# ADAPTIVE MARGIN SLACK MINIMIZATION IN RKHS FOR CLASSIFICATION

*Yinan Yu, Konstantinos I. Diamantaras, Senior Member, Tomas McKelvey, Senior Member, S.Y. Kung, Fellow*

Chalmers University of Technology, Gothenburg, Sweden
TEI of Thessaloniki, Thessaloniki, Greece
Princeton University, NJ, USA

## ABSTRACT

In this paper, we design a novel regularized empirical risk minimization technique for classification called Adaptive Margin Slack Minimization (AMSM). The proposed method is based on minimizing a regularized upper bound of the misclassification error. Compared to the cost function of the classical L2-SVM, AMSM can be interpreted as minimizing a tighter bound with some additional flexibilities regarding the choice of marginal hyperplane. A hyperparameter-free adaptive algorithm is presented for finding a solution to the proposed risk function. Numerical results shows that AMSM outperforms L2-SVM on the tested standard datasets.

***Index Terms*—** L2-SVM, Structural Risk Minimization, Reproducing Kernel Hilbert Space, Adaptive Margin

## 1. INTRODUCTION

Learning machines in the Reproducing Kernel Hilbert Space (RKHS) are some of the most popular nonlinear models applied to classification problems due to their success on a wide range of applications [20, 6]. They have been proposed in the context of the dual for solving Support Vector Machine (SVM) [22, 21] problems. Briefly speaking, in a learning model, when all data vectors appear in the formulation as pairs of inner products, kernel tricks can be applied to avoid computations in the possibly infinite-dimensional feature space. Solving the SVM problem in the dual becomes the standard approach, because the kernel trick can be naturally applied. However, it has been revealed that it is possible to train SVMs in the primal with the same performance [17]. In fact, it gives even better results when it comes to deriving approximate solutions. Furthermore, the training process in the primal mostly involves least squares and simple implementations, which is preferred in many scenarios.

In this paper, we develop a new regularized empirical risk minimization learning model, where the objective function is a regularized upper bound of the misclassification error. We show that such formulations can be interpreted as an extension to the L2-SVM by minimizing a tighter error bound with better flexibility. The corresponding algorithm solving this model can be interpreted as the training process for L2-SVM in the primal with data selection using an adaptive margin.

This paper is organized as follows. First, we motivate the new learning model called Adaptive Margin Slack Minimization (AMSM) in Sec. 2 by introducing an upper bound of the misclassification error. We show that AMSM is viewed as an extension to L2-SVM, since the objective function of L2-SVM can be interpreted as the minimization of a looser bound with less flexibility compared to AMSM. In Sec. 3, we develop an algorithm to solve the proposed learning model and numerical results are shown in Sec. 5.

## 2. BACKGROUND

### 2.1. Notation

Given the training set $\{(\mathbf{x}_i, t_i) \mid \mathbf{x}_i \in \mathbb{R}^p, t_i \in \{1 \cdots C\}, \forall i \in \{1 \cdots N\}\}$ with $\mathbf{x}_i$ being the data vector and $t_i$ the corresponding label information, the task is to find a function $f(\mathbf{x})$ that maps an unseen testing data vector $\mathbf{x}$ to its label $t$ as accurate as possible. In binary classification problems, linear methods attempt to parameterize the function using $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ with the classification rule $\hat{t} = \begin{cases} 1, & f(\mathbf{x}) \geq 0 \\ 2, & f(\mathbf{x}) < 0 \end{cases}$, where $\mathbf{w} \in \mathbb{R}^p$ is called the solution vector and $b \in \mathbb{R}$ the bias. Although enjoying the benefit of computational simplicity, linear models are not often adequate. That gives rise to the popularity of nonlinear models, such as kernel techniques, where we are given a(n) (implicit) nonlinear mapping $\varphi : \mathbf{x} \to \boldsymbol{\varphi}$, such that the dimensionality of $\boldsymbol{\varphi}$ is typically much greater than $p$. By using such transformations, the classification boundary becomes nonlinear and hence in general results in a better class separability power. The nonlinear boundary corresponds to a linear hyperplane in the new high dimensional feature space, called the Reproducing Kernel Hilbert space (RKHS). The training data is then denoted as $\mathcal{D} = \{(\boldsymbol{\varphi}_i, t_i)\}_{\forall i}$. From the Representer Theorem [6], the vector $\mathbf{w}$ can be written as a linear combination of all the training vectors in RKHS, i.e. $\exists \alpha_i$, such that $\mathbf{w} = \sum_{\boldsymbol{\varphi}_i \in \mathcal{D}} \alpha_i \boldsymbol{\varphi}_i, \forall i \in \{1, \cdots N\}$. However, one drawback of such learning models is that the computational complexity is at least $\mathcal{O}(N^2)$ due to the fact that we need to compute and store a gram matrix $\mathbf{K} = \boldsymbol{\Phi}_{\mathcal{D}}^T \boldsymbol{\Phi}_{\mathcal{D}}$, where $\boldsymbol{\Phi}_{\mathcal{D}} = [\boldsymbol{\varphi}_1 \cdots \boldsymbol{\varphi}_N]$, $\forall \boldsymbol{\varphi}_i \in \mathcal{D}$. To resolve such issues, kernel approximation techniques [16, 9, 10, 19, 18, 12, 11, 8, 5, 4, 2] are employed, a matrix $\boldsymbol{\Phi}_G$ with orthonormal columns is generated, such that $\|\boldsymbol{\Phi}_{\mathcal{D}}^T \boldsymbol{\Phi}_G \boldsymbol{\Phi}_G^T \boldsymbol{\Phi}_{\mathcal{D}} - \mathbf{K}\|$ is minimized for some predefined norm $\|\cdot\|$. In our paper, we apply a kernel approximation technique instead of using the whole training set to evaluate the kernel matrix for the sake of scalability.

