An Efficient Active Training Algorithm for SVM-based Binary Classifiers

Sanddeep Gutta and Qi Cheng

Abstract—Pattern classification has been a key task in many scientific and engineering applications. In this paper, the problem of support vector machine (SVM) based binary classification is considered. Though the SVM is an effective supervised learning algorithm, its high computational complexity often limits its real-time implementation. The support vector machines require enormous amount of training time for problems of large magnitude. In this paper, a new efficient active training approach is proposed, which makes use of the fact that the SVM decision boundary is completely characterized by a small subset of the training data. The proposed method is evaluated both theoretically and experimentally using the benchmark KDD 1999 intrusion detection datasets. The proposed method is also compared with some of the existing training methods. The experimental results show that the proposed method can successfully reduce the training time without significantly degrading the classification performance.

Keywords—Classification, Support Vector Machine, Active Training, Clustering, Convex Optimization.

I. INTRODUCTION

Pattern classification refers to a method of classifying the given input data into a number of different categories. This can be viewed as the problem of searching for patterns in the data, which is fundamental and has numerous scientific and engineering applications, e.g., cyber attack (intrusion) detection, fault detection and isolation in structural health monitoring, event region detection using wireless sensor networks, target detection and recognition, etc.

There have been many classification techniques developed in the past. More recently, machine learning principles became popular to design classifiers owing to the fact that many classification tasks which are difficult to describe algorithmically can be performed well by systems which can learn by exposure to many related examples. In other words, such classifiers mimic our human neural and cognitive systems and learn by looking at the given examples (a.k.a. training set). In this paper, we consider the problem of binary classification, i.e., classification of the input data into two categories. Specifically, we focus on support vector machine (SVM) based binary classifiers. The developed techniques can be easily extended to m-ary classifiers.

Support vector machines (SVMs) are generally considered to be the best off-the-shelf supervised classifiers. SVMs are maximal margin classifiers which tend to separate the data with a large gap/margin. That is, SVMs tend to find a decision boundary (hyperplane) which lies at a maximum distance from both classes. The main drawback of support vector machines is that the training procedure has a very high computational complexity. It is mainly due to the factors like the kernel trick, quadratic programming (QP) based solutions, etc. This greatly limits the real-time implementation of SVMs, especially in applications involving very large datasets like cyber attack detection. Hence, the main challenge is how to efficiently reduce the training time required when working with large datasets.

Several approaches like greedy approximation of the kernel matrix [2], low-rank kernel representation [3], etc. are proposed to address this issue. However, the problem with these approximation-based approaches is that they have a significant impact on the classification performance [4]. There are other approaches which try to avoid the quadratic program in the SVM algorithm. However, they still involve the kernel trick which has the high computational complexity. On the other hand, there are some efficient training approaches which do not require the approximation in the SVM algorithm. These approaches are based on the fact that the decision boundary of an SVM depends only on a small subset of the training data (generally referred to as support vectors). These methods are categorized as active training approaches. However, the key issue is how to identify the training samples that are the support vectors.

In recent years, increased research has been focused on the appropriate training data selection for SVMs. Tong and Koller propose an approach where the SVM learner (classifier) sequentially selects the training samples [5]. Three selection approaches are proposed using the notion of a version space. In [6], Shin and Cho propose a method which selects training samples near the boundary using the neighborhood properties of samples. In [7], the training sample selection is done using the k-means clustering technique. In [8], Abe and Inoue extract the samples close to the boundary using the Mahalanobis distance. In [9], a reduced SVM (RSVM) is proposed which uses random sampling. In [4], Li et al. propose a method based on edge detection. Fuzzy clustering based training data selection is proposed in [10]. However, most of these methods still tend to spend considerable amount of time for selecting the best training samples. Some of them even require training an SVM at every step. Such training approaches may not be suitable in applications like cyber attack detection, where fast online training and quick responses are required.

