# Fixed-Size Pegasos for Hinge and Pinball Loss SVM

Vilen Jumutc, Xiaolin Huang and Johan A.K. Suykens

***Abstract*— Pegasos has become a widely acknowledged algorithm for learning linear Support Vector Machines. It utilizes properties of hinge loss and theory of strongly convex optimization problems for fast convergence rates and lower computational and memory costs. In this paper we adopt the recently proposed pinball loss for the Pegasos algorithm and show some advantages of using it in a variety of classification problems. First we present the newly derived Pegasos optimization objective with respect to pinball loss and analyze its properties and convergence rates. Additionally we present extensions of the Pegasos algorithm applied to the kernel-induced and Nyström approximated feature maps which introduce non-linearity in the input space. This is done using a Fixed-Size kernel method approach. Second we give experimental results for publicly available UCI datasets to justify the advantages and the importance of pinball loss for achieving a better classification accuracy and greater numerical stability in the partially or fully stochastic setting. Finally we conclude our paper with a brief discussion of the applicability of pinball loss to real-life problems.**

## I. Introduction

RECENT research in linear Support Vector Machines (SVM) [1], [2], [3] justified the importance of the first order approaches in bringing these machine learning techniques to large scale. While computing full gradients sometimes might be not feasible it was proposed recently to apply stochastic approximations [4] to the original optimization problem. The latter approach considerably saves memory and only to some degree increases the number of iterations to converge. In this paper we consider another aspect of learning SVM models in primal and particularly put our attention to using another loss function.

Pegasos [2] has become a widely acknowledged algorithm for learning linear SVM and has attracted research interest because of the strongly convex optimization objective and better convergence bounds. Pegasos utilizes hinge loss which replaces the original linear constraints while making the SVM objective unconstrained. With the proper projection step Pegasos achieves a solution of accuracy $\epsilon$ in $O(\frac{R^2}{\lambda\epsilon})$ iterations where $\lambda$ is the regularization parameter. We would like to stress the fact that the hinge loss plays an important but not the essential role in establishing the results of Pegasos. Other authors [5] experimented with other types of loss (*e.g.* logarithmic loss) and optimization techniques.

In this paper we try to enrich the class of loss functions applicable for Pegasos with pinball loss [6] while establishing some essential theory to support our findings. We show some advantages and potential strengths of using the pinball loss within the Pegasos framework in a variety of classification problems. By the latter we assume achieving better classification performance and greater numerical stability in a partially or fully stochastic setting.

Together with the new loss function we employ in this paper Fixed-Size approach [7], [8], [9] which can be coupled with the Pegasos algorithm for learning linear classifiers in the induced feature space[1] for linearly non-separable cases. This setting generally enables us to solve any classification problem with the Pegasos algorithm and requires only one additional parameter for active subset selection. The details of this approach together with the corresponding algorithm will be given in Section IV.

This paper is organized as follows. Section II provides an introduction to hinge and pinball loss SVM and presents the corresponding properties of both loss functions. Section III briefly outlines Pegasos optimization objective and presents our main results. Section IV explains the Fixed-Size approach and outlines our algorithm for learning the Pagasos with the induced feature space. Experimental setup and numerical results are given in Section V while Section VI concludes the paper.

## II. Hinge and Pinball Loss SVM

The linear hinge loss SVM proposed by [10] takes the following formulation,

$$\min_{w} \quad \frac{\gamma}{2}\|w\|^2 + \frac{1}{m}\sum_{(x,y)\in\mathcal{S}} \mathbb{L}(w;(x,y)), \tag{1}$$

where $\mathcal{S}$ is the training dataset and

$$\mathbb{L}(w;(x,y)) = \max\{0, 1 - y\langle w, x\rangle\} \tag{2}$$

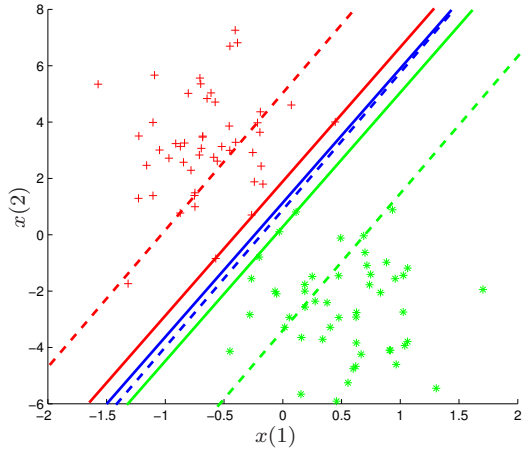
stands for the hinge loss with the $(x, y)$ pair. In this setting the decision function is given by $\hat{y} = \text{sign}(\langle w, x\rangle)$. Hinge loss SVM has a nice geometrical interpretation for separable cases. It tries to find the largest margin between the instances of two classes. Consider a two dimensional example: points in two classes follow Gaussian distribution $\mathcal{N}(\mu_1, \Sigma)$ and $\mathcal{N}(\mu_2, \Sigma)$, respectively, where $\mu_1 = [0.5, -3]^T, \mu_2 = [-0.5, 3]^T$, and $\Sigma = \begin{bmatrix} 0.2 & 0 \\ 0 & 3 \end{bmatrix}$. Fig.1(a) illustrates one sampling by red crosses and green stars. For this data set, via hinge loss SVM (1), we find a decision function $\langle w, x\rangle$ which has the largest margin between $\langle w, x\rangle = \pm 1$ among all the functions satisfying $y\langle w, x\rangle \geq 1$. In Fig.1(a), the obtained lines $\langle w, x\rangle = -1, 0$ and $1$ are illustrated by red,