To simplify the presentation, we limit our discussions to the binary classification problem. Extensions to multiple class can be found in the appendix. The advantage is that the complexity does not increase with the number of classes.

### 2.2. Motivation

A commonly used classifier called the Least Square Support Vector Machine (LS-SVM) [14] has been developed under different names in the literature, such as Proximal Support Vector Machines (PSVM) [13] and Kernel Ridge Regression (KRR) for classification. The main idea of such classifiers is to construct two optimal marginal hyperplanes, where data

points from each class are assigned to the closest marginal hyperplane with low misclassification rate. More precisely, for given training data pairs $(\boldsymbol{\varphi}_i, t_i) \in \mathcal{D}$ with $t_i \in \{-1, +1\}$, define the regularized empirical risk function as follows:

$$\mathcal{R}_{LS-SVM}(\mathbf{w}, b) = \sum_{i \in \mathcal{I}_\mathcal{D}} \xi_i^2 + \rho \|\mathbf{w}\|_2^2 \tag{1}$$

where $\xi$ is the slack variable associated with pattern $\boldsymbol{\varphi}$ defined as:

$$\xi = 1 - \left(\mathbf{w}^T \boldsymbol{\varphi} + b\right) t; \tag{2}$$

Set $\mathcal{I}_\mathcal{D}$ denotes the index set for the elements in the set $\mathcal{D}$; $(\mathbf{w}, b)$ are the design parameters and the scalar $\rho$ is the regularization parameter [1] that penalizes the complexity of the classifier. Given a testing pattern $\boldsymbol{\varphi}$, the classification criterion is $\hat{t} = \text{sign}\left(\mathbf{w}^T \boldsymbol{\varphi} + b\right)$.

As we can see that LS-SVM takes all training data as its support vectors without any selection and hence it might lead to overtraining. This problem gives rise to another important member of the SVM family called the L2-SVM [3]:

$$\mathcal{R}_{L2-SVM}(\mathbf{w}, b) = \sum_{\forall \xi_i > 0} \xi_i^2 + \rho \|\mathbf{w}\|_2^2 \tag{3}$$

By using the squared hinge loss instead of the quadratic loss function as in LS-SVM, L2-SVM is often considered as an extension to LS-SVM when training on the primal.

Clearly, the quantity $|\xi_i|$ measures how far $\boldsymbol{\varphi}_i$ is from the marginal hyperplane. The concept of slack variables is widely used in many classification models, such as soft margin SVM [7], slackmin [15], and as mentioned above, the LS-SVM [14, 13], etc.

By definition, a pattern $\boldsymbol{\varphi}$ is misclassified when $\xi \geq 1$ and hence $\Pr(\xi \geq 1)$ is the ultimate probability that we would like to minimize. Using the conditional Markov inequality, for any $\gamma \in [0, 1]$,

$$\begin{aligned} \Pr(\xi^2 \geq 1) &= \Pr(\xi^2 \geq 1 \mid \xi > \gamma) \Pr(\xi > \gamma) \\ &\leq \mathbb{E}(\xi^2 \mid \xi > \gamma) \Pr(\xi > \gamma) \quad (4) \\ &\leq \mathbb{E}(\xi^2 \mid \xi > \gamma) \quad (5) \end{aligned}$$

The risk function in Eq. (1) can be then modified into to minimizing the upper bound of $\Pr(\xi^2 \geq 1)$. By using either the upper bound in Eq. (4) or (5) with different $\gamma$, we obtain different learning objectives. To illustrate this, first, let us define the active set $\mathcal{S}_\gamma := \{\boldsymbol{\varphi}_i : \xi_i > \gamma,\ \forall i \in \mathcal{I}_\mathcal{D}\}$.