In this paper, we propose a new approach to scale down the
training set by eliminating the non-relevant training samples present in the training set. This belongs to the category of active training approaches. By doing this, the training time can be much reduced without significantly degrading the classification performance of the SVM. The main challenge here is how to detect those non-relevant samples in the training set. We solve this problem by finding clusters present in the data and detecting/eliminating those samples which are well inside the clusters.

The remainder of the paper is organized as follows. The problem formulation is provided in Section II. The proposed active training approach is described in Section III. The analysis of the proposed active training approach is done in Section IV. In Section V, the proposed approach is evaluated using the KDD 1999 dataset. Conclusions are provided in Section VI.

II. PROBLEM FORMULATION

The problem of supervised binary classification is formulated as follows. The training set \( T = \{(x_i, y_i) ; i = 1, ..., n\} \) is given. The input vector (sample) \( x_i \) has \( d \) features, i.e., \( x_i \in \mathbb{R}^d \). The label to the input vector \( x_i \) is \( y_i \in \{-1, +1\} \) corresponding to two different classes. Classifier design is to learn a function \( y = f(x) \) which not only classifies the training data accurately but also generalizes well to samples with unknown labels.

The SVM classifier is widely used and very efficient. It tends to separate the samples of different classes with maximum geometric margins possible. Theoretically, a support vector machine operates in two steps:

1. The \( d \)-dimensional original input space \( X \) is transformed into a higher dimensional feature space (Hilbert space) \( H \) through a general nonlinear mapping \( \Phi \).
2. The separating hyperplane is constructed in the \( H \) space and it has the maximum possible margin. This maximum margin hyperplane is the solution of a quadratic program (QP).

In the first step, the kernel trick is generally used which does not require the explicit computation of the nonlinear mapping \( \Phi \).

Mathematically, the SVM classifier is defined as

\[
y = f(x) = \text{sgn} (w^T \Phi(x) + b)
\]

where \( w \) and \( b \) are the parameters of the SVM. The optimal values of the SVM parameters are found through a quadratic program which maximizes the margin between the two classes. Since the SVM generates a linear hyperplane (decision boundary), in order to generalize to nonlinearly separable data, the kernel trick is used to transform the input space to a high dimensional space, where the data becomes linearly separable. Thus, the SVM classifier mainly involves two steps - input space transformation through the kernel trick, and generation of the decision boundary by solving a quadratic program. The computation involved in both of these is directly proportional to the size of training data. If \( n \) is large, then the computational complexity for training an SVM may be prohibitively high. Our goal is to find a subset \( D \subset T \) which contains only the relevant samples for building the decision boundary.

III. TRAINING SUBSET SELECTION ALGORITHM

In the active training approach, our goal is to scale down the training data by retaining the most relevant samples (most informative) and removing the least relevant samples. The main issue here is to determine which samples are the most informative and which are not. It is known that the samples which are close to the decision boundary are more important to form the boundary than the ones which are far away. In other words, the samples which are close to the boundary tend to be more informative for SVMs. We make use of this fact to design our active training approach.

A. Training Data Clustering

First, we form multiple clusters for each class present in the given training set. The basic idea is that the samples which are well inside each cluster have little contribution to determining the decision boundary, thus can be safely eliminated. One way to do this is to use any existing clustering based technique to form the clusters. However, this simple and heuristic approach may not perform well. This is because most of the existing clustering techniques tend to form spherical clusters only. But the true clusters in the data need not always be spherical. They can exist in any arbitrary shape. We need to accurately model the true shapes of clusters present in the data, in order to accurately detect the samples which are well inside the clusters. In this paper, we propose a more sophisticated method for this task.