Vilen Jumutc, Xiaolin Huang and Johan A.K. Suykens are with the Department of Electrical Engineering (ESAT-SCD-SISTA), Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, B-3001, Leuven (Heverlee), Belgium (email: {vilen.jumutc, johan.suykens}@esat.kuleuven.be, huangxl06@mails.tsinghua.edu.cn).

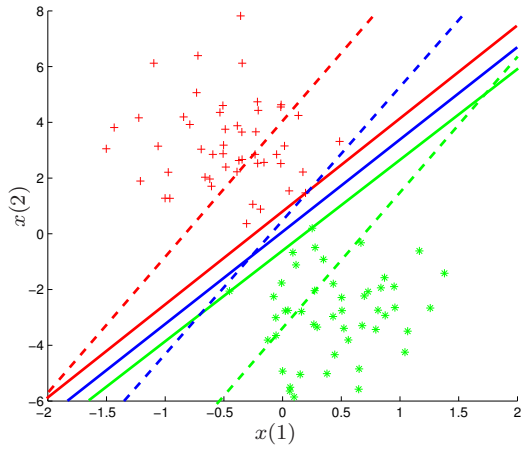
This work was supported by ERC Advanced Grant A-DATADRIVE-B

[1]hereafter the induced feature space stands for the feature space of the RBF kernel approximated by the Fixed-Size approach.

blue, and green solid line, respectively. For this data set, hinge loss SVM in Eq.(1) performs well but it is easy to be affected by noise around decision boundary. That means that hinge loss SVM in Eq.(1) may have very different results for training sets drawn from the same distribution. To illustrate this point, another data set, which has the same distribution in Fig.1(a), and the corresponding result of Eq.(1) are shown in Fig.1(b). One can see that though the data come from the same distribution, the results of Eq.(1) can be significantly different.




(a)




(b)

Fig. 1. Points in two classes are marked by red crosses and green stars. The decision functions are shown by red, blue, and red lines, corresponding to $\langle w, x\rangle = -1, 0$, and 1, respectively. The data in (a) and (b) come from the same distribution but the results of hinge loss SVM (the solid lines) differ significantly. By contrast, the results of pinball loss SVM (the dashed lines) are more stable to re-sampling, which is suitable for stochastic gradient methods.

The sensitivity to noise or the instability to re-sampling comes from the fact that in hinge loss SVM, the distance between two sets is measured by the nearest points. Hence, one way to overcome this weak point is to change the definition of distance between two sets. For example, if we use the distance of the nearest $30\%$ points to measure the distance between two sets, the results are less sensitive, as shown by the dashed lines in Fig.1. Such distance is a kind of quantile value, which is closely related to pinball loss $\mathbb{L}_\tau$ defined by

$$\mathbb{L}_\tau(w;(x,y)) = \begin{cases} 1 - y\langle w, x\rangle & y\langle w, x\rangle \le 1, \\ \tau(y\langle w, x\rangle - 1), & y\langle w, x\rangle > 1, \end{cases} \tag{3}$$

where the reasonable range of $\tau$ is $[0, 1]$ as explained in [6]. The pinball loss $\mathbb{L}_\tau$ has been applied for quantile regression, see, e.g. [11] [12] and [13]. Motivated by the relationship between pinball loss and quantile value, we proposed the following pinball loss SVM in [6],

$$\min_w \qquad \frac{\gamma}{2}\|w\|^2 + \frac{1}{m}\sum_{(x,y)\in\mathcal{S}} \mathbb{L}_\tau(w;(x,y)). \tag{4}$$

Hinge loss is a special case of pinball loss in Eq.(3) with $\tau = 0$. Accordingly, pinball loss SVM in Eq.(4) is an extension to hinge loss SVM. It has been shown in [6] that Eq.(4) and Eq.(2) have similar computational complexity and consistency property. Besides, the result of pinball loss SVM is less sensitive to noise around the boundary. Equivalently speaking, using pinball loss SVM, the results corresponding to different training data from the same distribution are stable. For the data in Fig.1, the decision functions obtained by Eq.(4) are shown by dashed lines. Compared with hinge loss SVM in Eq.(1), the results of pinball loss SVM are more stable for re-sampling from the same distribution, which is especially suitable for stochastic gradients methods.

