternative technique for clustering stream data by using the SVM(Support Vector Machine) method. Streaming Data objects are mapped to a high dimensional characteristic space, in which support-vectors are describe a arbitrary shape training them. The outer region of the data points forms in n-dimensional space a definitive set of closed contours including the data. Streaming kdd data objects are surrounded by each contour are defined as a cluster. Experimental results show proposed method outperformed well against outliers and noise handling over existing methods. This system achieves high accuracy detection rate and less error rate of KDD CUP 1999 training data set.

1. INTRODUCTION

Support Vector Clustering (SVC) algorithm points of information are mapped from data space to the high dimensional feature space making use of a Gaussian kernel. In feature space we glance when it comes to the smallest sphere that encloses the reputation of this very data. This sphere is mapped to data space, where it forms a range of contours which enclose the results points. These contours are interpreted as cluster boundaries. Points enclosed by each separate contour are linked with the very same cluster[1]. Clearly as the width parameter of this very Gaussian kernel is decreased, the volume of disconnected contours in data space increases, linking to a rising wide range of clusters. Considering that the contours can possibly be interpreted as delineating the support of this very underlying probability distribution, our algorithm can possibly be viewed as one identifying valleys within this probability distribution[3-6]. Generalization accuracy and response time are two important criteria for evaluating classifiers when applied into real-time BI systems. Classifiers are required not simply to describe training data but in addition in order to predict unseen data. Within the example about telecommunication companies, the classifiers are anticipated not simply to describe behaviour of current customers, but as well as, at the same time, to forecast behaviour of latest customers. Generalization accuracy would most likely be estimated only by some methods due to the distribution files is all too often unknown and of course the true accuracy (generalization accuracy) couldn't be calculated. Estimated accuracy is termed testing accuracy in this particular paper. Additionally, the response speed of classifiers is predicted to get high when applied into real-time BI systems, e.g., in share market surveillance and network intrusion detection. Even users sometimes may sacrifice somewhat testing accuracy of classifiers so that you can increase the response of classifiers in real-time BI systems. The earliest strategy to handle the nonlinear data is local learning [2,4]. The initial thought of local learning is: handed a testing example, we select a handful training examples within the the vicinity of a given testing example, and train a classifier exclusively with those selected training examples, then apply this classifier into the testing example. Sticking to this idea, there are numerous works with different heuristics, e.g., SVM-KNN [2] and quick local kernel machine (FaLKM) [9], etc. The disadvantage of local learning-based methods is the nature of lazy learning, which is certainly inefficient throughout the testing phase, due to the fact they would need to perform nearest neighbor searching and classifier training for any testing examples.

The next idea which includes been widely explored is using a divide-and-conquer strategy, which is like in spirit into the mixture of experts framework [8]. Typically, the input space is partitioned into disjoint clusters, followed which a local classifier is trained on examples falling in every cluster. This technique is all too often repeated within the manner of expectation maximization (EM) [3]. Representative methods included in this family include [6], to bring up several. The most ideal problem with mixture of experts-based methods is because more likely to local minima, effectively making model selection a nontrivial task. The proposed method in the current paper shares similar flavor using this breed of methods. However, versus performing clustering and education SVMs during a iterative way, we only perform it all the global regularization way, which is in fact highly effective and efficient[7-8].

Although SVM can build classifiers with high testing accuracy, the response time of SVM classifiers still must improve when applied into real-time BI systems. Two elements affecting the response time of SVM classifiers are classified as the large number of input variables understanding that of a given support vectors. While Viaene et al. (2001) improve response time by choosing aspects of input variables, this paper works to improve the response time of SVM classifiers by reducing support vectors. According to the above motivation, this paper proposes a fresh algorithm called K-means SVM (KMSVM). The KMSVM algorithm reduces support vectors by merging the K-means clustering technique and SVM. Because the K-means clustering technique can almost preserve the underlying structure and distribution of a given original data, the testing accuracy of KMSVM classifiers might be in check to some extent even if reducing support vectors could incur a degradation of testing accuracy. Within the KMSVM algorithm, the volume of clusters is added into your training process clearly as the input parameter except the kernel parameters plus the penalty part in SVM. In unsupervised learning, e.g., clustering, it is usual that the large number of clusters is subjectively influenced by users with domain knowledge. However, in the event the K-means clustering technique is in conjunction with SVM to unravel the issues in supervised learning, e.g., classification, some objective criteria outside of applications can easily be adopted to find these input parameters. In supervised learning, determining the input parameters is known as model selection. Some methods about model selection have also been proposed, e.g., the hold-out procedure, cross-validation, and leave-one-out. This paper adopts the hold-out path to decide on input parameters because of its good statistical properties and reduced training costs. Moreover, searching strategies are necessary and just one grid search will be the most well-known. The computational price grid search is high while it is made use to determine a little more than two input parameters. In accordance to grid search, this paper gives a more practical heuristic technique to decide on range of clusters.