We define a cluster, in a more general way, to be the convex hull of a set of data points, which is the set of all convex combinations of the points. In other words, it is the smallest convex set that contains all the points. Thus, the cluster of \( p \) points \( x_1, x_2, ..., x_p \in \mathbb{R}^d \) is defined as

\[
\text{conv} \{x_1, x_2, ..., x_p \} = \{ \sum_{i=1}^{p} \lambda_i x_i | \sum_{i=1}^{p} \lambda_i = 1, \lambda_i \geq 0 \}
\]

(2)

In order to determine the center of a cluster, we need to calculate the center of the convex hull. Since the convex hull of a set of points is a polyhedron defined by its vertices, finding the exact center of this polyhedron is in general difficult, especially in high dimensional spaces. To simplify the calculation, each cluster (convex hull) is approximated using the Löwner-John ellipsoid, which is the minimum volume ellipsoid containing the cluster (convex hull). There are many nice properties of ellipsoids. They are generally considered to be an universal geometric approximator of convex sets as they have sufficient degrees of freedom. Furthermore, ellipsoidal approximation is invariant under affine coordinate transformations. Finding this Löwner-John
ellipsoid is a convex optimization problem [11]. The minimum volume ellipsoid containing the finite set of points \( \{ x_1, x_2, ..., x_p \} \) is the same as the minimum volume ellipsoid containing the polyhedron \( \text{conv} \{ x_1, x_2, ..., x_p \} \), which can be obtained by solving the following convex optimization problem

\[
\begin{align*}
\text{minimize} & \quad \log \det A^{-1} \\
\text{subject to} & \quad \| Ax_i + b \|_2 \leq 1, \quad i = 1, ..., p
\end{align*}
\]

where the variables are \( A \) and \( b \). Here in (3), we have an implicit constraint: \( A \) must be positive definite. The condition \( \| Ax_i + b \|_2 \leq 1 \) defines an ellipsoid whose volume is proportional to \( \det A^{-1} \). This way of parameterizing the ellipsoid greatly helps us in formulating the computation of Löwner-John ellipsoid as a convex optimization problem. Generally, an ellipsoid with center \( e \) is defined as \( \{ x \mid \| x - e \|_2 \leq 1 \} \). By transforming the previous formulation of the ellipsoid in (3) into this general formulation, we find the center to be \( e = -A^{-1}b \).

Our proposed clustering algorithm for sample elimination is given as follows. We initially assume that the number of clusters of each class in the dataset is known to us. Later in this section, an approach to find the optimal number of clusters is given.

1. Let \( k_1 \) and \( k_2 \) be the number of clusters of classes 1 and 2 respectively in the training data.
2. First, \( k_1 \) convex hulls are initialized randomly. That is, \( k_1 \) samples are initially chosen as centers and correspondingly \( k_1 \) convex hulls are formed with the samples close to these \( k_1 \) initialized clusters.
3. Every convex hull (cluster) is characterized by its Löwner-John ellipsoid center, which is defined as the center of the minimum volume ellipsoid that contains the convex hull (cluster). For every convex hull formed, the corresponding Löwner-John ellipsoid is determined by (3).
4. The main loop of the algorithm includes these steps until convergence:
   (i) Each training sample is assigned to the closest cluster center (Löwner-John ellipsoid optical point).
   (ii) The cluster center is updated to be the Löwner-John ellipsoid center of all the training samples assigned to it.
5. The convergence criterion is: Stop when the changes in the cluster center update step are small.
6. The same procedure is followed for class 2 with \( k_2 \) clusters.

B. Training Data Elimination

After the convergence of the above active training algorithm, we have the Löwner-John ellipsoids defined for every cluster in the training data. In order to detect the training samples which are well inside the clusters, these Löwner-John ellipsoids can be shrunk so that they lie well inside the clusters. These Löwner-John ellipsoids have a unique property that when shrunk by a small factor (generally proportional to the dimension of the space) about their centers, they lie well inside the convex hull. All those samples which are inside the shrunk ellipsoids are considered to be deep inside the clusters, hence are assumed to be not informative and are removed from the training set. The ellipsoid shrinking factor \( q \) is not constant. It varies from one cluster to another. Typically, for each cluster, \( q \) is chosen to enclose some predefined percentage of samples in the cluster. This is illustrated (in \( \mathbb{R}^2 \) space) in Figure 1.

