COMPETITIVE AND ONLINE PIECEWISE LINEAR CLASSIFICATION

Huseyin Ozkan[1]*, Mehmet A. Donmez*[2]*, Ozgun S. Pelvan*[3]*, Arda Akman*[3]*, Suleyman S. Kozat*[1]

[1]Bilkent University, Electrical and Electronics Engineering Department, Ankara, Turkey
[2]Koc University, Electrical and Electronics Engineering Department, Istanbul, Turkey
[3]Turk Telekom Group R&D, Ankara, Turkey

ABSTRACT

In this paper, we study the binary classification problem in machine learning and introduce a novel classification algorithm based on the "Context Tree Weighting Method". The introduced algorithm incrementally learns a classification model through sequential updates in the course of a given data stream, i.e., each data point is processed only once and forgotten after the classifier is updated, and asymptotically achieves the performance of the best piecewise linear classifiers defined by the "context tree". Since the computational complexity is only linear in the depth of the context tree, our algorithm is highly scalable and appropriate for real time processing. We present experimental results on several benchmark data sets and demonstrate that our method provides significant computational improvement both in the test $(5 \sim 35\times)$ and training phases $(40 \sim 1000\times)$, while achieving high classification accuracy in comparison to the SVM with RBF kernel.

Index Terms— Online; Competitive; Classification; Piecewise linear; Context tree; LDA

1. INTRODUCTION

Classification is one of the most important tasks in machine learning. For this task, when given a set of training data of two classes, a classifier is trained via an algorithm with respect to a predefined criteria, e.g., a regularized empirical error minimization [1]. Among such algorithms, perhaps the most popular one is the nonlinear Support Vector Machines (SVM) due to its power of modeling any nonlinear separation between two classes with efficient generalization [2]. However, both the training and test phases of nonlinear SVMs scale poorly with the size of the training data [3], whereas linear machines are computation-wise significantly less complex [3, 4]. When given a nonlinear classification task, instead of solving for the entire problem, we divide it into smaller linear problems, each of which is solved via the Linear Discriminant Analysis (LDA) [5]. Here, any linear machine, e.g., linear SVM, can also be used, however, we choose LDA for its straightforward online extensions [5]. To this end, we introduce a novel online classification algorithm that approximates the nonlinear separations via piecewise linear boundaries. Since our algorithm operates sequentially through online updates, i.e., every data point is processed only once, it operates significantly faster than nonlinear SVM in the training phase. Moreover, we prove that our algorithm sequentially and asymptotically achieves, in the "soft sense", the batch performance of the best classifier in a certain class of piecewise-linear algorithms C_l that we also introduce. Hence, our algorithm is competitive. According to our experiments on several benchmark data sets [6], we obtain computational improvement $5 \sim 35\times$ in the test phase, and $40 \sim 1000\times$ in the training phase with comparable classification accuracy to SVM with RBF kernel. Furthermore, when the training size is in the order of ten thousands in the case of applications such as road sign detection, or human detection [3], the computational improvement is naturally even higher.

Our work is based on the concept of a "context tree", which has been applied with great success in various different fields ranging from compression to machine learning [7, 8, 9]. A context tree is basically an efficient way of representing a particular set of partitions of the observation space and assigning a "context" to a specific observation through weighting on those partitions [7]. In this paper, we consider a data instance x_t as the context of its label y_t, and based on the context, our algorithm predicts the label incorporating the Context Tree Weighting Method (CTW). The aforementioned "competition class" C_l is defined as the set of algorithms, each of which operates on a different partition defined by the context tree and applies LDA on every region of the corresponding partition independently. The CTW is used in the context of time series prediction in [8], where the predictions are based on regression analysis, which is repeatedly carried out on the past data at every time t. However, we study the problem of binary classification, that is based on discriminative analysis through LDA. Moreover, our algorithm works incrementally, i.e., each data point (context) is processed only once. In [9], context trees are used as decision trees, where the observation x_t is assumed to be binary and no coding schema is provided for real observations. In this study [9], since the prediction of y_t is solely based on the relative frequencies of the labels of the previous observations sharing the same context with x_t, the proposed algorithm potentially requires relatively high depth context trees for a satisfactory discrimination. On the contrary, we exploit the possible local linearities of the separation in the data by using LDA. Moreover, our algorithm can work with real observations, for which we explicitly provide a coding schema.