Intrusion detection techniques using data mining basically get into among the 2 categories; misuse recognition and anomaly recognition. In misuse detection, each example within a information set is labeled as ‘normal or ‘intrusion’ over the labeled data. These techniques can automatically retrain intrusion detection designs on different input information which include new kinds of attacks, as long as they have been labeled appropriately[1,2,3].

Unlike signature-based IDS, models of abuse are made automatically, and can feel more advanced and precise than manually created predefined signatures. A key advantage of abuse recognition techniques is their tall level of precision in detecting known attacks and their variants. Their apparent drawback is the inability to detect attacks whose times have not however been observed. Anomaly detection, however, builds designs of normal behavior, and automatically detects any deviation from this, flagging the last as suspect.

This paper presents the scope and status of our work both in misuse detection and anomaly detection. After the brief overview of building predictive models for learning from rare classes, the paper gives a comparative study of several anomaly detection schemes for identifying novel network intrusions.

2. LITERATURE SURVEY

The very first two of these questions are traditional research objectives in many different research domains in security informatics, notably in epidemiology. Among the many methods available to distinguish and delineate potential clusters, the scan statistic recently emerged as popular. This system actually works in identifying areas with clustering, and an important drawback in relation to its reliance on scanning windows of fixed shape (i.e., circular or elliptical) which suggests bias a priori in connection with shape of clustering events as well as limits power to describe changes within the spatiotemporal behaviour for instance change in cluster shape with much detail.

To handle these shortcomings, Zeng, Chang et al. (2004) and Chang, Zeng et al.(2005) proposed methodologies in accordance to support vector clustering (SVC). SVC is basically a kernel method and, much like all kernel methods, hinges on kernel transformation to high-dimensional feature space in order to make non-linear learning tractable. With this feature space a rather simple decision function, the lowest bounding hypersphere, is matched. If the representation of cluster boundaries is mapped back into the 1st input data space the cluster boundaries can easily be complex buff and made from multiple polygons. CluStream [1], which separates the clustering process into a web micro-clustering and an offline macroclustering. At first, the initial InitNumber points of information are collected and clustered by k-means to construct q initial microclusters. Every time a new data point arrives, it has been put into the closest micro cluster in case this reality is within its maximum boundary, which happens to be thought as an issue of t of one's root-mean-square deviation of the relevant data that are caused by the centroid; otherwise, a whole new microcluster is done.

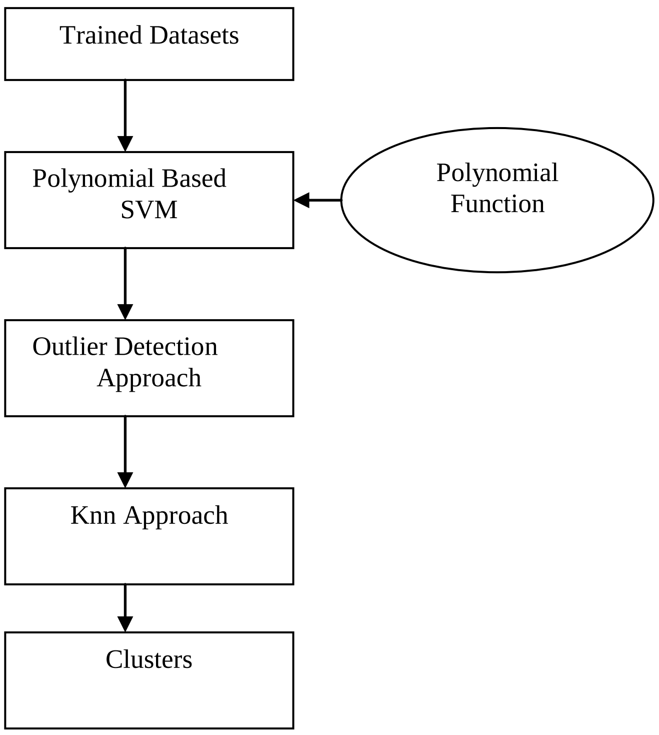
StreamKM, which processes data streams in chunks of M points of information. It summarizes each chunk by clustering its data into k centers, each weighted through large number of assigned points of information. Once M centers have also been collected, they are definitely clustered to create a smaller variety of 2k centers, with each being weighted via the sum of the weights of one's assigned centers. After one pass of a given stream data, these intermediate centers are further clustered into k final centers. Each data fact is assigned on the closest center. The chunk sizes M are set to 500 according to the four streams.