## III. Pegasos with Pinball Loss

In the original paper by Shalev-Shwartz *et al.* [2] the instantaneous optimization objective of Pegasos algorithm is given by

$$f(w;\mathcal{A}_t) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{(x,y)\in\mathcal{A}_t} \mathbb{L}(w;(x,y)), \tag{5}$$

where the hinge loss for the $(x, y)$ pair is denoted by Eq.(2) and $\mathcal{A}_t$ stands for our working random subsample at iteration $t$. We can write down the subgradient of the instantaneous objective as follows

$$\nabla_t = \lambda w_t - \frac{1}{|\mathcal{A}_t|}\sum_{(x,y)\in\mathcal{A}_t^+} yx, \tag{6}$$

where $\mathcal{A}_t^+$ denotes the subset of $\mathcal{A}_t$ for which $\mathbb{L}(w;(x,y)) > 0$. Finally after taking the gradient descent step with learning rate given by $\eta_t = 1/(\lambda t)$ we need to project back our solution onto the set

$$\mathcal{B} = \{w : \|w\| \le 1/\sqrt{\lambda}\}. \tag{7}$$

The latter effectively enables better convergence rates of the Pegasos algorithm.

### A. The algorithm

In this part we provide the main results of using the pinball loss within the Pegasos algorithm. We start with redefining the instantaneous optimization objective by the means of a new loss function which we already defined in Eq.(3). Now we can see that the subgradient term in Eq.(6) can be refined in terms of a new loss function as follows

$$\nabla_t = \lambda w_t - \frac{1}{|\mathcal{A}_t|}\left[\sum_{(x,y)\in\mathcal{A}_t^+} yx - \sum_{(x,y)\in\mathcal{A}_t^-} \tau yx\right] \quad (8)$$

where $\nabla_t$ additionally depends on the $\tau$ parameter of the pinball loss and $\mathcal{A}_t^-$ stands for the subset of $\mathcal{A}_t$ where $y\langle w, x\rangle > 1$ (see Eq.(3)).

Next we present a brief summary of the modified Pegasos in Algorithm 1 and extend it with the pinball loss in Algorithm 2. Next we continue with the analysis of Algorithm 2 in the next subsection. In Algorithm 1 we can see a major "for" loop where gradient and projection steps are taking place and a minor "if" condition which terminates execution if the norm of the difference of two subsequent $w$ vectors is less than $\epsilon$. In Algorithms 1–2 we denote the whole dataset by $\mathcal{S}$ and at each iteration select randomly $m$ samples for computation of the subgradient. Another important issue is related to the computation of the bias term. We should emphasize that the bias term $\rho$ is not part of our instantaneous optimization objective and we perform computation of it just to return convenient and ubiquitous representation of SVM decision function by $\hat{y} = \text{sign}(\langle w, x\rangle + \rho)$ where $\rho$ is returned in the Line 10 and 13 of Algorithm 1 together with $w$ vector. We should note that the incorporation of the bias term is fairly straightforward via adding a constant $c_{d+1} = 1$ to the feature vector $x_i \in \mathbb{R}^d$.

**Algorithm 1:** Pagasos with hinge loss

**Data**: $\mathcal{S}, \lambda, T, k, \epsilon$
1 Select $w_1$ randomly s.t. $\|w^{(1)}\| \leq 1/\sqrt{\lambda}$
2 **for** $t = 1 \rightarrow T$ **do**
3 Set $\eta_t = \frac{1}{\lambda t}$
4 Select $\mathcal{A}_t \subseteq \mathcal{S}$, where $|\mathcal{A}_t| = k$
5 $\rho = \frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{A}_t}(y - \langle w_t, x\rangle), \forall i$
6 $\mathcal{A}_t^+ = \{(x,y) \in \mathcal{A}_t : y(\langle w_t, x\rangle + \rho) < 1\}, \forall i$
7 $w_{t+\frac{1}{2}} = w_t - \eta_t(\lambda w_t - \frac{1}{k}\sum_{(x,y)\in\mathcal{A}_t^+} yx)$
8 $w_{t+1} = \min\left\{1, \frac{1/\sqrt{\lambda}}{\|w_{t+\frac{1}{2}}\|}\right\} w_{t+\frac{1}{2}}$
9 **if** $\|w_{t+1} - w_t\| \leq \epsilon$ **then**
10 **return** $(w_{t+1}, \frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{S}}(y - \langle w_t, x\rangle))$
11 **end**
12 **end**
13 **return** $(w_{T+1}, \frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{S}}(y - \langle w_t, x\rangle))$

By comparing Algorithm 1 and 2 we can see that the major difference is the computation of the gradient step which involves additional parameter $\tau$ and a new subset $\mathcal{A}_t^-$.