After we provide the problem definition in Section 2, we introduce a baseline classifier in Section 3. Using this baseline classifier, we design our online competitive classification algorithm in Section 4. We demonstrate the performance of our algorithm on several benchmark data sets in Section 5.

2. PROBLEM DEFINITION

Suppose we have a stream of i.i.d. samples $D_1^T = \{d_1, ..., d_T\}$, where d_t is the pair of the data point at time t and the corresponding label such that $d_t = (x_t, y_t)$, $x_t \in [-M, M]^d$ and $y_t \in \{-1, +1\}$. A classifier operating in the domain of the streamed data points is defined to be a function $f : I \subset [-M, M]^d \to \{-1, +1\}$. For training a classifier on this data stream, i.e., selecting a classifier with respect to a

pre-defined criteria, e.g., the error count, a training algorithm $\mathcal{F}$ is defined as $\mathcal{F}: \mathcal{I} \subset ([-M,M]^d \times \{-1,1\})^t \rightarrow \{f_j\}$, where f_j is a classifier and j is from an uncountable index set. For instance, $f = \mathcal{F}(\emptyset)$ provides the initial guess for the label of x_1. Given a training algorithm of this form, we also define a sequential loss function $l(\mathcal{F}; D_1^t) = \sum_{i=1}^{t} (f_{i-1}(x_i) - y_i)^2$, where $f_{i-1} = \mathcal{F}(D_1^{i-1})$. Note that at every time instant i, the additional error is computed for the classifier trained on only the past data D_1^{i-1}, i.e., x_i is a test point for the algorithm since it is not included in training. In this sense, we obtain a fair performance metric. Using this metric, we study the following problem: Given a class of N algorithms, $C = \{\mathcal{F}_1, ..., \mathcal{F}_N\}$, we seek for an algorithm such that it performs asymptotically as well as the best one in C, i.e.,

$$\frac{l(\mathcal{A}; D_1^t)}{t} \leq \frac{l(\mathcal{F}; D_1^t)}{t} + \frac{O(1)}{t}, \forall \mathcal{F} \in C, \qquad (1)$$

in a strong sense without any stochastic assumptions on the observations. In this paper, we define a competition class C_l of algorithms via the context trees [7] and then design a competitive classification algorithm incorporating the CTW [7] that achieves the bound in (1). To this end, in the following section, we introduce a baseline classifier, from which the competition class and our competitive algorithm are derived.

3. PIECEWISE LDA

In this section, we introduce a classifier, named "Piecewise LDA", which operates on a given partition of the input domain. Based on this, a certain collection of such partitions will be specified and hence, a collection of Piecewise LDA's will be obtained as our competition class C_l in the next section. Moreover, since Piecewise LDA is the main operational block in design of our competitive algorithm, this section also provides the intuition behind our work. Given a partition $\mathcal{P} = \{R_1, ..., R_{n_P}\}$ such that $\bigcup R_i = [-M,M]^d$, Piecewise LDA, denoted by f, classifies a streamed data point x_t as

$$f(x_t) = \text{sign}(w_j^T x_t + b_j) \text{ if } x_t \in R_j \text{ and } n_j^+, n_j^- \neq 0, \quad (2)$$
$$f(x_t) = 1, \text{ if } x_t \in R_j \text{ and } n_j^- = 0,$$
$$f(x_t) = -1, \text{ if } x_t \in R_j, n_j^+ = 0, n_j^- \neq 0,$$

where n_j^+, n_j^- are the number of points of D_1^{t-1} in region R_j labeled as 1 and -1, respectively. Also, (w_j, b_j) is obtained by applying Linear Discriminant Analysis (LDA) [5] independently in every region $R_j \in \mathcal{P}$ at time $t-1$ based on the past data D_1^{t-1}. We assume equal and identity class covariances in every region R_j, which leads to $w_j = \mu_+ - \mu_-$, where μ_+ and μ_- are the class mean vectors. Note that whenever an observation $x_t \in R_j$ is streamed, it is processed only once by updating μ_+, μ_- and w_j to form the classification at $t+1$ as $f(x_{t+1})$, i.e., $\hat{\mu}_+ = \frac{n_j^+ \mu_+ + x_t}{n_j^+ + 1}$ if $y_t = 1$, and similarly for $\hat{\mu}_-$. Due to this update process, Piecewise LDA can also be seen as an online algorithm operating on a given data stream. Here, the assumption of equal and identity class covariances can easily be dropped and a more sophisticated LDA can be applied, which would also have straightforward online extensions [5].