3. PROPOSED MODEL

As the network environment has grown rapidly, so has the problem of intrusions. MIT kdd99 dataset is currently available approaches to dealing with intrusions can be categorized as follows: Reconnaissance/Snooping/ Information Gathering (Probing): This kind of intrusion tries to gather useful information, including public or private data, using the powerful computing capability of a computer.
Gaining Access (User to Root): Intruders or hackers try to get access rights from a victim host, e.g., the access right of the root account.
Remote Control (Remote to Local): The intruder uses a back door program or takes advantage of application vulnerability to control remote victims through a network.
Denial of Service (DoS): The basic goal of DoS intrusion is to overwhelm the victim host with a huge number of requests. DoS intrusion is easy to achieve, and it can cause the host to crash.

In addition to the intrusions mentioned above, other intrusions may use physical or social strategies to intrude into a system by taking advantage of the vulnerability of the system or application.

1. Identification of large number of support vectors within the polynomial plane with overlapping clusters.
2. Removing the duplicate support vectors within the overlapping clusters to improve accuracy using disjoint cluster algorithm.
3. Eliminating noise after getting the disjoint clusters using k-nearest neighbor threshold mechanism. By introducing this approach time and accuracy will be exponentially increases compare to existing work.



KNN

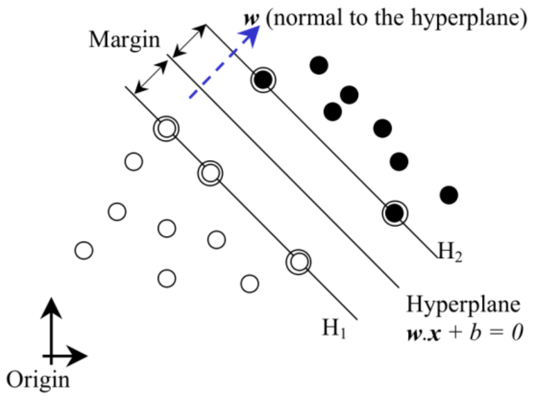
Let us start with KNN algorithm for classification. K-nearest neighbor is a supervised learning algorithm where the result of new instance is classified based on majority of K-nearest neighbor category. The purpose of this algorithm is to classify a new object based on attributes and training samples. The classifiers do not use any model to fit and only based on memory. Given a point object, we find K number of objects closest to the query point. The classification is using majority vote among the classification of the K objects.

POLYNOMIAL SVM

A support vector machine is primarily a two-class classifier. It is possible to solve multi class problems with
support vectors by treating each single class as a separate problem. It aims to maximize the width of the margin between classes, that is, the empty area

between the decision boundary and the nearest training patterns.

Given a set of points $\{x_i\}$ in n-dimensional space with corresponding classes $\{y_i : y_i \varepsilon \{-1,1\}\}$ then the training algorithm attempts to place an hyperplane between points where $y_i = 1$ and points where $y_i = -1$. Once this has been achieved a new pattern x can then be classified by testing which side of the hyper-plane the point lies on.



Step 1: Loading the training data.
Step 2: If the data is continuous then go to step 3.
Else
Go to step 1
Step 3: Apply polynomial kernel svm approach to detect the large space support vectors.
Step 4: Apply outlier detection algorithm to remove boundary points within the kernel space.
Step 5: Applying the knn approach to cluster the data points with disjoint clusters.
Step 6: Clustering results.