- The regularized risk function based on Eq. (5) ($\gamma = 0$) is then equivalent to the L2-SVM (3):

$$\mathcal{R}_0(\mathbf{w}, b) = \sum_{i \in \mathcal{I}_{\mathcal{S}_0}} \xi_i^2 + \rho \|\mathbf{w}\|_2^2 \tag{6}$$

- The regularized risk function based on Eq. (4) (adaptive $\gamma$), which we call the "Adaptive Margin Slack Minimization" (AMSM):

$$\begin{aligned} \mathcal{R}_\gamma(\mathbf{w}, b) &= \frac{|\mathcal{S}_\gamma|}{N} \sum_{i \in \mathcal{I}_{\mathcal{S}_\gamma}} \xi_i^2 + \|\mathbf{w}\|_2^2 \quad (7) \\ &\propto \sum_{i \in \mathcal{I}_{\mathcal{S}_\gamma}} \xi_i^2 + \frac{N}{|\mathcal{S}_\gamma|} \|\mathbf{w}\|_2^2 = B_\gamma(\mathbf{w}, b) \quad (8) \end{aligned}$$

  where $\frac{|\mathcal{S}_\gamma|}{N}$ is the empirical probability corresponding to $\Pr(\xi > \gamma)$ in Eq. (4).

Compared to Eq. (6), Eq. (8) replaces the hyperparameter $\rho$ by an adaptive regularization parameter $\rho = \frac{N}{|\mathcal{S}_\gamma|}$. Furthermore, Eq. (8) has the following advantages:

1) The regularized risk function for Eq. (7) is associated with a tighter upper bound (Eq. (4)) compared to (5);
2) The adaptive margin $\gamma$ introduces better flexibility;
3) Compared to L2-SVM (Eq. (6)), a lower computational complexity and storage requirement is achieved by reducing the selected data size for $\gamma \in (0, 1]$, i.e. $\mid \mathcal{S}_\gamma \mid < \mid \mathcal{S}_0 \mid$. Typically, it results in a faster convergence rate.

In the next section, a learning algorithm is presented to minimize Eq. (8), where instead of selecting data within a fixed margin $\gamma = 0$, an adaptive margin is updated iteratively. The classifier is trained in the primal using the automatic data selection scheme, which can be naturally extended to a sequential data acquisition and parallel framework.

## 3. ALGORITHM

According to the Representer Theorem [6], the solution vector $\mathbf{w}$ can be written as a linear combination of the training data in the RKHS. With kernel approximation techniques, we have $\mathbf{w} = \boldsymbol{\Phi}_G \mathbf{a}$, given $\boldsymbol{\Phi}_G$ a matrix with $m$ orthonormal columns, whose span approximates the span of the whole training data set in the RKHS, and $\mathbf{a} \in \mathbb{R}^m$. Hence, the parameters to be estimated are $(\mathbf{a}, b)$ in practice and the solution vector is thus defined as $\boldsymbol{\beta} = [\mathbf{a} \quad b]^T$.

Given $\gamma_0 = 0$ and $\boldsymbol{\beta}_0$ computed based on minimizing Eq. (8) for all $i \in \mathcal{I}_\mathcal{D}$ or a subset of $\mathcal{I}_\mathcal{D}$, define the active set at iteration $k$ as follows:

$$\mathcal{S}_{\gamma_k} = \{\boldsymbol{\varphi}_i : \xi_{k,i} > \gamma_k,\ \forall i \in \mathcal{I}_\mathcal{D}\} \tag{9}$$

where

$$\xi_{k,i} = 1 - \left(\mathbf{a}_k^T \boldsymbol{\Phi}_G^T \boldsymbol{\varphi}_i + b_k\right) t_i. \tag{10}$$

At iteration $k$, the algorithm is composed of two parts:

1) Computing $(\mathbf{a}_{k+1}, b_{k+1})$ based on $\mathcal{S}_{\gamma_k}$.
2) Finding $\gamma_{k+1}$ to determine the next active set $\mathcal{S}_{\gamma_{k+1}}$.

### 3.1. Step 1: Computing $\boldsymbol{\beta}_{k+1}$

Given $\mathcal{S}_{\gamma_k}$, the update of $\boldsymbol{\beta}_k \to \boldsymbol{\beta}_{k+1}$ is obtained by minimizing Eq. (8), which is a least-squares problem. More specifically, at each iteration $k$, Eq. (8) gives us the following regularized empirical risk function:

$$B_{\gamma_k}(\mathbf{a}, b) = \sum_{i \in \mathcal{I}_{\mathcal{S}_{\gamma_k}}} \left(1 - (\mathbf{a}^T \mathbf{K}_{Gi} + b) t_i\right)^2 + \rho_k \mathbf{a}^T \mathbf{K}_G \mathbf{a} \tag{11}$$

where $\rho_k = \frac{N}{|\mathcal{S}_{\gamma_k}|}$ is the regularization parameter. The vector $\mathbf{K}_{Gi} = \boldsymbol{\Phi}_G^T \boldsymbol{\varphi}_i$ represents the feature vector and $\mathbf{K}_G = \boldsymbol{\Phi}_G^T \boldsymbol{\Phi}_G$. Generally speaking, for any $\mathcal{S} \subset \mathcal{D}$, we define matrix $\mathbf{K}_{G\mathcal{S}} = \boldsymbol{\Phi}_G^T \boldsymbol{\Phi}_\mathcal{S}$. For kernel methods, these matrices can be evaluated using the kernel trick. It is easy to see that, for