We emphasize that Piecewise LDA is defined based on a given partition $\mathcal{P}$. Hence, for every possible partition of the input domain, one can obtain a different classifier as defined in (2). As an example, if we take $d = 2$, $M = 4$, $T = 500$ (250 for each class), and use the 4-Quadrant Partition, i.e., $\mathcal{P} = \{([-4,0] \times [-4,0]), ([-4,0] \times [0,4]), ([0,4] \times [-4,0]), ([0,4] \times [0,4])\}$, to train the corresponding classifier as explained, we obtain piecewise linear decision boundaries (hence, nonlinear) for a set of data shown in Fig. 1, which can be seen as the piecewise linear approximation of the nonlinear separation in the data. In this example, Piecewise LDA performs nearly as well as the SVM with RBF kernel (sigma parameter of RBF kernel is set to 1) in terms of the classification accuracy. On the contrary, SVM requires 432 dot products in the test phase, whereas Piecewise LDA requires only at most 4 dot products. This corresponds to $\sim$ 105X speed-up in the test phase.

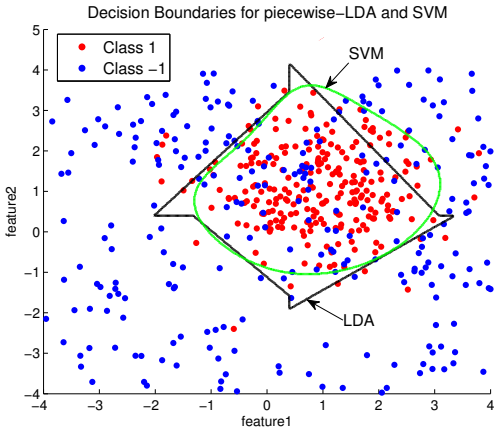

Fig. 1. Piecewise LDA vs SVM. See the text for details.

Unfortunately, in the realistic scenarios, the optimal partition of the input space to train Piecewise LDA is unknown. However, in the following section, we define the competition class C_l by specifying those partitions (and hence, algorithms) via the context trees [7]. Then we design our online competitive algorithm incorporating CTW [7] that competes against C_l, i.e., achieves the bound in (1), meaning that selects the best partition in C_l in the course of the data streaming.

4. ONLINE PIECEWISE LDA VIA CTW

In this section we introduce our competition class C_l and a novel algorithm, named "Piecewise LDA via CTW" and denoted by $\mathcal{A}$, based on the concept of a "context tree". For ease of exposition, we consider a stream of 2-dimensional data points, i.e., $d = 2$. However, the d-dimensional extension is straightforward. A K-depth context tree is basically a set of nodes, each of which corresponds to a region in $[-M,M]^2$ as shown in Fig. 2. The "root" node corresponds to the region $[-M,M]^2$ and is at the 0-depth, whereas the leaf nodes are at the K-depth. For any internal (non-leaf) node ν with depth k, i.e., $k < K$, the corresponding region is split vertically into two equal halves if k is even; and horizontally otherwise. A split at node ν generates two sub-regions, which are assigned to the nodes ν_l and ν_r that are the children of the node ν. Note here that the regions at the leaves of any pruned K-depth context tree gives a partition of the space $[-M,M]^2$, where pruning can be defined as removing all the subtrees rooted from some (or none) of the internal nodes of a given tree. For an example see Fig. 2. Then, we specify the class C_l as the set of algorithms $\{\mathcal{F}_i\}$ as defined in (2), each of which sequen-

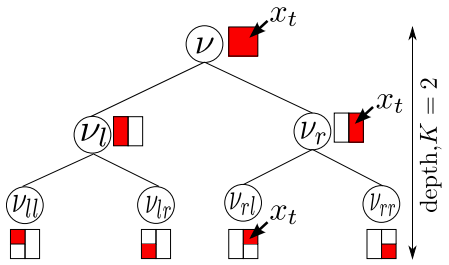

Fig. 2. An example context tree. All possible partitions defined by this tree is listed as: $\mathcal{P}_1 = \{\nu\}, \mathcal{P}_2 = \{\nu_l, \nu_r\}, \mathcal{P}_3 = \{\nu_l, \nu_{rr}, \nu_{rl}\}, \mathcal{P}_4 = \{\nu_r, \nu_{lr}, \nu_{ll}\}, \mathcal{P}_5 = \{\nu_{ll}, \nu_{lr}, \nu_{rr}, \nu_{rl}\}$.

tially operates on a different partition $\mathcal{P}_i$ given by the pruned versions of a K-depth context tree.

