Support Vector Machine for improving Performance of TCP on Hybrid Network

Angela Makolo
Department of Computer Science
University of Ibadan
Ibadan, Nigeria
amakolo@yahoo.com

ABSTRACT
The Internet transport protocol, Transmission Control Protocol (TCP) by design treats all packet losses as an indication of congestion and reacts to such by reducing its rate. This reduction is not justified on hybrid network where a substantial number of packet losses are due to random errors of wireless link. It leads to underutilization of network resources. Differentiating the cause of packet loss is thus important to enable TCP take actions to control congestion only when the loss is caused by congestion. This work presents the use of a machine learning algorithm-support vector machine (SVM) in differentiating between the two types of losses. The model was built using a labeled dataset consisting of 26,191 loss instances. The SVM model achieved 95.97% accuracy, this shows a substantial improvement in throughput without compromising TCP-friendliness on hybrid network.

Keywords- TCP, congestion control, machine learning, support vector machine.

1. INTRODUCTION
TCP was originally developed for wired network, where a packet loss is an indication of congestion on the network. It therefore reacts by reducing its sending rate in order to control congestion. Wired/Wireless hybrid networks have characteristics different from wired network. They are prone to random packet losses that occur not only as a result of congestion but other link errors. TCP however has no mechanism for differentiating congestion induced losses from error induced losses. It therefore reduces its sending rate each time a packet loss occurs. This leads to an underutilization of network resources on hybrid network.

To improve the performance of TCP on hybrid network there is need for a mechanism that enables it distinguish between the two types of losses.

In this paper we explore the use of SVM in building a classification model for the differentiation of the two types of losses.

2. RELATED WORKS
A lot of effort has been made to differentiate between congestion induced losses and error induced losses on hybrid network. These can be categorized into two: explicit loss discrimination and implicit loss discrimination. Explicit loss discrimination relies on the support of network apparatus to perform differentiation. In this case, the sender is explicitly informed of the type of packet loss by network routers or the end receiver.

Implicit loss discrimination determines the type of packet loss by assumptions and calculations made on measurements of network parameters which can be easily obtained at the end systems. This method requires the least amount of change in the network and can be implemented by endowing one of the terminal devices with a classification algorithm. Loss differentiation decision can be made based on adhoc rules formed using network parameters such as RTTs, ROTTs, and IAT. For example, Biaz [5], mBiaz [6], and the statistical packet loss discrimination (SPLD) [7] use packet inter-arrival times, the Zigzag [7] and the Spike [3] use relative one-way trip times, and the Vegas predictor [8] uses...
round trip times. The limitation of these methods is that the
decision-making thresholds are difficult to be determined.
Also, studies [9] have shown that using only one feature may
not provide a good accuracy.

Another approach is to combine multiple features and apply
algorithms from data mining and machine learning. These
can learn the relationship between the features and make
decision based on a derived model. Machine learning
approach was first used in [10]. Multiple features were
combined from which a learning algorithm automatically
builds a model. Based on the derived model, congestion loss
and wireless loss can be differentiated.

3. METHODOLOGY

3.1 Support Vector Machine

SVM is employed here to classify the loss causes. Support
Vector Machines is an effective statistical learning method
for pattern recognition [11]. The SVM based on statistical
learning theory has many advantages. One, unlike other
nonparametric techniques such as nearest-neighbors and
neural network that are based on the minimization of the
empirical risk, SVM operates on another induction principle,
called structural risk minimization, which can overcome the
problem of over fitting and local minimum and gain better
generalization capability. Two, Kernel function method
applied in SVM overcomes the problem of dimensionality
effectively without increasing the computational complexity.
Three, SVM has demonstrated higher generalization
capabilities in high dimensional space and spare samples.
Four, unlike many learning algorithms, SVM leads to good
performances without the need to incorporate prior
information. Moreover, the use of positive definite kernel in
the SVM can be interpreted as an embedding of the input
space into a high dimensional feature space where the
classification is carried out without using explicitly the
feature space.

A classification task usually employs training and testing data
sets that consist of several data instances. Each instance in the
training set contains one target value (class labels) and
several attributes (features). The goal of SVM is to produce a
model that can predict the target values of the data instances
in the testing set given the attributes only.

Support Vector Machines are based on the concept of
decision planes that define decision boundaries. A decision
plane is one that separates between a set of objects having
different class memberships, the SVM modeling algorithm
finds an optimal hyper plane with the maximal margin to
separate the two classes. This is illustrated in Fig. 1.

![SVM binary classification](image)

**Figure 1. SVM binary classification**

Given a training set of feature-labeled pairs \((x_i, y_i), \quad i = 1, 2, \ldots, l\)
represents the features, \(x_i\) a loss instance and \(y_i\) the
label denoting the type of loss. \(y_i\) is either 1 denoting a loss
due to congestion or -1 denoting a loss due to a link error.