C. The Number of Clusters Estimation

As mentioned before, we need to know the number of clusters present in the data beforehand. That is, we need to know the values of \( k_1 \) and \( k_2 \). For estimating this optimal number of clusters present in the data, we adopt the following approach: find the optimal number of clusters which minimizes the distances within each cluster (distances between the cluster center and other points in the cluster) and maximizes the inter-cluster distance (distance between two different clusters). Mathematically, the optimal number of clusters \( (k) \) is found by minimizing the following function

\[
\min_{k(x_i)} \sum_{j=1}^{n} \left( \sum_{i=1}^{n_j} \| x_{ij} - c_j \|_2 - \sum_{j=1}^{n} \| c_j - c_{ij} \|_2 \right)
\]

where \( c_j \) is the center of cluster \( j \), \( n_j \) is the number of samples in cluster \( j \) and \( x_{ij} \) is the \( i \)th sample in cluster \( j \). In order to do this, we adopt a pruning approach, where we initially start with a relatively high number of clusters and gradually remove them until the performance (optimal distance values) stays the same. The conventional \( k \)-means clustering is used to update the cluster centers here.

IV. COMPLEXITY AND PERFORMANCE ANALYSIS

In this section we provide an end-to-end analysis of the proposed active training algorithm. Our main objective is to show that the complexity of training SVMs is far reduced by using the proposed active training approach without significant degradation in performance.
A. Complexity Analysis

First we present the complexity of training SVMs using the conventional training procedure (using the entire training set). In general, support vector machines try to minimize the following function in a feature space,

\[
\begin{align*}
\min_w & \quad \|w\|^2 + C \sum_{i=1}^n y_i \lambda_i \\
\text{s.t.} & \quad y_i (w^T x_i + b) \geq 1 - \lambda_i, i = 1, \ldots, n
\end{align*}
\]

where \(w\) and \(b\) are the parameters of the SVM, \((x_i, y_i) : i = 1, \ldots, n\) is the training set, and \(C\) is the regularization parameter. In practice, the dual of the above optimization problem is solved, which is a quadratic programming (QP) problem.

\[
\begin{align*}
\max \lambda_i & = \sum_{i=1}^n \frac{1}{2} \sum_{j=1}^n y_i y_j \lambda_i \lambda_j \langle x_i, x_j \rangle \\
\text{s.t.} & \quad 0 \leq \lambda_i \leq C, i = 1, \ldots, n, \\
& \quad \sum_{i=1}^n \lambda_i y_i = 0
\end{align*}
\]

In the above dual QP problem, \(\lambda_i\)'s are the Lagrange multipliers. From the KKT dual complementarity conditions, we have \(\lambda_i = 0\) when \(y_i (w^T x_i + b) > 1\). This implies that the \(\lambda_i\)'s corresponding to the training samples that are far away from the decision boundary will be zero, and the \(\lambda_i\)'s corresponding to the training samples close to the boundary will be non-zero. The above QP problem is generally solved in a higher dimensional feature space. Kernel trick is generally employed. So the inner product in (6) is replaced by the corresponding kernel \(K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle\). The two standard approaches used to solve the above QP problem are: chunking QP method [12] and sequential minimal optimization (SMO) method [13].

In the chunking QP method, first the original QP problem is broken down into a series of smaller QP problems, which identify the non-zero Lagrange multipliers. The basic idea in this method is to scale down the size of the kernel matrix by discarding all those elements which correspond to zero Lagrange multipliers. Finally, the scaled down QP problem (6) is solved at last. Though this method scales down the size of the problem, the complexity of this method is still high for large training sets as the method still involves matrix operations, computation of gradients of \(D(\lambda)\), etc. In general, this method is found to have complexity \(O(n^2)\) [13].