given $\mathcal{S}_{\gamma_k}$, the minimization on $B_{\gamma_k}$ is convex and the solution is given by:

$$\boldsymbol{\beta}_{k+1} = \begin{bmatrix}\mathbf{a}_{k+1}\\ b_{k+1}\end{bmatrix} = \arg\min_{\mathbf{a},b} B_{\gamma_k} = \mathbf{M}_k^{-1}\mathbf{V}_k\mathbf{t}_{\mathcal{S}_{\gamma_k}} \tag{12}$$

where $\mathbf{t}_{\mathcal{S}_{\gamma_k}}$ is a vector contains all $t_i$ for $i \in \mathcal{I}_{\mathcal{S}_{\gamma_k}}$. Let $\mathbf{M}_k = \begin{bmatrix}\mathbf{K}_{G\mathcal{S}_{\gamma_k}}\mathbf{K}_{G\mathcal{S}_{\gamma_k}}^T + \rho_k\mathbf{I} & \mathbf{K}_{G\mathcal{S}_{\gamma_k}}\mathbf{e}\\ \mathbf{e}^T\mathbf{K}_{G\mathcal{S}_{\gamma_k}}^T & |\mathcal{S}_{\gamma_k}|\end{bmatrix}$ and $\mathbf{V}_k = \begin{bmatrix}\mathbf{K}_{G\mathcal{S}_{\gamma_k}}\\ \mathbf{e}^T\end{bmatrix}$, where $\mathbf{e}^T = [1\cdots 1]$.

### 3.2. Step 2: Finding $\gamma_{k+1}$

After $\boldsymbol{\beta}_{k+1}$ has been computed, a threshold $\gamma_{k+1}$ needs to be selected with the following considerations:

I. The larger $\gamma_{k+1}$ is, the more data are eliminated, which results in a lower computational complexity and hence a possibly higher variance in the solution vector.

II. We would like to choose $\gamma_{k+1}$ such that the classification performance is not degraded.

III. Convergence property.

At each iteration, since the empirical risk Eq. (8) is only minimized on the set $\mathcal{S}_{\gamma_k}$, we do not have any guarantee on $\mathcal{D} \setminus \mathcal{S}_{\gamma_k}$. In order to find $\gamma_{k+1}$ such that patterns from set $\mathcal{D} \setminus \mathcal{S}_{\gamma_k}$ are still correctly classified at iteration $k+1$, it suffices to find $\gamma_k$ that satisfies:

$$\xi_{k+1,i} \leq 1, \quad i \in \mathcal{I}_{\mathcal{D}\setminus\mathcal{S}_{\gamma_k}} \tag{13}$$

If such $\gamma_{k+1}$ can be identified, then condition II. is satisfied. Another remaining issue is the convergence of the algorithm (condition III.).

**Definition 1** (Inclusion Property)**.** *The two-step algorithm has "Inclusion Property" if there exists a sequence $\gamma_1, \cdots, \gamma_k, \gamma_{k+1}\cdots$, such that $S_{\gamma_{k+1}} \subseteq S_{\gamma_k}$ holds $\forall k$.*

Given the empirical risk defined in Eq. (7), we have the following theorem.

**Theorem 1.** *Assume that the inclusion property holds. Then there exists a sequence of decreasing $\mathcal{R}_{\gamma_k}$ by applying AMSM.*

*Proof.* To construct a sequence $\mathcal{R}_{\gamma_0} \geq \mathcal{R}_{\gamma_1} \geq \cdots \geq \mathcal{R}_{\gamma_k} \geq \cdots \geq \mathcal{R}_{\gamma}^* \geq 0$, it suffices to show that there exists a sequence, such that $\forall k$, the following holds:

$$\mathcal{R}_{\gamma_{k+1}}(\mathbf{a}_{k+1}, b_{k+1}) \overset{\text{ⓐ}}{\leq} \mathcal{R}_{\gamma_{k+1}}(\mathbf{a}_k, b_k) \overset{\text{ⓑ}}{\leq} \mathcal{R}_{\gamma_k}(\mathbf{a}_k, b_k).$$

Let $B_{\gamma_{k+1}}(\mathbf{a}_{k+1}, b_{k+1})$ be the regularized empirical risk defined in Eq. (11). Given $\mathbf{a}_{k+1}, b_{k+1}$ the updated optimal solution vectors, it is always true that $B_{\gamma_{k+1}}(\mathbf{a}_{k+1}, b_{k+1}) \leq B_{\gamma_{k+1}}(\mathbf{a}_k, b_k)$. From the definition, we know that $\mathcal{R}_{\gamma_k}(\mathbf{a}_k, b_k) = \frac{|\mathcal{S}_{\gamma_k}|}{N} B_{\gamma_k}(\mathbf{a}_k, b_k)$, then we have $\mathcal{R}_{\gamma_{k+1}}(\mathbf{a}_{k+1}, b_{k+1}) \leq \mathcal{R}_{\gamma_{k+1}}(\mathbf{a}_k, b_k)$ and hence ⓐ holds. Furthermore, since the inclusion property holds, ⓑ is also satisfied by definition in Eq. (7), since $S_{\gamma_{k+1}} \subseteq S_{\gamma_k}$ and hence $\frac{|\mathcal{S}_{\gamma_{k+1}}|}{N} \leq \frac{|\mathcal{S}_{\gamma_k}|}{N}$. It completes the proof. □