SVM can be used to learn a linear classifier

\[
f(x) = w^T x + b
\]  

where \(w\) is a weight vector and \(b\) is a bias. When the training
samples are not linearly separable, the sample is mapped to a
higher dimensional feature space \((\Phi)(x)\):

\[
f(x) = w^T (\Phi)(x) + b
\]

before a linear classification is performed in the new space.

The problem can be solved as the following optimization
problem:

\[
\min \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i
\]

Subject to \(y_i (\Phi)(x_i)(w^T(\Phi)(x_i) + b) \leq 1 - \xi_i\)

\[
\xi_i \geq 0, \quad i = 1, \ldots, l
\]

The equation can be rephrased as

\[
\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha
\]

\[
0 \leq \alpha_i \leq C, \quad i = 1, \ldots, l,
\]

subject to \(y^\top \alpha = 0\)

where \(\xi_i\) is the slack variable that indicates tolerances of
misclassification and \(C > 0\) is a constant that balances
maximizing the margin and minimizing the amount of n.e is
the vector of all ones, \(Q\) is an \(l \times l\) positive semi-definite matrix,
\[ Q_{ij} = \sum_{l=1}^{N} \mathbf{K}(x_{il}, x_{ij}) \] and \[ \mathbf{K}(x_{il}, x_{ij}) = \mathbf{K}(x_{il})^T \mathbf{K}(x_{ij}) \] is the kernel. The decision function is
\[ f(x) = sgn\left( \sum_{l=1}^{N} y_l a_l \mathbf{K}(x_l, x) + b \right) \] (5)

3.2 Description of dataset

The dataset used in this paper is the one generated and used in [10]. The database was generated by simulations with the network simulator ns-2 and contains 35,441 loss instances, 22,426 of which are due to congestion and the rest to link error. The network topologies, the number of wireless links, their place in the topology, the error model and the loss rate were drawn at random.

The parameters measured at the end of a loss event are the inter-arrival times and the relative one way delay. The inter-packet times denotes the arrival time difference between consecutive received packets. The one-way delay, computed by one of the two entities, is the difference between the timestamp of the acknowledgement and the timestamp of the TCP packet, and is actually the real one-way delay minus the difference between the clocks of the sender and the receiver. To make the model independent of the absolute values of these measures, the values were normalized in different ways using the average, the standard deviation, the minimum, and the maximum of the one-way delay and inter-packet time.

The motivation for using this dataset, is for the random generation of the network parameters. This will make the model as applicable as possible under different network conditions. In addition, it provides a standard for the comparative evaluation of our result with that obtained by [10].

3.3 Training process

LIBSVM 3.12 is used to train and optimize the SVM model. LIBSVM is integrated software for support vector classification(C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM). The dataset was randomly divided into two; the training set and the testing set. The training set contains 26,191 loss instances selected at random. The testing set contains the remaining 9250 loss instances. Light Data Agent application was used to put the dataset into the appropriate format.

The RBF kernel was chosen. This is the most appropriate here because the dataset is quite large and contains very few number of attributes compared with the number of instances. Also, RBF kernel nonlinearly maps samples into a higher dimensional space and so it can handle the case where the relation between class labels and attributes is nonlinear.

For selecting the best parameter value, the grid.py in the libsvm-3.12\tools directory was used.

grid.py is a parameter selection tool for C-SVM classification using the RBF (radial basis function) kernel. The training set is divided into five subsets of equal size. Sequentially, one subset is tested using the classifier trained on the remaining four subsets. Thus, each instance of the whole training set is predicted once. The cross-validation accuracy is the percentage of correctly classified data. The optimal parameters obtained using the above process are \( C = 128 \) and \( \gamma = 0.5 \). These parameters are used to conduct a more efficient training and to accurately predict unknown data. After cross validation, the best parameters obtained was used to train the dataset. The model achieved a cross validation accuracy is 96.184%.

3.4 Performance metric

The following metrics were used for the performance evaluation of the model:

True positive (TP): This refers to the group of positive instances that are correctly classified by the algorithm as positive. In this case this refers to the number of congestion losses that are truly identified as congestion.

True negative (TN): This refers to the group of negative instances that are correctly classified by the algorithm as negative. In this case this refers to the number of wireless losses that are correctly classified as wireless losses.

False negative (FN): This refers to the group of positive instances that are wrongly classified by the algorithm as negative. In this case this refers to the number of congestion losses that are misclassified as wireless losses. This is denoted as \( ERR_C \).

False positive (FP): This refers to the group of negative instances that are wrongly classified by the algorithm as positive. In this case this refers to the number of wireless losses misclassified as congestion losses. This is denoted by \( ERR_E \).

Accuracy: This is the number of correctly classified instances over total number of instances. As shown in Figure 4.1, this is
\[ \frac{TP + TN}{P+N} \]

Misclassification Rate: This is the total number of wrongly classified instances over total number of instances.
\[ \frac{FP + FN}{P+N} \]
4. RESULT AND ANALYSIS

The SVM model provides an estimate of the probability of each class $P_C$ and $P_E$ such that $P_C + P_E = 1$ for any given instance of loss. $P_C$ is the probability of classifying the loss as congestion and $P_E$ is the probability of classifying it as a link error.