**Algorithm 2:** Pagasos with pinball loss

**Data**: $\mathcal{S}, \lambda, \tau, T, k, \epsilon$
1 Select $w_1$ randomly s.t. $\|w^{(1)}\| \leq 1/\sqrt{\lambda}$
2 **for** $t = 1 \rightarrow T$ **do**
3 Set $\eta_t = \frac{1}{\lambda t}$
4 Select $\mathcal{A}_t \subseteq \mathcal{S}$, where $|\mathcal{A}_t| = k$
5 $\rho = \frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{A}_t}(y - \langle w_t, x\rangle), \forall i$
6 $\mathcal{A}_t^+ = \{(x,y) \in \mathcal{A}_t : y(\langle w_t, x\rangle + \rho) < 1\}, \forall i$
7 $\mathcal{A}_t^- = \{(x,y) \in \mathcal{A}_t : y(\langle w_t, x\rangle + \rho) > 1\}, \forall i$
8 $w_{t+\frac{1}{2}} = w_t - \eta_t(\lambda w_t - \frac{1}{k}\left[\sum_{(x,y)\in\mathcal{A}_t^+} yx - \sum_{(x,y)\in\mathcal{A}_t^-} \tau yx\right])$
9 $w_{t+1} = \min\left\{1, \frac{1/\sqrt{\lambda}}{\|w_{t+\frac{1}{2}}\|}\right\} w_{t+\frac{1}{2}}$
10 **if** $\|w_{t+1} - w_t\| \leq \epsilon$ **then**
11 **return** $(w_{t+1}, \frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{S}}(y - \langle w_t, x\rangle))$
12 **end**
13 **end**
14 **return** $(w_{T+1}, \frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{S}}(y - \langle w_t, x\rangle))$

### B. Analysis

In this subsection we present a convergence analysis which brings to our algorithm the same convergence bounds as in Pegasos. We extend the analysis presented in [2] to our new instantaneous objective by presenting Theorem 1. But first we recap the important lemma from [2] which establishes necessary conditions for our theorem.

**Lemma 1** (Shalev-Shwartz et al., 2007)**.** *Let $f_1, \ldots, f_T$ be a sequence of $\lambda$-strongly convex functions w.r.t. the function $\frac{1}{2}\|\cdot\|^2$. Let $\mathcal{B}$ be a closed convex set and define $\prod_{\mathcal{B}}(w) = \arg\min_{w'\in\mathcal{B}}\|w - w'\|$. Let $w_1, \ldots, w_{T+1}$ be a sequence of vectors such that $w_1 \in \mathcal{B}$ and for $t \geq 1$, $w_{t+1} = \prod_{\mathcal{B}}(w_t - \eta_t\nabla_t)$, where $\nabla_t$ is a subgradient of $f_t$ at $w_t$ and $\eta_t = 1/(\lambda t)$. Assume that for all $t$, $\|\nabla_t\| \leq G$. Then, for all $u \in \mathcal{B}$ we have*

$$\frac{1}{T}\sum_{t=1}^{T} f_t(w_t) \leq \frac{1}{T}\sum_{t=1}^{T} f_t(u) + \frac{G^2(1 + \ln(T))}{2\lambda T}.$$

Based on the above lemma, we are now ready to bound the average instantaneous objective of Algorithm 2.

**Theorem 1.** *Assume $\|x\| \leq R$ for all $(x, y) \in \mathcal{S}$. Let*

$$w^* = \arg\min_{\mathrm{w}}\left[\frac{\lambda}{2}\|\mathrm{w}\|^2 + \frac{1}{|\mathcal{A}_\mathrm{t}|}\sum_{(\mathrm{x},\mathrm{y})\in\mathcal{A}_\mathrm{t}} \mathbb{L}_\tau(\mathrm{w}; (\mathrm{x}, \mathrm{y}))\right]$$

*and let $c = (\sqrt{\lambda} + (\tau + 1)R)$. Then, for $T \geq 3$ we have*

$$\frac{1}{T}\sum_{t=1}^{T} f(w_t; \mathcal{A}_t) \leq \frac{1}{T}\sum_{t=1}^{T} f(w^*; \mathcal{A}_t) + \frac{c^2\ln(T)}{\lambda T}.$$

*Proof:* To prove our theorem it suffices to show that all conditions of Lemma 1 hold. First we show that our problem is strongly convex. It is easy to verify that the first term in Eq.(5) is strictly convex with respect to $w$. Since $f$ is a

sum of $\lambda$-strongly convex function $\frac{\lambda}{2}\|w\|$ and another convex function (pinball loss), it is also $\lambda$-strongly convex. Next by assuming $\mathcal{B} = \{w : \|w\| \leq 1/\sqrt{\lambda}\}$ and the fact that $\|x\| \leq R$ we can bound subgradient $\nabla_t$. The explicit form for the subgradient evaluated at point $x$ is given in Eq.(8). Using the triangular inequality and denoting 2-norm by $\|\cdot\|$ one obtains

$$\|\nabla_t\| \leq \lambda\|w\| + \tfrac{1}{|\mathcal{A}_t|}\textstyle\sum_i(1+\tau)\|x_i\| \leq \sqrt{\lambda} + (1+\tau)R.$$