To describe Piecewise LDA via CTW, we also define an algorithm $\mathcal{G}$ applying LDA as defined in (2), except that it does not operate on a partition of the input space $[-M, M]$, but on regions of the context tree. Namely, given the data stream D_1^t, the algorithm $\mathcal{G}$ finds the data, which fall in the region of the node ν and then applies LDA to train a classifier. Hence, for every node ν, we have the associated loss $l\left(\mathcal{G}; D_1^t(\nu)\right)$, where $D_1^t(\nu) = \{(x_i, y_i) : x_i \text{ is included in the region of node } \nu\}$. Next, we introduce our competitive online classification algorithm Piecewise LDA via CTW as follows: Given an arbitrary data stream D_1^{t-1}, let the observation x_t fall in the regions of nodes $\{\nu_1, \nu_2, ..., \nu_{K+1}\}$, where the subscript i indicates the depth and ν_1 is the root node. Then the algorithm $\mathcal{A}$ is defined as a linear combination of the Piecewise LDA's operating at v_i's such that $\mathcal{A}(D_1^{t-1}) = \sum_{i=1}^{K+1} \mu_i f_i$, where $f_i = \mathcal{G}\left(D_1^{t-1}(\nu_i)\right)$ and $\sum_{i=1}^{K+1} \mu_i = 1$. The following theorem defines Piecewise LDA via CTW by specifying a certain set of weights μ_i in definition of algorithm $\mathcal{A}$ and states that it asymptotically performs as well as the best algorithm in the competition class C_l.

Theorem For the algorithm $\mathcal{A}$, we can find a certain set of weights μ_i such that the algorithm $\mathcal{A}$ asymptotically performs as well as the best $\mathcal{F} \in C_l$, i.e.,

$$\frac{l(\mathcal{A}; D_1^t)}{t} \leq \frac{l(\mathcal{F}; D_1^t)}{t} + \frac{O(1)}{t}, \forall \mathcal{F} \in C_l.$$

Proof: We first emphasize that the algorithm $\mathcal{A}$ is a linear combination of Piecewise LDA's at nodes of varying depths in the context tree. In order to have weighting over these Piecewise LDA's depending on the depths, we define the "weighted probability" for each node ν as: $P_w^t(\nu) = P(D_1^t(\nu)|\mathcal{G})$ if ν is a leaf node in the context tree; $P_w^t(\nu) = \frac{1}{2}P(D_1^t(\nu)|\mathcal{G}) + \frac{1}{2}P_w^t(\nu_r)P_w^t(\nu_l)$ otherwise. Here, $P(D_1^t(\nu)|\mathcal{G})$ is the likelihood of the algorithm $\mathcal{G}$ defined as $P(D_1^t(\nu)|\mathcal{G}) = \exp\left(\frac{-1}{2h} l\left(\mathcal{G}; D_1^t(\nu)\right)\right)$. Based on this definition, it is straightforward to show that $P_w^t(\nu_1) = \sum_{\forall \mathcal{F}_i \in C_l} P(\mathcal{F}_i) P(D_1^t|\mathcal{F}_i)$, where ν_1 is the root node, $P(\mathcal{F}_i) = 2^{-\Gamma_K(\mathcal{F}_i)}$ is the prior probability for an algorithm $\mathcal{F}_i \in C_l$ with the constant $\Gamma_K(\mathcal{F}_i)$, where $\sum_{\forall \mathcal{F}_i} P(\mathcal{F}_i) = 1$ (cf. *Lemma 2* in [7]). Moreover, we have an efficient recursive procedure to compute $P_w^t(\nu_1)$ at every time t, as a new data point is streamed. Let us consider the example shown in Fig. 2. At time t, the streamed data point x_t is included in regions of $K+1 = 3$ nodes and hence, $P(D_1^{t-1}(\nu)|\mathcal{G})$, at only those nodes, need be updated to compute $P_w^t(\nu_1)$ as follows:

$$P_w^t(\nu_1) = \gamma_1 P(D_1^t(\nu_1)|\mathcal{G}) + \gamma_2 P(D_1^t(\nu_r)|\mathcal{G}) + \gamma_3 P(D_1^t(\nu_{rl})|\mathcal{G}),$$

where $\gamma_1 = \frac{1}{2}$, $\gamma_2 = \frac{\gamma_1}{2} P_w^t(\nu_l)$ and $\gamma_3 = \gamma_2 P_w^t(\nu_l) P_w^t(\nu_{rr})$. Then, for a general K-depth context tree, one can obtain:

$$P_w^t(\nu_1) = \sum_{i=1}^{K+1} \gamma_i P(D_1^{t-1}(\nu_i)|\mathcal{G}) \exp\left(\frac{-1}{2h}(f_i(x_t) - y_t)^2\right),$$

where $f_i = \mathcal{G}\left(D_1^{t-1}(\nu_i)\right)$; $\gamma_i = \frac{\gamma_{i-1}}{2} P_w^t(\bar{\nu}_i)$ for $2 \leq i \leq K$, $\gamma_0 = \frac{1}{2}$, $\gamma_{K+1} = \gamma_K P_w^t(\bar{\nu}_i)$; $\bar{\nu}_i$ is the sibling node of ν_i, and ν_1 is the root. Let us consider

$$\frac{P_w^t(\nu_1)}{P_w^{t-1}(\nu_1)} = \frac{\sum_{i=1}^{K+1} \gamma_i P(D_1^{t-1}(\nu_i)|\mathcal{G}) \exp\left(\frac{-1}{2h}(f_i(x_t) - y_t)^2\right)}{\sum_{i=1}^{K+1} \gamma_i P(D_1^{t-1}(\nu_i)|\mathcal{G})}.$$

If we let $\mu_i = \frac{\gamma_i P(D_1^{t-1}(\nu_i)|\mathcal{G})}{\sum_{i=1}^{K+1} \gamma_i P(D_1^{t-1}(\nu_i)|\mathcal{G})}$, then $\sum_i \mu_i = 1$ and

$$\frac{P_w^t(\nu_1)}{P_w^{t-1}(\nu_1)} = \sum_{i=1}^{K+1} \mu_i \exp\left(\frac{-1}{2h}(f_i(x_t) - y_t)^2\right).$$

Then, since $\exp\left(\frac{-1}{2h}(f_i(x_t) - y_t)^2\right)$ is concave as a function of $f_i(x_t)$ for all values $(f_i(x_t) - y_t)^2 < h$, we apply Jensen's inequality and obtain the following inequality:

$$\exp\left(\frac{-1}{2h}(y_t - \sum_{i=1}^{K+1} \mu_i f_i(x_t))^2\right) \geq \frac{P_w^t(\nu_1)}{P_w^{t-1}(\nu_1)}, \text{ where } h > 4.$$

Recall that $f_i = \mathcal{G}\left(D_1^{t-1}(\nu_i)\right)$ (f_i depends on $t-1$). Based on the definition of algorithm $\mathcal{A}$, if we let $\mathcal{A}(D_1^{t-1}) = \alpha_{t-1}$ then, since $P_w^t(\nu_1) = \sum_{\forall \mathcal{F}_i \in C_l} 2^{-\Gamma_K(\mathcal{F}_i)} P(D_1^t|\mathcal{F}_i)$,

$$\exp\left(\sum_{j=1}^{t} \frac{-(y_j - \alpha_{j-1}(x_j))^2}{2h}\right) = \exp\left(\frac{-1}{2h} l(\mathcal{A}; D_1^{t-1})\right)$$
$$= P(D_1^t|\mathcal{A}) \geq P_w^t(\nu_1) \geq 2^{-\Gamma_K(\mathcal{F}_i)} P(D_1^t|\mathcal{F}_i) \quad (3)$$

is obtained. Taking the logarithm of both sides of the inequality in (3), we can complete the proof:

$$\frac{l(\mathcal{A}; D_1^t)}{t} \leq \frac{l(\mathcal{F}; D_1^t)}{t} + \frac{2h \ln 2 \Gamma_K(\mathcal{F})}{t}, \forall \mathcal{F} \in C_l. \blacksquare$$