The class given by the model is a link error if $P_E$ is greater than a threshold $P_{th}$ and Congestion if otherwise. By default, the value of $P_{th}$ is fixed to 0.5 so as to treat each class fairly. However, by changing the threshold, one can easily favour the accuracy of the prediction of one class over the other.

**Accuracy:** The ability of the classifier to correctly identify the category to which each of the loss belongs to given a new situation of a loss event is very important. The higher the accuracy the better the performance of TCP + classifier. The SVM model used here achieved an accuracy of 95.97% on the testing data at a threshold of 0.5. This is quite high and means that the TCP+ classifier has a 0.9597 probability of making the right decision when faced with a new loss situation. The accuracy of the classifier is shown in Fig.2 as the threshold is varied.

**Misclassification rate:** An algorithm that attempts to classify each loss into one of two classes can be judged by its misclassification rate, the fraction of cases which are classified incorrectly. The SVM model here misclassifies only 4.03%. However, since misclassifying a wireless loss as a congestion loss does not have the same impact as the other way around, the performance is judged by examining the two misclassification rates separately.

**Error on congestion losses** ($ERR_C$): This is the % of congestion losses misclassified as wireless losses. The effect of this is that the rate will not be reduced when the network is congested and can lead to congestion collapse if too high. If there is congestion, the sending rate should be decreased to maintain TCP friendliness. At a threshold of 0.5 the classifier misclassifies only 5.52% of congestion losses. Again this is low enough.

**Error on wireless losses** ($ERR_E$): This is the % of wireless loss misclassified as congestion loss. Misclassifying wireless loss as congestion loss does not cause congestion problems for the network, but it will limit the protocol’s ability to improve throughput in the case of wireless network. At a threshold of 0.5 the model misclassifies 26.2% wireless losses.

Fig.3 shows the graph of the two types of errors when plotted against each other with threshold ranging from 0.05 to 0.95. As seen an increase in one leads to a decrease in the other. The curve is very close to the origin showing lower values of errors on both sides. This implies a good model.

**TCP Friendliness**

To maintain TCP-Friendliness, the TCP+classifier should have a throughput belonging to $[1/KB_{tcp}, KB_{tcp}]$ with $K \leq 1.78$ [12] where $B_{tcp}$ is the throughput of TCP following the same path as the TCP+classifier. Let $p$ be the proportion of the packets loss on a network. The TCP+classifier is a normal TCP except that it reacts only to a proportion $p (1- ERR_C)$ of packets instead of $p$ as in the case of a normal TCP. This means TCP+classifier will only reduce its congestion rate for a fraction $1- ERR_C$ of $p$. Therefore it can be seen that to maintain TCP friendliness, the misclassification of congestion should not be too high. If the misclassification of congestion remains close to zero, the classifier will be TCP friendly.
However from figure 4.2, it can be seen that the lower the value of $ERR_C$, the higher $ERR_L$. If $ERR_L$ goes too high, the gain in terms of wireless link will be compromised. How low should $ERR_C$ be without losing TCP Friendliness and at the same time maintaining an optimal throughput on wireless link?

[13] proved that if $ERR_C$ is kept below 18%. The TCP+classifier will maintain both TCP friendliness and an optimal throughput on wireless link. The value of $ERR_C$ obtained at a threshold of 0.5 is much lower than 18%.

**Evaluation of the optimum threshold**

Fig.4 below shows the change in the $ERR_C$ as the threshold is varied. As explained above, to maintain TCP friendliness $ERR_C$ should be kept below 18%. This means that the threshold should always be lower than 0.92.

![Graph of error on congestion against threshold](image)

**Figure 4.** Graph of error on congestion against threshold

**Throughput:** In order to have an optimal throughput, the correct classification of a loss as due to a link error ($T_{wireless}$) must be high.

Using the equation proposed by [14]. $r$ is the transmission rate and $I$ is the loss event rate,(the equation is already defined in 1.2). From this, it can be seen that a misclassification of wireless loss as congestion loss decreases the throughput by a factor of $l$. The current TCP misclassifies every loss due to a link error and thus reacts to all the losses $l$ decreasing the sending rate for each.

For a threshold of 0.5, the SVM model correctly classifies an approximate of 73 out of every 100 losses due to a link error. This is good enough and as seen in Fig.4, the accuracy increases with the threshold. At a threshold of 0.85 the two curves intersect. This threshold is lower than the threshold that is required to maintain TCP friendliness (0.92) and also provides a wireless accuracy of 87.15% which is high enough for optimum throughput.

In [10] several machine learning algorithms were trained and tested on the same dataset. It was found that decision tree boosting achieved the highest accuracy of 93.66. SVM has a higher accuracy than this(95.97%).

**5. CONCLUSION**

SVM is employed to differentiate between the two types of losses that occur on hybrid network. The SVM model obtained showed a great capability in classifying correctly a new loss instance. The result showed that when incorporated into TCP it improved the throughput of TCP substantially on wired/wireless networks without compromising TCP-friendliness.

**REFERENCES**