Finally we have to show that $w^* \in \mathcal{B}$. To do so, we need to derive the dual form of our objective for the whole dataset $|\mathcal{S}| = n$ in terms of the dual variables $\eta_i \in [-\frac{\tau}{n}, \frac{1}{n}]$. By [6] we know that dual form of SVM with pinball loss can be cast as the same SVM objective but with $\zeta_i = \alpha_i - \beta_i$ and additional constraints $\alpha_i + \frac{1}{\tau}\beta_i = \frac{1}{n}, \forall i$, such that the final Lagrangian form is given by

$$\max_{\eta_i}\min_{w}\sum_{i=1}^{n}\zeta_i - \frac{\lambda}{2}\|w\|^2$$

and because strong duality holds for the optimal solution *w.r.t* the primal variable $w^*$ and dual variables $\zeta_i^*$ one gets

$$\frac{\lambda}{2}\|w^*\|^2 + \frac{1}{n}\sum_{x\in\mathcal{S}}\mathbb{L}_\tau(w^*;x) = -\frac{\lambda}{2}\|w^*\|^2 + \sum_{i=1}^{n}\zeta_i^*.$$

Rearranging the above, using the non-negativity of the pinball loss and the bound on $\|\zeta\|_1 \leq 1$ we obtain our initial condition of set $\mathcal{B}$: $\|w\| \leq 1/\sqrt{\lambda}$. Now we can plug-in everything back to the inequality in Lemma 1, follow our initial bound on $T$ and complete the proof by putting all together.□

## IV. Fixed-Size approach

### A. Nyström approximation of the RBF kernel feature map

While linear SVM techniques operating in the primal space achieved good generalization capabilities in some specific application areas, like text categorization [14] and microarray analysis [15] one cannot in general deliver solutions with a low generalization error. To overcome restrictions of Algorithm 2 which operates only in the primal space we apply a Fixed-Size approach [9], [8], [7] to approximate the RBF kernel feature map with some higher dimensional explicit and approximate feature vector. In some of our previous works [16] we already exploited the Fixed-Size approach for achieving an approximation of the RBF kernel while bringing the algorithm to the large scale. The latter introduces desired level of the non-linearity in our solution.

First we use an entropy based criterion to select the prototype vectors (small working sample of size $m \ll n$)[2] and construct the RBF kernel matrix $K$ with

$$K_{ij} = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}. \tag{9}$$

[2]see Section 4 of [8] for additional details

Based on the Nyström approximation [7] an expression for the entries of the approximation of the feature map $\hat{\Phi}(x) : \mathbb{R}^d \to \mathbb{R}^m$, with $\hat{\Phi}(x) = (\hat{\Phi}_1(x), \ldots, \hat{\Phi}_m(x))^T$ is given by

$$\hat{\Phi}_i(x) = \frac{1}{\sqrt{\lambda_{i,m}}}\sum_{t=1}^{m}u_{ti,m}k(x_t, x), \tag{10}$$

where $\lambda_{i,m}$ and $u_{i,m}$ denote the $i$-th eigenvalue and the $i$-th eigenvector of $K$ defined in Eq.(9). Using the above expression for $\hat{\Phi}(x)$ we can proceed with the original formulation of Algorithms 1–2 and find the solution of our problem in primal.

### B. Complete procedure with the Fixed-Size approach

In Algorithms 3–4 we present the complete outline of the Pegasos algorithm evaluated together with the Fixed-Size approach and two types of loss functions.

**Algorithm 3:** Fixed-Size Pegasos with hinge loss

**input** : training data $\mathcal{S}$, labeling $Y$, parameters $\lambda, T, k, \epsilon, m$
**output**: mapping $\hat{\Phi}(x), \forall x \in \mathcal{S}$, SVM model given by $w$ and $\rho$

**begin**
  $\mathcal{S}_r \leftarrow$ `FindActiveSet(`$\mathcal{S}, m$`)`;
  $\hat{\Phi}(x) \leftarrow$ `ComputeNystromApprox(`$\mathcal{S}_r$`)`;
  $X \leftarrow [\hat{\Phi}(x_1)^T, \ldots, \hat{\Phi}(x_n)^T]$;
  $[w, \rho] \leftarrow$ `PegasosHL(`$X, Y, \lambda, T, k, \epsilon$`)`;
**end**

In Algorithm 3 "PegasosHL" function stands for the shortcut of Algorithm 1 and "ComputeNystromApprox" function denotes the Fixed-Size part where we first compute $m \times m$ RBF kernel matrix from the data points found by the maximization of Renyi entropy in "FindActiveSet" function and than we apply Eq.(10) to derive our approximate feature map. Finally we stack our explicit feature vectors in matrix $X$ and proceed to the actual Pegasos implementation.