PROPOSED ALGORITHMS

Input: Data Points S

Output: Clusters

1. **Polynomial Based SVM:**

```
Input: a training set S
Initialization: C0={}; and S1 and S2 which are two class samples.
For i:=1 to c do
W+:=1 and w-:=1;
For j:=1 to t do
Mi:=POLYFUNCTION(S1,S2)

Outlier(Mi);
If( Pi contains O)
Then
Remove datapoint from Pi.
Else
W+=W-+thres;
End
Ci:=P2i-1 U Mi.
```

//Knn approach: Dijoint Approach

Set nn = number of nearest neighbors.
Calculate the distance between the each object and all the training samples from Ci using

Chebychev:

$$d(x,y) = max_{i=1}^{m} |x_i - y_i|$$

Sort the distance in ascending order and get Kth minimum distance as nearest neighbors.
Group the nearest neighbors as cluster
Output all clusters.

POLYFUNCTION(U,V)

Polynomial kernel has two dimensions as

$$\vec{u} = (u_1\ u_2), \vec{v} = (v_1\ v_2),$$

Consider

$$K(\vec{u},\vec{v}) = (1 + \vec{u}^T\vec{v})^2$$

i.e., that

$K(\vec{u},\vec{v}) = \phi(\vec{u})^T\phi(\vec{v})$ for some ϕ

Consider

$$\phi(\vec{u}) = \left(1\ u_1^2\ \sqrt{2}u_1u_2\ u_2^2\ \sqrt{2}u_1\ \sqrt{2}u_2\right)$$

Then:

$$K(\vec{u},\vec{v}) = (1 + \vec{u}^T\vec{v})^2$$

$$= 1 + u_1^2v_1^2 + 2u_1v_1u_2v_2 + u_2^2v_2^2 + 2u_1v_1 + 2u_2v_2$$

$$= \left(1\ u_1^2\ \sqrt{2}u_1u_2\ u_2^2\ \sqrt{2}u_1\sqrt{2}u_2\right)^T$$

$$\left(1\ v_1^2\ \sqrt{2}v_1v_2\ v_2^2\ \sqrt{2}v_1\ \sqrt{2}v_2\right)$$

$$=\phi(\vec{u})^T\phi(\vec{v})$$

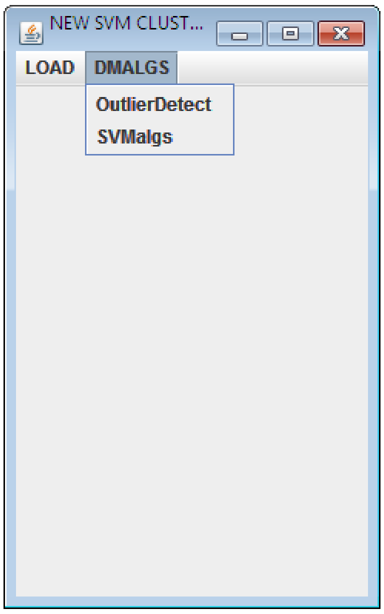
Outlier(Mi):

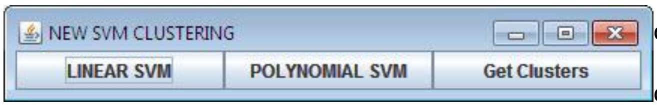
```
Randomly select m objects in Mi as initial representative objects
For each data object perform
Assign each remaining object to the group with the nearest representative object.
Randomly select a non representative object Oi.
Lamda:=Compute the similarity between Oj with Oi.
If lamda!=0
 Then
Set p1=Oi or Oj;
Else
swap Oj with Oi to form new set of k points.
Set p2=Oi,Oj
Continue.
Until no change.
```

4. EXPERIMENTAL RESULTS

All experiments were performed with the configurations Intel(R) Core(TM)2 CPU 2.13GHz, 2 GB RAM, and the operation system platform is Microsoft Windows XP Professional (SP2). Kddcup 99 dataset is used for network intrusion detection.

Experimental results:





Linear Kernel Method:

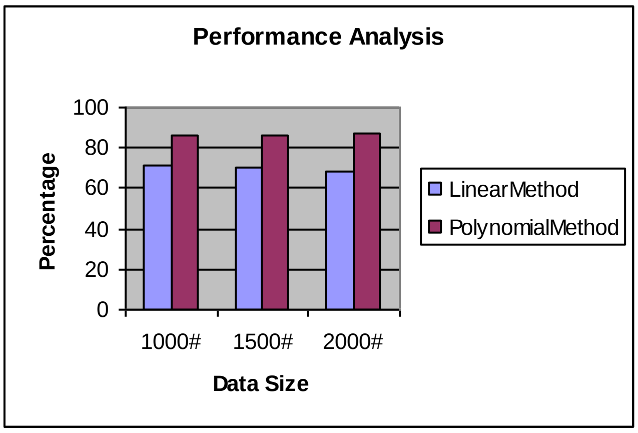
LINEARKERNELSVM

```
 -    1    * <0 0.000036 0 0 0 0 0 0 0 0 0.001961
0 0 1 0.055118 0 > * X]
 -    1    * <0 0.000009 0.000309 0 0 0 0 0 0 0
0.419608 0.419608 0 1 0.980315 0 > * X]
 +    1    * <0 0.000002 0 0 0 0 0 0 0 0 0.05098
0 0.003937 0.503937 0 > * X]
 +    1    * <0 0.010629 0.055903 0 1 0 0 0 0 0
0.005882 0.005882 0 0.26378 0.26378 0.03 > * X]
 +    1    * <0 0 0 0 0 0 0 0 0 0 0.831373 0 1 1 0
1 > * X]
 +    1    * <0 0.000002 0 0 0 0 0 0 0 0 0
0.066667 0 0 0.23622 0 > * X]
 +    1    * <0 0 0 0 0 0 0 0 0 0 0.272549 0 0 1 0
0 > * X]
 +    1    * <0 0 0 0 0 0 0 0 0 0 0.558824
0.003922 0 1 0.007874 0 > * X]
 +    1    * <0 0 0 0 0 0 0 0 0 0 0.468627 0 0 1 0
0 > * X]
 -    1    * <0 0.000059 0.050409 0 1 0 0 0 0 0
0.04902 0.04902 0 1 1 0 > * X]
 +    1    * <0 0.000002 0 0 0 0 0 0 0 0 0.05098
0 0 0.15748 0 > * X]
 +    1    * <0 0 0 0 0 0 0 0 0 0 0.256863
0.011765 0 1 0.023622 0 > * X]
 +    1    * <0 0.010629 0.055903 0 1 0 0 0 0 0
0.005882 0.005882 0 0.011811 0.011811 0 > * X]

Correctly Classified Instances       710
71.0711 %
Incorrectly Classified Instances     289
28.9289 %
```