Note that the algorithm $\mathcal{A}$, Piecewise LDA via CTW, yields "soft" decisions for a given test point, i.e., $g(x_t) \in \mathbb{R}$, $g = \mathcal{A}(D_1^{t-1})$. Hence, we proved that in the soft sense, Piecewise LDA via CTW performs asymptotically as well as even the batch performance of the best algorithm in class C_l. Then, we simply obtain the classification based on the soft decisions by $\text{sign}(g)$. In the following section, we present the experimental evaluation of $\text{sign}(g)$. In these experiments, the algorithm $\mathcal{A}$ is trained sequentially using a training set, in which our theorem demonstrates the fitting capability of $\mathcal{A}$ w.r.t. C_l. On a separate test set, the training is turned off, and the classification performance along with the test phase complexity is compared with SVM with the RBF kernel. Nevertheless, whenever a new data point is streamed along with the corresponding label, Piecewise LDA via CTW is capable of keep training in an online and computationally efficient manner as shown in Section 5.

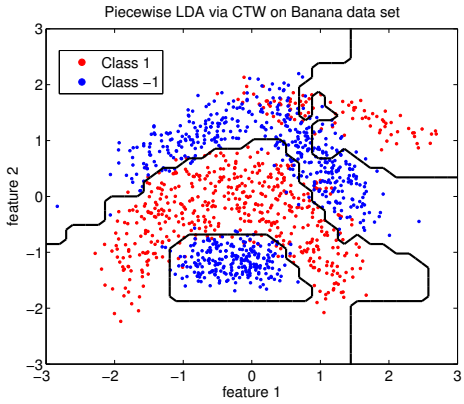

Fig. 3. Piecewise linear approximation of the nonlinear separation in Banana data set. See text for details.

5. EXPERIMENTAL EVALUATION

In this section, the performance of our online classification algorithm is demonstrated on four different benchmark data sets, which are commonly used in machine learning applications [6]. Among these data sets, the followings are chosen for their relatively large number of data points in training: (1) Banana data set, which is a synthetic, $d = 2$ dimensional data set with a training set of $N_{\text{train}} = 1000$ points, and a test set of $N_{\text{test}} = 4300$ points; (2) Image data set, which is a $d = 18$ dimensional real data set with $N_{\text{train}} = 1300$, and $N_{\text{test}} = 1010$. We also have two other real data sets with relatively small N_{train}: (3) Waveform data set, which is $d = 21$ dimensional with $N_{\text{train}} = 400$, and $N_{\text{test}} = 4600$; and finally, (4) Titanic data set, which is $d = 3$ dimensional with $N_{\text{train}} = 150$, and $N_{\text{test}} = 2051$. For each of these data sets, our online classification algorithm is sequentially trained in the training set, where the context tree depth parameter D is optimized through cross validations (the parameter h is fixed, $h = 8$). For a comparison, we also train SVM with RBF, $k(x, y) = \exp(\frac{-\|x-y\|^2}{2\sigma^2})$, with the kernel parameter σ and the SVM fitting parameter C as provided in [6]. Then, we calculate the empirical classification error rates on the test set. Here, we point out that we used LIBSVM [10] for the implementation of the SVM, which is an efficient open source SVM library implemented in C++.

In these experiments, Piecewise LDA via CTW is observed to perform nearly as well as the SVM in terms of the classification accuracy. This indicates that our algorithm is able to successfully approximate the nonlinear separations in the data sets through piecewise linear boundaries. For an example, when our algorithm is trained with $D = 10$ for the Banana data set, the piecewise linear classification boundaries that we obtain are shown in Fig. 3. In this case, the classification error rate is observed to be $\%14$, which is only $\%4$ worse than that of the SVM (trained with the parameters as in [6]). On the other hand, since the context tree depth parameter is used as $D = 10$, only 10 dot products for classification of a new test point is required, whereas SVM requires as many dot products as the number of support vectors $nSV = 228$. Note that the decision function for SVM [2] is given as $f(x) = \sum_{i=1}^{nSV} \alpha_i k(x, x_i) + b$. This corresponds to $\sim 25\times$ computational reduction achieved by our algorithm in the test phase. As for the training, SVM takes ~ 121 milliseconds (ms) processing time with LIBSVM [10]. Here, we point out that SVM is not a sequential algorithm, i.e., whenever the classifier needs update, the algorithm is re-trained on the entire training set. Thus, in order to have a fair comparison, this re-training time for the SVM is compared in Table 1 to the update time of our algorithm for the last point in the training set, i.e., the update time when the N_{train}'th data point is streamed. This update takes 0.142 ms, which indicates $\sim 1000\times$ speed up in the training phase of the Banana data set. Corresponding findings for each data set are summarized in Table 1.

Table 1. Classification error rates, training times and test phase complexity for each data set and each algorithm. Piecewise LDA via CTW performs nearly as well as SVM, whereas the training and test phase complexity is significantly reduced. See text for details.

Data Sets	Piecewise LDA via CTW			SVM with RBF		
	Error	Depth	Training	Error	nSV	Training
Banana	0.140	10	0.14 ms	0.101	228	121 ms
Image	0.060	20	0.40 ms	0.030	153	168 ms
Waveform	0.141	6	0.11 ms	0.118	200	27.0 ms
Titanic	0.230	10	0.11 ms	0.213	65	4.19 ms

In particular, for the Waveform data set, we have notably high degree of sparseness due to the high dimensionality $d = 21$, when compared to the small number of training points $N_{\text{train}} = 400$. For this reason, relatively simpler models are found to be more appropriate such as $D = 6$. Even in this case of high sparseness, Piecewise LDA via CTW still performs comparable to SVM with significant computational reduction both in training ($\sim 250\times$ speed-up in training) and test phases ($\sim 35\times$ speed-up in test). Nevertheless, the computational gain in the training phase is much smaller than that in the case of Banana data set. This is related to the size of training. In general, when we have larger size of training sets, the computational gain is also greater w.r.t. SVM with RBF due to its poor scaling capability with the size of training data [3]. Hence, we would expect to obtain drastically higher computational gains in case of applications such as Road Sign Detection or Human Detection [3], where the training sets are usually in the order of ten thousands. On the other hand, we obtain less computational gain in case of the Titanic data set, where $N_{\text{train}} = 150$. In these experiments, our algorithm is shown to be appropriate for real time processing. Furthermore, since the computational complexity of our algorithm is directly controllable by the context tree depth parameter D, it can be computationally further optimized, if desired.

6. CONCLUSIONS

In this paper, we proposed a novel, online classification algorithm, which provides significant computational improvement corresponding to $5 \sim 35\times$ in the test phase and $40 \sim 1000\times$ in the training phase with comparable classification accuracy to SVM with RBF kernel in our experiments. The proposed algorithm operates on a given data stream through sequential updates and approximates complex nonlinear separations by piecewise linear decision boundaries with computational complexity that is only linear in depth of the context tree. Hence, our method is scalable and appropriate for real time processing. In addition, we proved that our algorithm is sequentially "competitive", i.e., it asymptotically achieves the batch performance of the best classifier in the class of algorithms C_l that we also introduced.

7. REFERENCES

[1] O. Bousquet, S. Boucheron, and G. Lugosi, "Introduction to statistical learning theory," *Advanced Lectures on Machine Learning*, pp. 169 – 207, 2004.

[2] C.J.C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[3] F. Porikli and H. Ozkan, "Data driven frequency mapping for computationally scalable object detection," in *AVSS*, 2011, pp. 30 –35.

[4] T. Joachims, "Training linear svms in linear time," in *the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 217–226.

[5] K. Hiraoka, M. Hamahira, K. Hidai, H. Mizoguchi, T. Mishima, and S. Yoshizawa, "Fast algorithm for online linear discriminant analysis," in *Proceedings of ITC*, 2000, pp. 274 – 277.

[6] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.R. Mullers, "Fisher discriminant analysis with kernels," in *Neural Networks for Signal Proces.*, 1999, pp. 41 –48.

[7] F.M.J. Willems, Y.M. Shtarkov, and T.J. Tjalkens, "The context-tree weighting method: basic properties," *IEEE Transac. on Inform. Theory*, vol. 41, pp. 653 –664, 1995.

[8] S. S. Kozat and Zeitler G. C. Singer, A. C., "Universal piecewise linear prediction via context trees," *IEEE Transactions on Signal Processing*, 2006.

[9] D. P. Helmbold and R. E. Schapire, "Predicting nearly as well as the best pruning of a decision tree," *Mach. Learn.*, vol. 27, pp. 51–68, 1997.

[10] Chih-Chung Chang and Chih-Jen Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27:1–27:27, 2011.