**Algorithm 4:** Fixed-Size Pegasos with pinball loss

**input** : training data $\mathcal{S}$, labeling $Y$, parameters $\lambda, \tau, T, k, \epsilon, m$
**output**: mapping $\hat{\Phi}(x), \forall x \in \mathcal{S}$, SVM model given by $w$ and $\rho$

**begin**
  $\mathcal{S}_r \leftarrow$ `FindActiveSet(`$\mathcal{S}, m$`)`;
  $\hat{\Phi}(x) \leftarrow$ `ComputeNystromApprox(`$\mathcal{S}_r$`)`;
  $X \leftarrow [\hat{\Phi}(x_1)^T, \ldots, \hat{\Phi}(x_n)^T]$;
  $[w, \rho] \leftarrow$ `PegasosPBL(`$X, Y, \lambda, \tau, T, k, \epsilon$`)`;
**end**

In Algorithm 4 "PegasosPBL" function stands for the shortcut of Algorithm 2 and the rest procedure is the same as in Algorithm 3.

## V. Experiments

### A. Setup

In all our experiments we use a 2-step procedure for tuning the $\lambda$ parameter of Algorithm 2 and bandwidth $\sigma$ of RBF kernel used to approximate our feature map in Eq.(10). This procedure consists of Coupled Simulated Annealing [17] initialized with 5 random sets of parameters for the first step and the simplex method [18] for the second step. After CSA converges to some local minima we select the tuple of parameters that attains the lowest error and start the simplex procedure to refine our selection. On every iteration step for CSA and simplex method we proceed with a 10-fold cross-validation. While being considerably faster than the straightforward grid search technique obtained parameters tend to vary more because of the randomness in initialization.

For both Pegasos-based algorithms (with hinge and pinball losses) we approximate RBF kernel with Fixed-Size approach [8] where the $\sigma$ parameter was inferred via cross-validation procedure described above. The active subset was selected via maximization of Renyi entropy. The size of this subset $m$ was set to be $\sqrt{n}$ for all UCI and toy datasets of size less than 10000 and $\sqrt{n}/5$ for larger datasets. Parameter $\epsilon$ in Algorithm 2 was to $1e-6$ and pinball loss related $\tau$ parameter was provided separately for each experiment. Finally we performed two different settings with the fixed $m$ parameter in Algorithm 2. During the first setup (partially stochastic) we set $k$ to be 10% of $|\mathcal{S}|$ and in the second one (fully stochastic) $k = 1$. We first consider the toy dataset described in Section II. Experiments with the toy dataset were performed with the increasing number of switched labels (error) subsequently set to $0.05$, $0.15$ and $0.35$. All experiments were repeated 50 times with the random split to training and test sets in proportion 1:1. 10-fold cross-validation was performed only on training set. Descriptions of all public [19] and toy datasets we can find in Table I.

TABLE I
Involved datasets

| Dataset | # of attributes | # of classes | # of data points |
|---|---|---|---|
| Toy Data | 2 | 2 | 10000 |
| Pima | 8 | 2 | 768 |
| White Wine | 12 | 2 | 4898 |
| Sambase | 57 | 2 | 4601 |
| Transfusion | 5 | 2 | 748 |
| Magic | 11 | 2 | 19020 |
| Shuttle | 9 | 2 | 58000 |
| Skin | 4 | 2 | 245057 |

### B. Results

In this subsection we give numerical results on running Pegasos-based Algorithm 2 with pinball loss in different settings and choices of user-defined parameter $\tau$. In Table II we can see our results for toy dataset generated with different number of switched labels (artificial noise). From Table II we can notice that everywhere pinball loss incorporated within Algorithm 2 attains better classification rates and smaller errors which are very close to initially embedded noise.

TABLE II
Test errors for Pegasos

| Dataset (% of errors) | Hinge Loss | Pinball Loss $\tau = 0.1$ | Pinball Loss $\tau = 0.5$ | Pinball Loss $\tau = 1$ |
|---|---|---|---|---|
| Toy Data (5%) | 0.08262 | **0.06908** | 0.06926 | 0.07446 |
| Toy Data (15%) | 0.18753 | **0.15843** | 0.16141 | 0.16538 |
| Toy Data (35%) | 0.36094 | 0.31829 | 0.32335 | **0.31571** |

In Table III we can observe results for UCI datasets within partially stochastic setting where we set $k$ to be 10% of $|\mathcal{S}|$. We can notice that we perform almost equally good as hinge loss while giving better classification rates for Pima, Wine and Skin datasets with $\tau = 1$.