Proposed Results:

```
Machine linear: showing attribute weights, support vectors.
Classifier for classes: normal, anomaly

       0.0216 * (attack_normalized) duration
 +      0.6326 * (attack_normalized) src_bytes
 +     -0.1467 * (attack_normalized) dst_bytes
 +     -0.0079 * (attack_normalized)
num_failed_logins
 +     -1.898  * (attack_normalized) logged_in
 +     -0.0079 * (attack_normalized)
num_file_creations
 +      0.3125 * (attack_normalized) is_guest_login
 +      1.3939 * (attack_normalized) count
 +     -0.3166 * (attack_normalized) srv_count
 +     -0.3459 * (attack_normalized) srv_rerror_rate
 +     -0.2109 * (attack_normalized) dst_host_count
 +     -2.2088 * (attack_normalized)
dst_host_srv_count
 +      0.4042 * (attack_normalized)
dst_host_srv_rerror_rate
 +      1.1513

Number of kernel evaluations: 63277 (77.532%)

KERNELSVM
Machine linear: showing attribute weights, support vectors.
Classifier for classes: anomaly, neptune
```

```
         0.0003 * (attack_normalized) duration
 +       -0.0235 * (attack_normalized) src_bytes
 +       -0.0406 * (attack_normalized) dst_bytes
 +       -0.0026 * (attack_normalized)
num_failed_logins
 +        0.002  * (attack_normalized) logged_in
 +       -0.0025 * (attack_normalized) is_guest_login
 +        0.0002 * (attack_normalized) count
 +       -0.0015 * (attack_normalized) srv_count
 +        0.0111 * (attack_normalized) srv_rerror_rate
 +        0.0012 * (attack_normalized) dst_host_count
 +        0.0011 * (attack_normalized)
dst_host_srv_count
 +       -0.0118 * (attack_normalized)
dst_host_srv_rerror_rate
 -        1.0012

Number of kernel evaluations: 54959 (93.013%)
Time taken on training data: 0.02 seconds

=== classified instances data ===

Correctly Classified Instances        863
86.3864 %
Incorrectly Classified Instances       136
13.6136 %

Cluster centroids:

Cluster 0
                0.0 tcp private S0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 278.0
1.0 1.0 1.0 0.0 0.0 0.0 0.05 0.0 255.0 1.0 0.0 0.06
0.0 0.0 1.0 1.0 0.0 0.0 anomaly
Cluster 1
                0.0 tcp http REJ 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 2.0
0.0 0.0 1.0 1.0 1.0 0.0 1.0 1.0 206.0 1.0 0.0 1.0 0.19
0.0 0.0 1.0 1.0 normal
Cluster 2
                12039.0 tcp telnet SF 798.0 276683.0 0.0
0.0 0.0 0.0 0.0 1.0 462.0 1.0 2.0 512.0 0.0 0.0 3.0
0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 255.0
37.0 0.15 0.06 0.0 0.0 0.57 0.3 0.04 0.11 normal
Cluster 3
                0.0 icmp ecr_i SF 1032.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
511.0 511.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 255.0 248.0
0.97 0.01 0.97 0.0 0.0 0.0 0.0 0.0 anomaly
Cluster 4
                36.0 tcp ftp SF 1463.0 4152.0 0.0 0.0 0.0
30.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 2.0 1.0 0.5 1.0 0.5
0.0 0.0 0.0 0.0 0.0 normal



=== Clustering stats for training data ===

Clustered Instances
0       1557 (29%)
1        607 (11%)
2         16 (0%)
3        844 (16%)
4       2267 (43%)
```



5. CONCLUSION

Feature selection is a crucial idea of Network Intrusion application. Now a days, wide range of attacks are threatening network and knowledge security. Previous single intrusion detection methods are substituted by ensemble of many different machine learning algorithms detection models. This project presented a hybrid machine learning intrusion detection model using polynomial area based knn as poly kernel svm. Using Feature selection approach kdd attacks are detected with less error rate top accuracy. Soon this work can easily be extended to feature svm with high optimized kernel functions. This work will need to implement in the real time web analytics for intrusion detection.

6. REFERENCES

[1] SV Stream: A Support Vector-Based Algorithm for Clustering Data Streams, Chang-Dong Wang, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 25, NO. 6, JUNE 2013. Data Bases (VLDB), 2003.

[2] H. Zhang, A. C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In CVPR (2), pages 2126–2136, 2006.

[3] H. Kargupta and B.-H. Park, A Fourier Spectrum-Based Approach to Represent Decision Trees for Mining Data Streams in Mobile Environments, IEEE Trans. Knowledge Data Eng., vol. 16, no. 2, pp. 216-229, Feb. 2004.

[4] P. Zhang, X. Zhu, and Y. Shi, Categorizing and Mining Concept Drifting Data Streams, Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, 2008.

[5] P. Wang, H. Wang, X. Wu, W. Wang, and B. Shi, "A Low- Granularity Classifier for Data Streams with Concept Drifts and Biased Class Distribution," IEEE Trans. Knowledge Data Eng., vol. 19, no. 9, pp. 1202-1213, Sept. 2007.

[6] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-Based Clustering over an Evolving Data Stream with Noise,"

Proc. Sixth SIAM Int'l Conf. Data Mining, 2006.

[7] C.C. Aggarwal, J. Han, J. Wang, and P.S. Yu, "A Framework for on-Demand Classification of Evolving Data Streams," IEEE Trans. Knowledge Data Eng., vol. 18, no. 5, pp. 577-589, May 2006.

[8] S. Hashemi, Y. Yang, Z. Mirzamomen, and M. Kangavari, "Adapted One-versus-All Decision Trees for Data Stream Classification," IEEE Trans. Knowledge Data Eng., vol. 21, no. 5, pp. 624- 637, May 2009.

[9] N. Segata and E. Blanzieri. Fast and scalable local kernel machines. Journal of Machine Learning Research, 11:1883–1926, 2010