Therefore, one way to assure convergence is to choose $\gamma$ at each step, such that the inclusion property is upheld. It suffices to select $\gamma_{k+1}$ such that the following holds:

$$\gamma_{k+1} \geq \gamma_k \tag{14}$$

and

$$\xi_{k+1,i} \leq \gamma_{k+1}, \ \forall i \in \mathcal{I}_{\{\xi_{k,j}:\xi_{k,j}\leq\gamma_{k+1}\}} \tag{15}$$

**Remark 1** (Inclusion property implies increasing penalty on the complexity (condition I.))**.** *We can see from Eq.* (8)*, the complexity of the classifier is penalized by the regularization term $\rho\|\mathbf{w}\|_2^2$. If the inclusion property holds, the weight of the penalty term, which is controlled by $\rho$, will increase w.r.t. iteration $k$. It means that by eliminating more and more training data, the penalty of the model complexity will automatically increase to avoid overfitting.*

**Remark 2** (Reduce the computational cost using the Woodbury identity [23])**.** *At each iteration, the cost for computing the solution vector $\boldsymbol{\beta}_{k+1}$ is $\mathcal{O}(|\mathcal{S}_{\gamma_k}|^3)$ (c.f. Eq.*(12)*). However, if the number of rejected patterns is smaller than the basis size, the computational complexity can be reduced using approximations.*

*Let $\mathbf{V}_{r_k} = \begin{bmatrix}\mathbf{K}_{G\tilde{\mathcal{S}}_{\gamma_k}}\\ \mathbf{e}^T\end{bmatrix}$, where $\tilde{\mathcal{S}}_{\gamma_k} = \mathcal{S}_{\gamma_{k-1}} \setminus \mathcal{S}_{\gamma_k}$ is the subset being removed from the training process at step $k$. When $|\tilde{\mathcal{S}}_{\gamma_k}| \ll m$, we rewrite the matrix $\mathbf{M}_k$ as:*

$$\mathbf{M}_k = \mathbf{M}_{k-1} - \mathbf{V}_{r_k}\mathbf{V}_{r_k}^T + \Delta\rho_k\mathbf{I}_0 \tag{16}$$

*where $\mathbf{I}_0 = \left(\begin{array}{c|c}\mathbf{I} & \mathbf{0}\\ \hline \mathbf{0} & 0\end{array}\right)$ and $\Delta\rho_k = \rho_k - \rho_{k-1} = \frac{N\rho}{|\mathcal{S}_{\gamma_k}|} - \frac{N\rho}{|\mathcal{S}_{k-1}|} = \frac{N|\tilde{\mathcal{S}}_{\gamma_k}|\rho}{|\mathcal{S}_{k-1}||\mathcal{S}_{\gamma_k}|}$. When $\Delta\rho_k$ is a small number, then we have:*

$$\mathbf{M}_k^{-1} = \mathbf{N}_k^{-1} - \Delta\rho_k\mathbf{N}_k^{-1}\mathbf{I}_0\mathbf{N}_k^{-1} + \mathcal{O}(\Delta\rho_k^2) \tag{17}$$

*where $\mathbf{N}_k^{-1} = (\mathbf{M}_{k-1} - \mathbf{V}_{r_k}\mathbf{V}_{r_k}^T)^{-1}$.*

*Therefore, with error up to the order of $\Delta\rho_k^2$, we can approximate $\mathbf{M}_k^{-1}$ using:*

$$\mathbf{M}_k^{-1} \approx \mathbf{N}_k^{-1} - \Delta\rho_k\mathbf{N}_k^{-1}\mathbf{I}_0\mathbf{N}_k^{-1} \tag{18}$$

*where $\mathbf{N}_k^{-1}$ can be updated using the Woodbury identity:*

$$\mathbf{N}_k^{-1} = \mathbf{M}_{k-1}^{-1} + \mathbf{M}_{k-1}^{-1}\mathbf{V}_{r_k}\left(I - \mathbf{V}_{r_k}^T\mathbf{M}_{k-1}^{-1}\mathbf{V}_{r_k}\right)^{-1}\mathbf{V}_{r_k}^T\mathbf{M}_{k-1}^{-1}. \tag{19}$$

*The overall computational complexity is:*

$$\mathcal{O}\left(\max(|\tilde{\mathcal{S}}_{\gamma_k}|^3, m^2)\right) \tag{20}$$

## 4. IMPLEMENTATION

In this section, the implementation of the two-step iterative procedure discussed in Sec. 4.1 is presented.