TABLE III
Test errors for Pegasos with $k = 10\%$ of $|\mathcal{S}|$
(partially stochastic)

| Dataset | Hinge Loss | Pinball Loss $\tau = 0.1$ | Pinball Loss $\tau = 0.5$ | Pinball Loss $\tau = 1$ |
|---|---|---|---|---|
| Pima | 0.28885 | 0.28417 | 0.27797 | **0.27625** |
| Spambase | **0.13496** | 0.13992 | 0.15047 | 0.15923 |
| Transfusion | **0.23684** | 0.23904 | 0.23824 | 0.23989 |
| White Wine | 0.27281 | 0.27124 | 0.26340 | **0.26274** |
| Magic | **0.17992** | 0.19330 | 0.20912 | 0.21371 |
| Shuttle | **0.01015** | 0.01943 | 0.02515 | 0.02705 |
| Skin | 0.00944 | 0.00997 | 0.01039 | **0.00859** |

In Table IV we can see that when going to a completely stochastic setting with $k = 1$ advantages of pinball loss become more apparent and degradation of performance is not so crucial in comparison with hinge loss. We can notice that for almost all UCI datasets optimal value of $\tau = 0.5$. We can conclude that pinball loss SVM is more stable for re-sampling as shown in Figure 1(b). Comparing our approach to Pegasos with hinge loss we can see general sustainability in fully stochastic setting which leads to immediate and practical application-wise importance in online learning.

TABLE IV
Test errors for Pegasos with $k = 1$
(fully stochastic)

| Dataset | Hinge Loss | Pinball Loss $\tau = 0.1$ | Pinball Loss $\tau = 0.5$ | Pinball Loss $\tau = 1$ |
|---|---|---|---|---|
| Pima | 0.28896 | 0.29422 | **0.28870** | 0.29198 |
| Spambase | 0.21444 | 0.21229 | **0.20816** | 0.21903 |
| Transfusion | 0.23406 | 0.23465 | **0.23396** | 0.23465 |
| White Wine | 0.29607 | 0.29526 | 0.29694 | **0.28898** |
| Magic | 0.22667 | **0.22385** | 0.22481 | 0.22750 |
| Shuttle | 0.04505 | 0.04145 | **0.03499** | 0.03736 |
| Skin | 0.02705 | 0.02498 | **0.02172** | 0.02401 |

### C. Convergence

In this subsection we present some plots which give a better understanding of the convergence properties for the

Pegasos algorithm with pinball loss. On plots we see the convergence over time for Pegasos with hinge loss (blue) and pinball loss (red) incorporated into the instantaneous optimization objective. In general we should note that the pinball loss should give a higher value of optimization objective and thereafter we are more interested in the convergence rates. We experiment with different values of $\lambda$ and $k$ parameters in Algorithm 2.

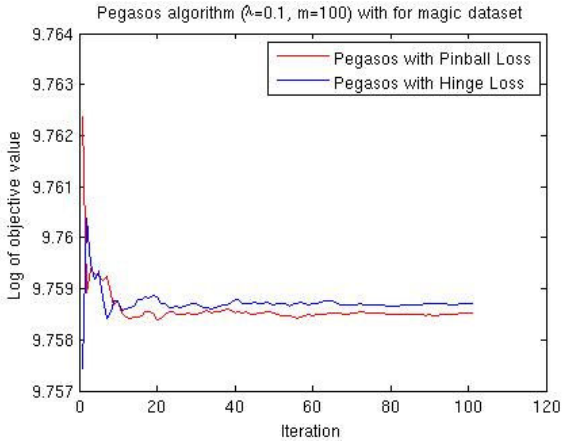



Fig. 2. Convergence of Pegasos algorithm for Magic dataset in a short term (100 iterations) for hinge loss (blue) and pinball loss (red) respectively. In the experimental setup $\lambda = 0.1$ and $k = 100$.

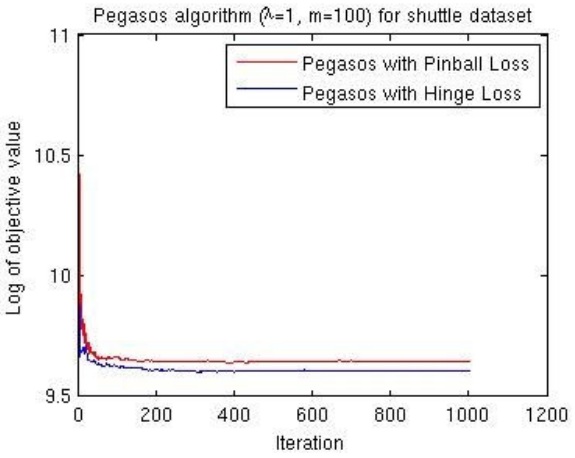



Fig. 3. Convergence of Pegasos algorithm for Shuttle dataset in a long term (1000 iterations) for hinge loss (blue) and pinball loss (red) respectively. In the experimental setup $\lambda = 1$ and $k = 100$.