Sequential minimal optimization (SMO) is a simpler and more efficient algorithm for solving the above QP problem. The SMO algorithm breaks the original QP problem into a series of smaller QP sub-problems. At every step, the SMO solves a smaller optimization problem of finding the optimal values of two Lagrange multipliers, and updates the SVM accordingly. The main advantage of the SMO lies in the fact that solving for two Lagrange multipliers at each step is done analytically, thus avoiding the matrix computations and standard QP calculations on the whole. In general, the SMO method is found to have complexity \(O(n^2)\) [13]. In this paper, we use the SMO method as it is more efficient.

For the analysis purpose, the proposed algorithm is divided into three phases where most of the computation happens:

1. Calculating the number of clusters present in the data.
2. Computing the Löwner-John ellipsoids.
3. Training the SVM using the new (active) training set.

Initially the number of clusters present in the data is estimated. A pruning approach is adopted here. In each step, the function in (4) is calculated. This clearly implies that the complexity involved in this phase is \(O(n)\). The next phase involves the computation of Löwner-John ellipsoids by solving the optimization problem (3). The problem (3) is a second-order cone programming (SOCP) problem. This SOCP problem can be solved using the interior-point methods. In this paper, we have used the self-dual minimization method. The complexity of this self-dual minimization method is \(O(d^3n)\) [14]. Finally, after the extraction of the active training set containing \(k(<n)\) training samples, the SVM is trained using this training set. The complexity of this phase is \(O(k^2)\), as we use the SMO method. Thus the total complexity of our active training algorithm is \(O(n) + O(d^3n) + O(k^2)\) (7) whereas the complexity of conventional SVM training is \(O(n^2)\). With \(k<<n\), this clearly demonstrates the reduction in the complexity achieved by using the proposed active training algorithm.

B. Performance Analysis

Our main objective here is to decrease the complexity of SVM training without any huge degradation in the classification performance of the SVM. This is taken care in the design of our active training algorithm where we try to omit only the least relevant samples. The overall performance of our active training algorithm mainly depends on how accurately the true clusters in the data are identified. Two main factors which are found to be influential here are the overlap between different clusters and the accuracy in estimating the true number of clusters present in the data. The first issue is a more general problem and is mainly data dependent. However, the second issue of calculating the true number of clusters has been addressed efficiently in our proposed active training approach.

V. EXPERIMENTAL RESULTS

The proposed active training approach is evaluated on the popular 1999 KDD intrusion detection dataset [15]. This dataset is widely considered to be the only publicly available benchmark intrusion detection dataset. The dataset covers normal connections and four major attack categories: DoS, R2L, U2R, and Probe. The data stored in the 1999 KDD datasets are feasible measurements of connections under some well defined protocol [15]. There are 41 features representing 41 measurements for each connection. Since the datasets come
from three different sensors: network sniffing data from the
sniffer (sensor 1), Sun Solaris BSM audit data from a Solaris
host (sensor 2), and disk dump data from three UNIX
machines (sensor 3). By nature, the 41 features in the dataset
are grouped into three feature subsets, each for one sensor.

In our experiments, we group all the attacks into one
category and consider the 10% KDD normal vs. attack dataset.
The dataset has 494,021 training records (samples). As
mentioned earlier, this is further divided into three subsets
Corresponding to the three different sensors. Initially, 10% of
these datasets are set aside as validation datasets. The
performance of the proposed active training approach is
evaluated upon these validation datasets. Support vector
machines (SVMs) with a Gaussian kernel are used in the
experiments.

The proposed approach finds the optimal number of clusters
for normal (class 1) and attack (class 2) classes to be 27 and
34, respectively. The $q$ factor for each cluster is chosen
to enclose around 65% of the samples present in the cluster, thus
eliminating around 65% of the original training samples. The
performance of the proposed algorithm is compared with the
conventional SVM without any active training and also with
the existing active training approaches using neighborhood
properties [6] and the k-means cluster [7].

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>PERFORMANCE OF PROPOSED ACTIVE TRAINING APPROACH (SENSOR 1 DATASET)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional SVM (without any active training)</td>
<td>Neighborhood properties based active training</td>
</tr>
<tr>
<td>False alarm rate</td>
<td>1.62%</td>
</tr>
<tr>
<td>Missed detection rate</td>
<td>0.23%</td>
</tr>
<tr>
<td>Total training time (in minutes)</td>
<td>153</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>PERFORMANCE OF PROPOSED ACTIVE TRAINING APPROACH (SENSOR 3 DATASET)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional SVM (without any active training)</td>
<td>Neighborhood properties based active training</td>
</tr>
<tr>
<td>False alarm rate</td>
<td>0.57%</td>
</tr>
<tr>
<td>Missed detection rate</td>
<td>0.94%</td>
</tr>
<tr>
<td>Total training time (in minutes)</td>
<td>124</td>
</tr>
</tbody>
</table>

The above tables show the false alarm and missed detection
rates on the validation datasets of different sensors. The results
clearly show the performance of the proposed active training
approach on reducing the training time, without affecting the
classification performance of the SVMs significantly. The
time taken by the active training method based on
neighborhood properties [6] is almost equal to that of the
conventional SVM. This is due to the fact that the algorithm
requires the calculation of k-nearest neighbors for a large
number of training samples in the training set. Though the k-
means clustering based active training method [7] drastically
reduces the training time, the classification performance was
significantly affected. This seems to be the result of using a
simple clustering technique in the algorithm. In contrast, the
proposed active training method is successful in reducing the
training time without having any significant effect on the
SVM classification performance. Though there is an increase
in the false alarm and missed detection rates, this increase is
less when compared with the decrease in the training time
taken. Hence, this slight degradation in the performance is
generally acceptable in view of the significant decrease in the
computational complexity of the overall system. This also
demonstrates the real-time adaptability of the proposed active
training approach.

VI. CONCLUSIONS

In this paper, an efficient active training approach is
proposed. The proposed method employs an Löwner-John
ellipsoid based clustering technique to accurately capture the
clusters present in the data. The least informative samples are
detected by the approach, and are eliminated from the training
data. This greatly reduces the computational complexity in the
SVM training process. The complexity analysis of the
proposed method is given, which theoretically shows the
reduced computational complexity of the method. To analyze
the effectiveness of the proposed method in reducing the
training time while maintaining the good SVM classification
performance, the proposed method is evaluated on the
benchmark KDD 1999 intrusion detection datasets. A
comparative evaluation of the proposed method with some of
the existing training methods is also given. The proposed
approach proved to be successful in reducing the training time
without significantly affecting the classification performance,
which demonstrates its real-time adaptability.

REFERENCES

binary classifiers,” Proceedings of the International Conference on
Smart Technologies (ICST), Chennai, India, Jan. 2011.

machine learning,” Proceedings of the Seventeenth International

kernel representations,” Journal of Machine Learning Research, 2001,
pp 243-264.

datasets,” Proceedings of International Joint Conference on Neural
Networks, Atlanta, Georgia, USA, June 2009.
Sandeep Gutta received the B.E. degree in electronics and communication engineering from Andhra University, Andhra Pradesh, India, in 2008. He is pursuing the M.S. degree in electrical engineering at Oklahoma State University, Stillwater, OK, USA. His research interests are in the areas of statistical signal processing and machine learning.

Qi Cheng (M'06) received the B.E. degree in electrical engineering (highest honors) from Shanghai Jiao Tong University, Shanghai, China, in July 1999, and the M.S. and Ph.D. degrees in electrical engineering from Syracuse University, Syracuse, NY, in 2003 and 2006, respectively. From 1999 to 2000, she worked as a System Engineer at Guxin Lucent Technologies Network Technologies Company, Ltd., Shanghai, China. Since August 2006, she has been with Oklahoma State University, as an Assistant Professor with the School of Electrical and Computer Engineering. Her area of interest mainly focuses on statistical signal processing and data fusion with applications in wireless communications and distributed sensor networks.