### 4.1. Inner loop: finding $\gamma_{k+1}$

- Input: $\mathbf{K}_{G\mathcal{D}}$, $\gamma_k$, $\mathbf{M}_k^{-1}$, $\boldsymbol{\beta}_{k+1} = \begin{bmatrix}\mathbf{a}_{k+1}\\ b_{k+1}\end{bmatrix}$, $\mathcal{I}_{\mathcal{S}_{\gamma_k}}$.

1) Initialization:
   - Stopping flag: $F = 0$.
   - Compute $\xi_{k+1,i}$:
   $$\xi_{k+1,i} = 1 - \boldsymbol{\beta}_{k+1}^T\begin{bmatrix}\mathbf{K}_{Gi}\\ 1\end{bmatrix}t_i$$
   where $\mathbf{K}_{Gi} = \mathbf{K}_{G\mathcal{D}}(:,i), \forall i \in \mathcal{I}_{\mathcal{S}_{\gamma_k}}$.
   - Sort all $\xi_{k+1,i}$ in an ascending order,
   $$\xi_{k+1,d_1} \leq \cdots \leq \xi_{k+1,d_{|\mathcal{S}_{\gamma_k}|}}. \tag{21}$$
   - Let $\mathbf{v}_{r,d_i} = \begin{bmatrix}\mathbf{K}_{Gd_i}\\ 1\end{bmatrix}$ and construct $\mathbf{V}_{r_0} = [\mathbf{v}_{r,d_1}\cdots\mathbf{v}_{r,d_m}]$, where
   $$m = \arg\min_m \{\xi_{k+1,d_m} \geq \gamma_k\}. \tag{22}$$
   $$\xi_{k+1,d_1} \leq \cdots \xi_{k+1,d_{m-1}} \leq \gamma_k \leq \xi_{k+1,d_m} \leq \cdots \xi_{k+1,d_{|\mathcal{S}_{\gamma_k}|}}$$
   - Let the iteration number $l = 0$ and $j = m$.
   - Initialize $(\mathbf{M}_{k+1}^0)^{-1}$ by removing matrix $\mathbf{V}_{r_0}$ from $\mathbf{M}_k^{-1}$ using the Woodbury identity (c.f. Eq. (18).

– Let $\mathcal{I}^0_{\mathcal{S}_{\gamma_{k+1}}} = \{i : \xi_{k+1,i} \geq \gamma_k,\ \forall i \in \mathcal{I}_{\mathcal{S}_{\gamma_k}}\}$.

2) Repeat:

– $l \leftarrow l + 1$.

– Let $\mathcal{I}_{\mathcal{S}^l_{\gamma_{k+1}}} = \mathcal{I}_{\mathcal{S}^{l-1}_{\gamma_{k+1}}} \setminus d_j$ and $\mathbf{V}^l_{k+1} = \begin{bmatrix} \mathbf{K}_{GS^l_{\gamma_{k+1}}} \\ \mathbf{e}^T \end{bmatrix}$, where $\mathbf{e} = [1 \cdots 1]^T$.

– Update $(\mathbf{M}^l_{k+1})^{-1}$ by removing vector $\mathbf{v}_{r,d_j}$ using the Woodbury identity:

$$(\mathbf{M}^l_{k+1})^{-1} = (\mathbf{M}^{l-1}_{k+1})^{-1} + \Delta\mathbf{M}^l_{k+1}$$

assuming that $\frac{N\rho}{|\mathcal{S}_{\gamma_k}|(|\mathcal{S}_{\gamma_k}|-1)} \approx 0$, where $\Delta\mathbf{M}^l_{k+1} = \frac{(\mathbf{M}^{l-1}_{k+1})^{-1}\mathbf{v}_{r,d_j}\mathbf{v}^T_{r,d_j}(\mathbf{M}^{l-1}_{k+1})^{-1}}{I - \mathbf{v}^T_{r,d_j}(\mathbf{M}^{l-1}_{k+1})^{-1}\mathbf{v}_{r,d_j}}$.

– Update $\boldsymbol{\beta}^l_{k+2} = (\mathbf{M}^l_{k+1})^{-1}\mathbf{V}^l_{k+1}\mathbf{t}_{\mathcal{S}^l_{\gamma_{k+1}}}$.

– Compute $\xi^l_{k+2,i} = 1 - (\boldsymbol{\beta}^l_{k+2})^T \begin{bmatrix} \mathbf{K}_{Gi} \\ 1 \end{bmatrix}, \forall i \in \mathcal{I}_{\mathcal{D}}$.

– Stop if any of the following stopping criteria are satisfied:

i. $\exists\ q$, such that $\xi^l_{k+2,q} > 1$, for $q \in \mathcal{I}_{\mathcal{D}} \setminus \mathcal{I}_{\mathcal{S}_{\gamma_k}}$ (Eq. (13)).

ii. $\exists\ p$, such that $\xi^l_{k+2,p} > \xi_{k+1,d_j}$, for $\xi_{k+1,p} \leq \xi_{k+1,d_m}$ (Eq. (15)).

iii. Matrix $\mathbf{M}^{l+1}_{k+1}$ is rank deficient.

– $j \leftarrow j + 1$.

3) Return:

– If $j = m$, $F = 1$ (stopping flag for outer loop).

– Else, $\gamma_{k+1} \leftarrow \xi_{k+1,d_j}$, $\mathbf{M}^{-1}_{k+1} \leftarrow (\mathbf{M}^l_{k+1})^{-1}$, $\boldsymbol{\beta}_{k+2} \leftarrow \boldsymbol{\beta}^l_{k+2}$.

- Output: $F, \gamma_{k+1}, \mathbf{M}^{-1}_{k+1}, \boldsymbol{\beta}_{k+2}$.

### 4.2. Outer loop: update $\boldsymbol{\beta}$ and $\mathcal{I}_{\mathcal{S}\gamma}$

- Initialization:
  - $\gamma_0 = 0$, $\boldsymbol{\Phi}_G$, $\mathbf{M}^{-1}_0 = \mathbf{M}^{-1}_{\mathcal{D}}$, $\mathcal{I}_{\mathcal{S}_{\gamma_0}} = \mathcal{I}_{\mathcal{D}}$, $k = 0$.
  - Compute $\boldsymbol{\beta}_1$ (c.f. Eq. (12), (17) and (19)).
  - Identify $\mathcal{I}_{\mathcal{S}_{\gamma_1}}$.
- Repeat:
  - Given $\mathbf{M}^{-1}_k$, $\gamma_k$ and $\boldsymbol{\beta}_{k+1}$, update $\gamma_{k+1}$, $\mathbf{M}^{-1}_{k+1}$, $\boldsymbol{\beta}_{k+2}$ and $F$ according to Sec. 4.1.
  - Identify $\mathcal{I}_{\mathcal{S}_{\gamma_{k+2}}}$.
  - Stop if :$\mathcal{I}_{\mathcal{S}_{\gamma_{k+1}}} = \mathcal{I}_{\mathcal{S}_{\gamma_{k+2}}}$ or $F$ = 1.
  - Else $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta}_{k+1}$, $\mathcal{I}_{\mathcal{S}_\gamma} \leftarrow \mathcal{I}_{\mathcal{S}_{\gamma_{k+2}}}$ and $k = k + 1$.
- Output: $\boldsymbol{\beta}$ and $\mathcal{I}_{\mathcal{S}_\gamma}$.

## 5. RESULTS

We present the results in terms of the testing error on five standard UCI datasets [24] described in Tab. 1. The hyperparameter for rbf kernel is the standard deviation $\sigma$ of the Gaussian kernel and degree $d$ for polynomial kernel. The hyperparameters are chosen using grid search and cross-validation. The searching spaces are $\sigma \in \{0.01, 0.03, 0.05, 0.1, 0.3, 0.5, 5\}$ and $d \in \{2, 4, 6\}$, respectively. The ridge parameter $\rho$ is fixed to be 0.1. The hyperparameter $C$ in L2-SVM is tuned to achieved the best result. We can see from Tab. 1 that AMSM gives a higher classification accuracy compared to L2-SVM.

| Dataset | Size | Classes | L2-SVM Error | AMSM Error |
|---|---|---|---|---|
| Ringnorm | 7400 (-) | 2 | 1.98% | 1.34% |
| Phoneme | 5404 (-) | 2 | 12.16% | 10.67% |
| Landsat | 4435 (2000) | 6 | 8.12% | 7.45% |
| Pendigits | 3498 (-) | 10 | 0.52% | 0.34% |
| Optdigits | 1797 (-) | 10 | 2.08% | 1.00% |

**Table 1**. Testing results comparison of 1) L2-SVM and 2) Adaptive Margin Slack Minimization (AMSM) algorithm. The kernel approximation technique adopted is the Greedy Spectral Embedding [16]. The column "size" indicates the training (testing) size, where 10-fold validation method is employed when marked as (-).

## 6. CONCLUSION

In this paper, we present a new empirical risk objective function that can be considered as an extension to SVM with a squared hinge loss (L2-SVM). An iterative algorithm called Adaptive Margin Slack Minimization (AMSM) is developed for finding a solution with respect to our proposed cost function. The algorithm is based on selecting training data within an adaptive marginal hyperplane in an iterative manner. The advantages are 1) hyperparameter free; 2) better flexibility; and 3) faster convergence compared to L2-SVM due to a more aggressive data elimination at each step. Empirical results on some standard datasets have shown the outperformance of AMSM. Testing and analytics for the overall performance on large-scale datasets are under progress.

# Appendix: Multiclassification

In the multiclass case, we solve the problem using a multivariable linear regression model, where instead of a scalar, the output is a vector taking $-1$ and $+1$ as its possible values. Define each element of vector $\mathbf{T}_{c,i}$:

$$\mathbf{T}_{c,i}(j) = \begin{cases} 1, & \text{if } j = c \\ -1, & \text{if } j \neq c \end{cases} \tag{23}$$

There are three modifications regarding the multiclass learning model compared to the binary case:

1) For the multiclass case, Eq. 12 for updating $\mathbf{a}_k$ and $b_k$ is modified into:

$$\boldsymbol{\beta} = \begin{bmatrix} \mathbf{A} \\ \mathbf{b}^T \end{bmatrix} = \mathbf{M}_S^{-1}\mathbf{V}_S\mathbf{T}_S \tag{24}$$

where $\mathbf{A}$ is a $m \times C$ matrix and $\mathbf{b}$ is a $C \times 1$ vector.

2) The classification criterion becomes:

$$\hat{y}_i = \arg\max_c \mathbf{A}(:,c)^T\boldsymbol{\Phi}_G^T\boldsymbol{\varphi}_i + \mathbf{b}(c) \tag{25}$$

3) Given Eq. (25), the definition of the active set is naturally extended to the following:

$$\mathcal{S}_{\gamma_k} = \{\boldsymbol{\varphi}_{c,i} : \frac{1}{2}(\boldsymbol{\xi}_{c,i}(c) + \tilde{\xi}_{\max,i}) \geq \gamma_k\} \tag{26}$$

where $\boldsymbol{\xi}_{c,i} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} - (\mathbf{A}^T\boldsymbol{\Phi}_G^T\boldsymbol{\varphi}_{c,i} + \mathbf{b})diag(\mathbf{T}_{c,i}), \quad \forall\boldsymbol{\varphi}_{c,i}$,

and $\tilde{\xi}_{\max,i} = \max\left(\{\boldsymbol{\xi}_i(1), \cdots, \boldsymbol{\xi}_i(C)\} \setminus \boldsymbol{\xi}_i(c)\right)$ with $\boldsymbol{\xi}_i(c)$ indicating the $c^{th}$ element of the vector $\forall c$. Note that the choice of $\gamma_k$ does not differ from Sec. 3.

## 7. REFERENCES


[1] A. E. Hoerl & R. W. Kennard, (1970), Ridge regression: Biased estimation for nonorthogonal problems, *Technometrics*, 42(1):80-86.

[2] A. Rahimi, A. & B. Recht, (2007), Random features for large-scale kernel machines. *In Proceeding of NIPS*.

[3] A. Shigeo, (2010), Support Vector Machines for Pattern Classification. Springer-Verlag London.

[4] A. Smola, & B. Schölkopf, (2000), Sparse greedy matrix approximation for machine learning. *Proceedings of the 17th International Conference on Machine Learning,* 911-918.

[5] B. Savas, & I.S. Dhillon, (2011), Clustered low rank approximation of graphs in information science applications. *In SDM*.

[6] B. Schölkopf, and A. J. Smola, (2002), Learning with kernels. MIT Press.

[7] C. Cortes, & V. Vapnik, (1995), Support-vector networks. *Machine Learning*, 20(3): 273.

[8] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, (2004), Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214225.

[9] C. K. I. Williams and M. Seeger, (2001) Using the Nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems*, 13:682688.

[10] F. R. Bach & M. I. Jordan, (2002), Kernel independent component analysis. *Journel of Machine Learning Research*, 3:148.

[11] F. R. Bach & M. I. Jordan, (2005), Predictive low-rank decomposition for kernel methods, *in Proceeding of ICML*, 2005.

[12] F. R. Bach, (2013), Sharp analysis of low-rank kernel matrix approximations, *Proceeding of Conference on Learning Theory, 2013*, 185-209.

[13] G. Fung & O. L. Mangasarian, (2001), Proximal Support Vector Machine Classifiers, *Proceedings of Knowledge Discovery and Data Mining*, 77-86.

[14] J. A. K. Suykens & J. Vandewalle, (1999), Least squares support vector machine classifiers, *Neural Processing Letters*, 9(3): 293-300.

[15] M. Kotti & K. I. Diamantaras, (2015), Efficient binary classification through energy minimisation of slack variables, *Neurocomputing*, Elsevier, 148: 498-511.

[16] M. Ouimet and Y. Bengio, (2005), Greedy spectral embedding, *In Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 253-260.

[17] O. Chapelle, (2007), Training a Support Vector Machine in the Primal, *Neural Computation*, 19(5): 1155-1178.

[18] S. Kumar, M. Mohri, & A. Talwalkar, (2012), Sampling methods for the Nyström method, *Journal of Machine Learning*, Research, 13:981-1006.

[19] S. Si, C. J. Hsieh, & I. Dhillon, (2014), Memory Efficient Kernel Approximation, *Proceedings of The 31st International Conference on Machine Learning*, 701-709.

[20] S.Y. Kung, (2014), *Kernel Methods and Machine Learning*, Cambridge Press, 2014.

[21] S. Fine, & K. Scheinberg, (2001), Efficient SVM Training Using Low-Rank Kernel Representations, *Journal of Machine Learning Research*, 2: 243-264.

[22] V. N. Vapnik, (1998), Statistical Learning Theory, Wiley-Interscience.

[23] W. W. Hager, (1989), Updating the inverse of a matrix. *SIAM Review* 31 (2): 221-239.

[24] http://archive.ics.uci.edu/ml/.