In Figures 2–3 we could see that Pegasos with pinball loss gives more stable and even better results. In Figure 2 the pinball loss reaches smaller objective values and in Figure 3 the convergence to near-optimal solution is faster than the corresponding convergence for hinge loss. We also observe more numerical stability in convergence for pinball loss which is justified by a smaller number of sudden peaks in the objective value.

## VI. Conclusion

In this paper we proposed an extension of the Pegasos algorithm for pinball loss and showed that it results into better classification rates and better numerical stability while running with pinball loss incorporated into its instantaneous optimization objective. We showed that the Fixed-Size approach helps to deal with linearly non-separable cases while applying the same learning procedure used for the linear SVM. Extensive numerical experiments show certain advantages of using the Pegasos algorithm with pinball loss and the Fixed-Size approach in the fully stochastic setting which in the turn implies a practical application-wise importance in online learning.

## Acknowledgments

This work was supported in part by the scholarship of the Flemish Government; Research Council KUL: GOA/11/05 Ambiorics, GOA/10/09 MaNet, CoE EF/05/006 Optimization in Engineering (OPTEC), IOF-SCORES4CHEM, several PhD/postdoc & fellow grants; Flemish Government:FWO: PhD/postdoc grants, projects: G0226.06 (cooperative systems and optimization), G.0302.07 (SVM/Kernel), G.0320.08 (convex MPC), G.0558.08 (Robust MHE), G.0557.08 (Glycemia2), G.0588.09 (Brain-machine) research communities (WOG: ICCoS, ANMMM, MLDM); G.0377.09 (Mechatronics MPC), G.0377.12 (Structured models), IWT: PhD Grants, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, SBO POM, O&O-Dsquare; Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007-2011); IBBT; EU: ERNSI; ERC AdG A-DATADRIVE-B, FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, FP7-EMBOCON (ICT-248940); Contract Research: AMINAL; Other: Helmholtz: viCERP, ACCM, Bauknecht, Hoerbiger. Johan Suykens is a professor at KU Leuven, Belgium.

## References

[1] T. Joachims, "Training linear SVMs in linear time," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '06. New York, NY, USA: ACM, 2006, pp. 217–226.

[2] S. Shalev-Shwartz, Y. Singer, and N. Srebro, "Pegasos: Primal Estimated sub-GrAdient SOlver for SVM," in *Proceedings of the 24th international conference on Machine learning*, ser. ICML '07, New York, NY, USA, 2007, pp. 807–814.

[3] O. Chapelle, "Training a support vector machine in the primal," *Neural Computation*, vol. 19, pp. 1155–1178, 2007.

[4] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM J. on Optimization*, vol. 19, no. 4, pp. 1574–1609, Jan. 2009.

[5] K. wei Chang, C. jui Hsieh, C. jen Lin, and L. Bottou, "A dual coordinate descent method for large-scale linear SVM," in *ICML*, 2008.

[6] X. Huang, L. Shi, and J. A. K. Suykens, "Support vector machine classifier with pinball loss," Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, Leuven, Tech. Rep. KUL-12-162, 2012.

[7] C. Williams and M. Seeger, "Using the nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems 13*. MIT Press, 2001, pp. 682–688.

[8] K. De Brabanter, J. De Brabanter, J. A. K. Suykens, and B. De Moor, "Optimized fixed-size kernel models for large data sets," *Comput. Stat. Data Anal.*, vol. 54, no. 6, pp. 1484–1504, Jun. 2010.

[9] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Scientific, 2002.

[10] V. Vapnik, *Statistical learning theory*, 1st ed. Wiley, Sep. 1998.

[11] R. Koenker, *Quantile Regression*, ser. Econometric Society Monographs. Cambridge University Press, 2005.

[12] A. Christmann and I. Steinwart, "How SVMs can estimate quantiles and the median," in *NIPS*, 2007, pp. 305–312.

[13] I. Steinwart and A. Christmann, "Estimating conditional quantiles with the help of the pinball loss," *Bernoulli*, vol. 17, no. 1, pp. 211–225, 2011.
[14] Y. Gao and S. Sun, "An empirical evaluation of linear and nonlinear kernels for text classification using support vector machines," in *FSKD*, 2010, pp. 1502–1505.
[15] J. Li, Y. Jia, J. Du, and F. Yu, "A new support vector machine for microarray classification and adaptive gene selection," in *Proceedings of the 2009 conference on American Control Conference*, ser. ACC'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 5410–5415.
[16] V. Jumutc and J. A. K. Suykens, "Multi-class supervised novelty detection," Submitted.
[17] S. Xavier-De-Souza, J. A. K. Suykens, J. Vandewalle, and D. Bollé, "Coupled simulated annealing," *IEEE Trans. Sys. Man Cyber. Part B*, vol. 40, no. 2, pp. 320–335, Apr. 2010.
[18] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, vol. 7, pp. 308–313, 1965.
[19] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml