Approximation Vector Machines
for Large-scale Online Learning

Trung Le   TRUNG.L@DEAKIN.EDU.AU

Tu Dinh Nguyen   TU.NGUYEN@DEAKIN.EDU.AU

Vu Nguyen   V.NGUYEN@DEAKIN.EDU.AU

Dinh Phung   DINH.PHUNG@DEAKIN.EDU.AU

Centre for Pattern Recognition and Data Analytics,
School of Information Technology,
Deakin University, Australia, Waurn Ponds Campus

Abstract

One of the most challenging problems in kernel online learning is to bound the model size and to promote model sparsity. Sparse models not only improve computation and memory usage, but also enhance the generalization capacity – a principle that concurs with the law of parsimony. However, inappropriate sparsity modeling may also significantly degrade the performance. In this paper, we propose Approximation Vector Machine (AVM), a model that can simultaneously encourage sparsity and safeguard its risk in compromising the performance. In an online setting context, when an incoming instance arrives, we approximate this instance by one of its neighbors whose distance to it is less than a predefined threshold. Our key intuition is that since the newly seen instance is expressed by its nearby neighbor the optimal performance can be analytically formulated and maintained. We develop theoretical foundations to support this intuition and further establish an analysis for the common loss functions including Hinge, smooth Hinge, and Logistic (i.e., for the classification task) and ℓ_1, ℓ_2, and ε-insensitive (i.e., for the regression task) to characterize the gap between the approximation and optimal solutions. This gap crucially depends on two key factors including the frequency of approximation (i.e., how frequent the approximation operation takes place) and the predefined threshold. We conducted extensive experiments for classification and regression tasks in batch and online modes using several benchmark datasets. The quantitative results show that our proposed AVM obtained comparable predictive performances with current state-of-the-art methods while simultaneously achieving significant computational speed-up due to the ability of the proposed AVM in maintaining the model size.

Keywords: kernel, online learning, large-scale machine learning, sparsity, big data, core set, stochastic gradient descent, convergence analysis

1. Introduction

In modern machine learning systems, data usually arrive continuously in stream. To enable efficient computation and to effectively handle memory resource, the system should be able to adapt according to incoming data. Online learning represents a family of efficient and scalable learning algorithms for building a predictive model incrementally from a sequence

of data examples (Rosenblatt, 1958; Zinkevich, 2003). In contrast to the conventional learning algorithms (Joachims, 1999; Chang and Lin, 2011), which usually require a costly procedure to retrain the entire dataset when a new instance arrives, online learning aims to utilize the new incoming instances to improve the model given the knowledge of the correct answers to previous processed data (and possibly additional available information), making them suitable for large-scale online applications wherein data usually arrive sequentially and evolve rapidly.

The seminal line of work in online learning, referred to as *linear online learning* (Rosenblatt, 1958; Crammer et al., 2006; Dredze et al., 2008), aims at learning a linear predictor in the input space. The crucial limitation of this approach lies in its over-simplified linear modeling choice and consequently may fail to capture non-linearity commonly seen in many real-world applications. This motivated the works in *kernel-based online learning* (Freund and Schapire, 1999; Kivinen et al., 2004) in which a linear model in the feature space corresponding with a nonlinear model in the input space, hence allows one to cope with a variety of data distributions.

One common issue with *kernel-based online learning approach*, also known as the *curse of kernelization*, is that the model size (i.e., the number of vectors with non-zero coefficients) may grow linearly with the data size accumulated over time, hence causing computational problem and potential memory overflow (Steinwart, 2003; Wang et al., 2012). Therefore in practice, one might prefer kernel-based online learning methods with guaranty on a limited and bounded model size. In addition, enhancing model sparsity is also of great interest to practitioners since this allows the generalization capacity to be improved; and in many cases leading to a faster computation. However, encouraging sparsity needs to be done with care since an inappropriate sparsity-encouraging mechanism may compromise the performance. To address the curse of kernelization, budgeted approaches (Crammer et al., 2004; Dekel et al., 2005; Cavallanti et al., 2007; Wang and Vucetic, 2010; Wang et al., 2012; Le et al., 2016a,c) limits the model size to a predefined budget B. Specifically, when the current model size exceeds this budget, a budget maintenance strategy (e.g., removal, projection, or merging) is triggered to recover the model size back to the budget B. In these approaches, determining a suitable value for the predefined budget in a principled way is important, but challenging, since setting a small budget makes the learning faster but may suffer from underfitting, whereas a large budget makes the model fit better to data but may dramatically slow down the training process. An alternative way to address the curse of kernelization is to use random features (Rahimi and Recht, 2007) to approximate a kernel function (Ming et al., 2014; Lu et al., 2015; Le et al., 2016b). For example, Lu et al. (2015) proposed to transform data from the input space to the random-feature space, and then performed SGD in the feature space. However, in order for this approach to achieve good kernel approximation, excessive number of random features is required which could lead to a serious computational issue. To reduce the impact number of random features, Le et al. (2016b) proposed to distribute the model in dual space including the original feature space and the random feature space that approximates the first space.

---

1. In fact, we used a subset of the dataset a9a which has 123 features. We then project all data points onto 3D using t-SNE. We note that the t-SNE does not do clustering, it only reduces the dimensionality into 3D for visualization while trying to preserve the local properties of the data.
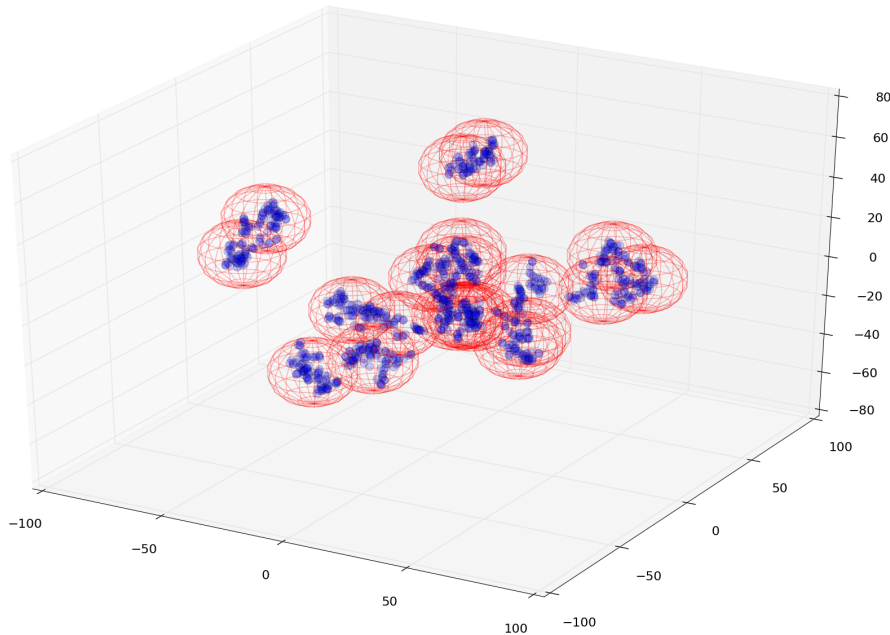
Figure 1: An illustration of the hypersphere coverage for $1,000$ data samples which locate in 3D space. We cover this dataset using hyperspheres with the diameter $\delta = 7.0$, resulting in 20 hypersphere cells as shown in the figure (cf. Sections (6.3,9)). All data samples in a same cell are approximated by a core point in this cell. The model size is therefore significantly reduced from 1,000 to 20.[1]

In this paper, we propose *Approximation Vector Machine* (AVM) to simultaneously encourage model sparsity[2] while preserving the model performance. Our model size is theoretically proven to be bounded regardless of the data distribution and data arrival order. To promote sparsity, we introduce the notion of δ-*coverage* which partitions the data space into overlapped cells whose diameters are defined by δ (cf. Figure 1). This coverage can be constructed in advance or on the fly. Our experiment on the real datasets shows that the coverage can impressively boost sparsity; for example with dataset *KDDCup99* of $4,408,589$ instances, our model size is 115 with $\delta = 3$ (i.e., only 115 cells are required); with dataset *airlines* of $5,336,471$ instances, our model size is 388 with $\delta = 1$.

In an online setting context, when an incoming instance arrives, it can be approximated with the corresponding core point in the cell that contains it. Our intuitive reason is that when an instance is approximated by an its nearby core point, the performance would be largely preserved. We further developed rigorous theory to support this intuitive reason. In particular, our convergence analysis (covers six popular loss functions, namely Hinge, smooth Hinge, and Logistic for classification task and ℓ_2, ℓ_1, and ε-insensitive for regression task) explicitly characterizes the gap between the approximate and optimal solutions. The analysis shows that this gap crucially depends on two key factors including the cell diameter

---

2. Model sparsity can be computed as the ratio of the model size and the number of vectors received so far.

δ and the approximation process. In addition, the cell parameter δ can be used to efficiently control the trade-off between sparsity level and the model performance. We conducted extensive experiments to validate the proposed method on a variety of learning tasks, including classification in batch mode, classification and regression in online mode on several benchmark large-scale datasets. The experimental results demonstrate that our proposed method maintains a comparable predictive performance while simultaneously achieving an order of magnitude speed-up in computation comparing with the baselines due to its capacity in maintaining model size. We would like to emphasize at the outset that unlike budgeted algorithms (e.g., (Crammer et al., 2004; Dekel et al., 2005; Cavallanti et al., 2007; Wang and Vucetic, 2010; Wang et al., 2012; Le et al., 2016a,c, 2017)), our proposed method is nonparametric in the sense that the number of core sets grow with data on demand, hence care should be exercised in practical implementation.

The rest of this paper is organized as follows. In Section 2, we review works mostly related to ours. In Section 3, we present the primal and dual forms of Support Vector Machine (SVM) as they are important background for our work. Section 4 formulates the proposed problem. In Section 5, we discuss the standard SGD for kernel online learning with an emphasis on the *curse of kernelization*. Section 6 presents our proposed AVM with full technical details. Section 7 devotes to study the suitability of loss functions followed by Section 8 where we extend the framework to multi-class setting. Finally, in Section 9, we conduct extensive experiments on several benchmark datasets and then discuss experimental results as well as their implications. In addition, all supporting proof is provided in the appendix sections.

2. Related Work

One common goal of online kernel learning is to bound the model size and to encourage sparsity. Generally, research in this direction can be broadly reviewed into the following themes.

Budgeted Online Learning. This approach limits the model size to a predefined budget B. When the model size exceeds the budget, a budget maintenance strategy is triggered to decrement the model size by one. Three popular budget maintenance strategies are *removal*, *projection*, and *merging*. In the removal strategy, the most redundant support vector is simply eliminated. In the projection strategy, the information of the most redundant support vector is conserved through its projection onto the linear span of the remaining support vectors. The merging strategy first selects two vectors, and then merges them into one before discarding them. Forgetron (Dekel et al., 2005) is the first budgeted online learning method that employs the removal strategy for the budget maintenance. At each iteration, if the classifier makes a mistake, it conducts a three-step update: (i) running the standard Perceptron (Rosenblatt, 1958) update; (ii) shrinking the coefficients of support vectors with a scaling factor; and (iii) removing the support vector with the smallest coefficient. Randomized Budget Perceptron (RBP) (Cavallanti et al., 2007) randomly removes a support vector when the model size overflows the budget. Budget Perceptron (Crammer et al., 2004) and Budgeted Passive Aggressive (BPA-S)(Wang and Vucetic, 2010) attempt to discard the most redundant support vector (SV). Orabona et al. (2009) used the projection to automatically discover the model size. The new vector is added to the support set if its

projection onto the linear span of others in the feature space exceeds a predefined threshold, or otherwise its information is kept through the projection. Other works involving the projection strategy include Budgeted Passive Aggressive Nearest Neighbor (BPA-NN) (Wang and Vucetic, 2010; Wang et al., 2012). The merging strategy was used in some works (Wang and Vucetic, 2009; Wang et al., 2012).

Random Features. The idea of random features was proposed in (Rahimi and Recht, 2007). Its aim is to approximate a shift-invariant kernel using the harmonic functions. In the context of online kernel learning, the problem of model size vanishes since we can store the model directly in the random features. However, the arising question is to determine the appropriate number of random features D to sufficiently approximate the real kernel while keeping this dimension as small as possible for an efficient computation. Ming et al. (2014) investigated the number of random features in the online kernel learning context. Recently, Lu et al. (2015) proposed to run stochastic gradient descent (SGD) in the random feature space rather than that in the real feature space. The theory accompanied with this work shows that with a high confidence level, SGD in the random feature space can sufficiently approximate that in the real kernel space. Nonetheless, in order to achieve good kernel approximation in this approach, excessive number of random features is required, possibly leading to a serious computational issue. To reduce the impact of the number of random features to learning performance, Le et al. (2016b) proposed to store core vectors in the original feature space, whilst storing remaining vectors in the random feature space that sufficiently approximates the first space.

Core Set. This approach utilizes a core set to represent the model. This core set can be constructed on the fly or in advance. Notable works consist of the Core Vector Machine (CVM) (Tsang et al., 2005) and its simplified version, the Ball Vector Machine (BVM) (Tsang et al., 2007). The CVM was based on the achievement in computational geometry (Badoiu and Clarkson, 2002) to reformulate a variation of ℓ_2-SVM as a problem of finding minimal enclosing ball (MEB) and the core set includes the points lying furthest away the current centre of the current MEB. Our work can be categorized into this line of thinking. However, our work is completely different to (Tsang et al., 2005, 2007) in the mechanism to determine the core set and update the model. In addition, the works of (Tsang et al., 2005, 2007) are not applicable for the online learning.

3. Primal and Dual Forms of Support Vector Machine

Support Vector Machine (SVM) Cortes and Vapnik (1995) represents one of the state-of-the-art methods for classification. Given a training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, the data instances are mapped to a feature space using the transformation $\Phi(.)$, and then SVM aims to learn an optimal hyperplane in the feature space such that the margin, the distance from the closest data instance to the hyperplane, is maximized. The optimization problem

of SVM can be formulated as follows

$$\min_{\mathbf{w},b} \left(\frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_{i=1}^{N} \xi_i \right) \tag{1}$$

$$\text{s.t.} : y_i \left(\mathbf{w}^\mathsf{T} \Phi(x_i) + b \right) \geq 1 - \xi_i, \ i = 1, ..., N$$

$$\xi_i \geq 0, \ i = 1, ..., N$$

where $\lambda > 0$ is the regularization parameter, $\Phi(.)$ is the transformation from the input space to the feature space, and $\boldsymbol{\xi} = [\xi_i]_{i=1}^{N}$ is the vector of slack variables.

Using Karush-Kuhn-Tucker theorem, the above optimization problem is transformed to the *dual form* as follows

$$\min_{\boldsymbol{\alpha}} \left(\frac{1}{2} \boldsymbol{\alpha}^\mathsf{T} Q \boldsymbol{\alpha} - \boldsymbol{e}^\mathsf{T} \boldsymbol{\alpha} \right)$$

$$\text{s.t.} : \boldsymbol{y}^\mathsf{T} \boldsymbol{\alpha} = 0$$

$$0 \leq \alpha_i \leq \frac{1}{\lambda N}, \ i = 1, ..., N$$

where $Q = [y_i y_j K(x_i, x_j)]_{i,j=1}^{N}$ is the Gram matrix, $K(x, x') = \Phi(x)^\mathsf{T} \Phi(x')$ is a kernel function, $\boldsymbol{e} = [1]_{N \times 1}$ is the vector of all 1, and $\boldsymbol{y} = [y_i]_{i=1,...,N}^\mathsf{T}$.

The dual optimization problem can be solved using the solvers (Joachims, 1999; Chang and Lin, 2011). However, the computational complexity of the solvers is over-quadratic (Shalev-Shwartz and Srebro, 2008) and the dual form does not appeal to the online learning setting. To scale up SVM and make it appealing to the online learning, we rewrite the constrained optimization problem in Eq. (1) in the *primal form* as follows

$$\min_{\mathbf{w}} \left(\frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_{i=1}^{N} l(\mathbf{w}; x_i, y_i) \right) \tag{2}$$

where $l(\mathbf{w}; x, y) = \max\left(0, 1 - y\mathbf{w}^\mathsf{T} \Phi(x)\right)^3$ is Hinge loss.

In our current interest, the advantages of formulating the optimization problem of SVM in the primal form as in Eq. (2) are at least two-fold. First, it encourages the application of SGD-based method to propose a solution for the online learning context. Second, it allows us to extend Hinge loss to any appropriate loss functions (cf. Section 7) to enrich a wider class of problems that can be addressed.

---

3. We can eliminate the bias b by simply adjusting the kernel.

4. Problem Setting

We consider two following optimization problems for batch and online settings respectively in Eqs. (3) and (4)

$$\min_{\mathbf{w}} f(\mathbf{w}) \triangleq \frac{\lambda}{2} \|\mathbf{w}\|^2 + \mathbb{E}_{(x,y) \sim \mathbb{P}_N} [l(\mathbf{w}; x, y)]$$

$$\triangleq \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_{i=1}^{N} l(\mathbf{w}; x_i, y_i) \tag{3}$$

$$\min_{\mathbf{w}} f(\mathbf{w}) \triangleq \frac{\lambda}{2} \|\mathbf{w}\|^2 + \mathbb{E}_{(x,y) \sim \mathbb{P}_{\mathcal{X}, \mathcal{Y}}} [l(\mathbf{w}; x, y)] \tag{4}$$

where $l(\mathbf{w}; x, y)$ is a *convex* loss function, $\mathbb{P}_{\mathcal{X}, \mathcal{Y}}$ is the joint distribution of (x, y) over $\mathcal{X} \times \mathcal{Y}$ with the data domain $\mathcal{X}$ and the label domain $\mathcal{Y}$, and $\mathbb{P}_N$ specifies the empirical distribution over the training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$. Furthermore, we assume that the convex loss function $l(\mathbf{w}; x, y)$ satisfies the following property: there exists two positive numbers A and B such that $\left\| l'(\mathbf{w}; x, y) \right\| \leq A \|\mathbf{w}\|^{1/2} + B, \forall \mathbf{w}, x, y$. As demonstrated in Section 7, this condition is valid for all common loss functions. Hereafter, for given any function $g(\mathbf{w})$, we use the notation $g'(\mathbf{w}_0)$ to denote the gradient (or any sub-gradient) of $g(.)$ w.r.t $\mathbf{w}$ evaluated at $\mathbf{w}_0$.

It is clear that given a fixed $\mathbf{w}$, there exists a random variable g such that $\mathbb{E}[g \mid \mathbf{w}] = f'(\mathbf{w})$. In fact, we can specify $g = \lambda \mathbf{w} + l'(\mathbf{w}; x_t, y_t)$ where $(x_t, y_t) \sim \mathbb{P}_{\mathcal{X}, \mathcal{Y}}$ or $\mathbb{P}_N$. We assume that a *positive semi-definite* (p.s.d.) and *isotropic* (iso.) kernel Rasmussen and Williams (2005) is used, i.e., $K\left(x, x'\right) = k\left(\left\| x - x' \right\|^2\right)$, where $k : \mathcal{X} \to \mathbb{R}$ is an appropriate function. Let $\Phi(.)$ be the feature map corresponding the kernel (i.e., $K\left(x, x'\right) = \Phi(x)^{\mathsf{T}} \Phi\left(x'\right)$). To simplify the convergence analysis, without loss of generality we further assume that $\|\Phi(x)\|^2 = K(x, x) = 1, \forall x \in \mathcal{X}$. Finally, we denote the optimal solution of optimization problem in Eq. (3) or (4) by $\mathbf{w}^*$, that is, $\mathbf{w}^* = \mathrm{argmin}_{\mathbf{w}} f(\mathbf{w})$.

5. Stochastic Gradient Descent Method

We introduce the standard kernel stochastic gradient descent (SGD) in Algorithm 1 wherein the standard learning rate $\eta_t = \frac{1}{\lambda t}$ is used (Shalev-Shwartz et al., 2007, 2011). Let α_t be a scalar such that $l'(\mathbf{w}_t; x_t, y_t) = \alpha_t \Phi(x_t)$ (we note that this scalar exists for all common loss functions as presented in Section 7). It is apparent that at the iteration t the model $\mathbf{w}_t$ has the form of $\mathbf{w}_t = \sum_{i=1}^{t} \alpha_i^{(t)} \Phi(x_i)$. The vector x_i $(1 \leq i \leq t)$ is said to be a *support vector* if its coefficient $\alpha_i^{(t)}$ is nonzero. The model is represented through the support vectors, and hence we can define the model size to be $\left\| \boldsymbol{\alpha}^{(t)} \right\|_0$ and model sparsity as the ratio between the current model size and t (i.e., $\left\| \boldsymbol{\alpha}^{(t)} \right\|_0 / t$). Since it is likely that α_t is nonzero (e.g., with Hinge loss, it happens if x_t lies in the margins of the current hyperplane), the standard kernel SGD algorithm is vulnerable to the curse of kernelization, that is, the model size, is almost linearly grown with the data size accumulated over time Steinwart (2003). Consequently, the computation gradually becomes slower or even infeasible when the data size grows rapidly.

Algorithm 1 Stochastic Gradient Descent algorithm.

Input: λ, p.s.d. kernel $K\left(.,.\right) = \Phi\left(\cdot\right)^{\mathsf{T}} \Phi\left(\cdot\right)$

1: $\mathbf{w}_1 = \mathbf{0}$
2: **for** $t = 1, 2, \dots T$ **do**
3: Receive (x_t, y_t) $//(x_t, y_t) \sim \mathbb{P}_{\mathcal{X},\mathcal{Y}}$ *or* $\mathbb{P}_N$
4: $\eta_t = \frac{1}{\lambda t}$
5: $g_t = \lambda \mathbf{w}_t + l'\left(\mathbf{w}_t; x_t, y_t\right) = \lambda \mathbf{w}_t + \alpha_t \Phi\left(x_t\right)$
6: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t g_t = \frac{t-1}{t} \mathbf{w}_t - \eta_t \alpha_t \Phi\left(x_t\right)$
7: **end for**
Output: $\overline{\mathbf{w}}_T = \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t$ or $\mathbf{w}_{T+1}$

---

6. Approximation Vector Machines for Large-scale Online Learning

In this section, we introduce our proposed Approximation Vector Machine (AVM) for online learning. The main idea is that we employ an overlapping partition of sufficiently small cells to cover the data domain, i.e., $\mathcal{X}$ or $\Phi(\mathcal{X})$; when an instance arrives, we approximate this instance by a corresponding core point in the cell that contains this instance. Our intuition behind this approximation procedure is that since the instance is approximated by its neighbor, the performance would not be significantly compromised while gaining significant speedup. We start this section with the definition of δ-coverage, its properties and connection with the feature space. We then present AVM and the convergence analysis.

6.1 δ-coverage over a domain

To facilitate our technical development in sequel, we introduce the notion of δ-coverage in this subsection. We first start with the usual definition of a diameter for a set.

Definition 1. (*diameter*) Given a set A, the diameter of this set is defined as $D\left(A\right) = \sup_{x, x' \in A} ||x - x'||$. This is the maximal pairwise distance between any two points in A.

Next, given a domain $\mathcal{X}$ (e.g., the data domain, input space) we introduce the concept of δ-coverage for $\mathcal{X}$ using a collection of sets.

Definition 2. (δ-*coverage*) The collection of sets $\mathcal{P} = (P_i)_{i \in I}$ is said to be an δ-coverage of the domain $\mathcal{X}$ iff $\mathcal{X} \subset \cup_{i \in I} P_i$ and $D\left(P_i\right) \leq \delta, \forall i \in I$ where I is the index set (not necessarily discrete) and each element $P_i \in \mathcal{P}$ is further referred to as a *cell*. Furthermore if the index set I is finite, the collection $\mathcal{P}$ is called a *finite δ-coverage*.

Definition 3. (*core set, core point*) Given an δ-coverage $\mathcal{P} = (P_i)_{i \in I}$ over a given domain $\mathcal{X}$, for each $i \in I$, we select an arbitrary point c_i from the cell P_i, then the collection of all c_i (s) is called the *core set* $\mathcal{C}$ of the δ-coverage $\mathcal{P}$. Each point $c_i \in \mathcal{C}$ is further referred to as a *core point*.

We show that these definitions can be also extended to the feature space with the mapping Φ and kernel K via the following theorem.

8

Theorem 4. *Assume that the p.s.d. and isotropic kernel $K(x, x^{'}) = k\left(||x - x^{'}||^2\right)$, where $k(.)$ is a monotonically continuous decreasing function with $k(0) = 1$, is examined and $\Phi(.)$ is its induced feature map. If $\mathcal{P} = (P_i)_{i \in I}$ is an δ-coverage of the domain $\mathcal{X}$ then $\Phi(\mathcal{P}) = (\Phi(\mathcal{P}_i))_{i \in I}$ is **also** an δ_Φ-coverage of the domain $\Phi(\mathcal{X})$, where $\delta_\Phi = \sqrt{2(1 - k(\delta^2))}$ is a monotonically increasing function and $\lim\limits_{\delta \to 0} \delta_\Phi = 0$.*

In particular, the Gaussian kernel given by $K(x, x^{'}) = \exp(-\gamma \left\| x - x^{'} \right\|^2)$ is a p.s.d. and iso. kernel and $\delta_\Phi = \sqrt{2(1 - \exp(-\gamma\delta^2))}$. Theorem 4 further reveals that the image of an δ-coverage in the input space is an δ_Φ-coverage in the feature space and when the diameter δ approaches 0, so does the induced diameter δ_Φ. For readability, the proof of this theorem is provided in Appendix 11.

We have further developed methods and algorithms to efficiently construct δ-coverage, however to maintain the readability, we defer this construction to Section 6.3.

6.2 Approximation Vector Machines

We now present our proposed Approximation Vector Machine (AVM) for online learning. In an online setting, instances arise on the fly and we need an efficient approach to incorporate incoming instances into the learner. Different from the existing works (cf. Section 2), our approach is to construct an δ-coverage $\mathcal{P} = (P_i)_{i \in I}$ over the input domain $\mathcal{X}$, and for each incoming instance x we find the cell P_i that contains this instance and approximate this instance by a core point $c_i \in P_i$. The coverage $\mathcal{P}$ and core set $\mathcal{C}$ can either be constructed in advance or on the fly as presented in Section 6.3.

In Algorithm 2, when receiving an incoming instance (x_t, y_t), we compute the scalar α_t such that $\alpha_t \Phi(x_t) = l^{'}(\mathbf{w}_t; x_t, y_t)$ (cf. Section 7) in Step 5. Furthermore at Step 7 we introduce a Bernoulli random variable Z_t to govern the approximation procedure. This random variable could be either statistically independent or dependent with the incoming instances and the current model. In Section 9.2, we report on different settings for Z_t and how they influence the model size and learning performance. Our findings at the outset is that, the naive setting with $\mathbb{P}(Z_t = 1) = 1, \forall t$ (i.e., always performing approximation) returns the sparsest model while obtaining comparable learning performance comparing with the other settings. Moreover, as shown in Steps 9 and 11, we only approximate the incoming data instance by the corresponding core point (i.e., c_{i_t}) if $Z_t = 1$. In addition, if $Z_t = 1$, we find a cell that contains this instance in Step 8. It is worth noting that the δ-coverage and the cells are constructed on the fly along with the data arrival (cf. Algorithms 3 and 4). In other words, the incoming data instance might belong to an existing cell or a new cell that has the incoming instance as its core point is created.

Furthermore to ensure that $\|\mathbf{w}_t\|$ is bounded for all $t \geq 1$ in the case of ℓ_2 loss, if $\lambda \leq 1$ then we project $\mathbf{w}_t - \eta_t h_t$ onto the hypersphere with centre origin and radius $y_{\max}\lambda^{-1/2}$, i.e., $\mathcal{B}(\mathbf{0}, y_{\max}\lambda^{-1/2})$. Since it can be shown that with ℓ_2 loss the optimal solution $\mathbf{w}^*$ lies in $\mathcal{B}(\mathbf{0}, y_{\max}\lambda^{-1/2})$ (cf. Theorem 23 in Appendix 12), this operation could possibly result in a faster convergence. In addition, by reusing the previous information, this operation can be efficiently implemented. Finally, we note that with ℓ_2 loss and $\lambda > 1$, we do not need to perform a projection to bound $\|\mathbf{w}_t\|$ since according to Theorem 25 in Appendix 12, $\|\mathbf{w}_t\|$

9

is bounded by $\frac{y_{\max}}{\lambda-1}$. Here it is worth noting that we have defined $y_{\max} = \max_{y \in \mathcal{Y}} |y|$ and this notation is only used in the analysis for the regression task with the ℓ_2 loss.

Algorithm 2 Approximation Vector Machine.

Input: λ, p.s.d. & iso. $K(.,.) = \Phi(\cdot)^{\mathsf{T}} \Phi(\cdot)$, δ-coverage $\mathcal{P} = (P_i)_{i \in I}$

1: $\mathbf{w}_1 = 0$
2: **for** $t = 1, \ldots, T$ **do**
3:    Receive (x_t, y_t) $//(x_t, y_t) \sim \mathbb{P}_{\mathcal{X}, \mathcal{Y}}$ or $\mathbb{P}_N$
4:    $\eta_t = \frac{1}{\lambda t}$
5:    $l'(\mathbf{w}_t; x_t, y_t) = \alpha_t \Phi(x_t)$ $//$cf. Section 7
6:    Sample a Bernoulli random variable Z_t
7:    **if** $Z_t = 1$ **then**
8:       Find $i_t \in I$ such that $x_t \in P_{i_t}$
9:       $h_t = \lambda \mathbf{w}_t + \alpha_t \Phi(c_{i_t})$ $//do\ approximation$
10:    **else**
11:       $h_t = \lambda \mathbf{w}_t + \alpha_t \Phi(x_t)$
12:    **end if**
13:    **if** ℓ_2 loss is used **and** $\lambda \leq 1$ **then**
14:       $\mathbf{w}_{t+1} = \prod_{\mathcal{B}(\mathbf{0}, y_{\max} \lambda^{-1/2})}(\mathbf{w}_t - \eta_t h_t)$
15:    **else**
16:       $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t h_t$
17:    **end if**
18: **end for**
Output: $\overline{\mathbf{w}}_T = \frac{\sum_{t=1}^{T} \mathbf{w}_t}{T}$ or $\mathbf{w}_{T+1}$

In what follows, we present the theoretical results for our proposed AVM including the convergence analysis for a general convex or smooth loss function and the upper bound of the model size under the assumption that the incoming instances are drawn from an arbitrary distribution and arrive in a random order.

6.2.1 Analysis for Generic Convex Loss Function

We start with the theoretical analysis for Algorithm 2. The *decision of approximation* (i.e., the random variable Z_t) could be statistically independent or dependent with the current model parameter $\mathbf{w}_t$ and the incoming instance (x_t, y_t). For example, one can propose an algorithm in which the *decision of approximation* is performed iff the confidence level of the incoming instance w.r.t the current model is greater than 1, i.e., $y_t \mathbf{w}_t^{\mathsf{T}} \Phi(x_t) \geq 1$. We shall develop our theory to take into account all possible cases.

Theorem 5 below establishes an upper bound on the regret under the possible assumptions of the statistical relationship among the decision of approximation, the data distribution, and the current model. Based on Theorem 5, in Theorem 8 we further establish an inequality for the error incurred by a single-point output with a high confidence level.

Theorem 5. *Consider the running of Algorithm 2 where (x_t, y_t) is uniformly sampled from the training set $\mathcal{D}$ or the joint distribution $\mathbb{P}_{\mathcal{X}, \mathcal{Y}}$, the following statements hold*

i) If Z_t and $\mathbf{w}_t$ are independent for all t (i.e., the decision of approximation only depends on the data distribution) then

$$\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{H\left(\log\left(T\right) + 1\right)}{2\lambda T} + \frac{\delta_\Phi M^{1/2} W^{1/2}}{T} \sum_{t=1}^{T} \mathbb{P}\left(Z_t = 1\right)^{1/2}$$

where H, M, W are positive constants.

ii) If Z_t is independent with both (x_t, y_t) and $\mathbf{w}_t$ for all t (i.e., the decision of approximation is independent with the current hyperplane and the data distribution) then

$$\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{H\left(\log\left(T\right) + 1\right)}{2\lambda T} + \frac{\delta_\Phi M^{1/2} W^{1/2}}{T} \sum_{t=1}^{T} \mathbb{P}\left(Z_t = 1\right)$$

iii) In general, we always have

$$\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{H\left(\log\left(T\right) + 1\right)}{2\lambda T} + \delta_\Phi M^{1/2} W^{1/2}$$

Remark 6. Theorem 5 consists of the standard convergence analysis. In particular, if the approximation procedure is never performed, i.e., $\mathbb{P}\left(Z_t = 1\right) = 0, \forall t$, we have the regret bound $\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{H(\log(T)+1)}{2\lambda T}$.

Remark 7. Theorem 5 further indicates that there exists an error gap between the optimal and the approximate solutions. When δ decreases to 0, this gap also decreases to 0. Specifically, when $\delta = 0$ (so does δ_Φ), any incoming instance is approximated by itself and consequently, the gap is exactly 0.

Theorem 8. *Let us define the gap by d_T, which is $\frac{\delta_\Phi M^{1/2} W^{1/2}}{T} \sum_{t=1}^{T} \mathbb{P}\left(Z_t = 1\right)^{1/2}$ (if Z_t is independent with $\mathbf{w}_t$), $\frac{\delta_\Phi M^{1/2} W^{1/2}}{T} \sum_{t=1}^{T} \mathbb{P}\left(Z_t = 1\right)$ (if Z_t is independent with (x_t, y_t) and $\mathbf{w}_t$), or $\delta_\Phi M^{1/2} W^{1/2}$. Let r be any number randomly picked from $\{1, 2, \ldots, T\}$. With the probability at least $1 - \delta$, the following statement holds*

$$f\left(\mathbf{w}_r\right) - f\left(\mathbf{w}^*\right) \leq \frac{H\left(\log\left(T\right) + 1\right)}{2\lambda T} + d_T + \Delta_T \sqrt{\frac{1}{2} \log \frac{1}{\delta}}$$

where $\Delta_T = \max\limits_{1 \leq t \leq T} \left(f\left(\mathbf{w}_t\right) - f\left(\mathbf{w}^*\right)\right)$.

We now present the convergence analysis for the case when we output the α-suffix average result as proposed in Rakhlin et al. (2012). With $0 < \alpha < 1$, let us denote

$$\overline{\mathbf{w}}_T^\alpha = \frac{1}{\alpha T} \sum_{t=(1-\alpha)T+1}^{T} \mathbf{w}_t$$

where we assume that the fractional indices are rounded to their ceiling values.

Theorem 9 establishes an upper bound on the regret for the α-suffix average case, followed by Theorem 10 which establishes an inequality for the error incurred by a α-suffix average output with a high confidence level.

Theorem 9. *Consider the running of Algorithm 2 where (x_t, y_t) is uniformly sampled from the training set $\mathcal{D}$ or the joint distribution $\mathbb{P}_{\mathcal{X}, \mathcal{Y}}$, the following statements hold*

i) If Z_t and $\mathbf{w}_t$ are independent for all t (i.e., the decision of approximation only depends on the data distribution) then

$$\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T^\alpha\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{\lambda\left(1-\alpha\right)}{2\alpha}W_T^\alpha + \frac{\delta_\Phi M^{1/2}W^{1/2}}{\alpha T}\sum_{t=(1-\alpha)T+1}^{T}\mathbb{P}\left(Z_t = 1\right)^{1/2} + \frac{H\log\left(1/\left(1-\alpha\right)\right)}{2\lambda\alpha T}$$

where H, M, W are positive constants and $W_T^\alpha = \mathbb{E}\left[\left\|\mathbf{w}_{(1-\alpha)T+1} - \mathbf{w}^*\right\|^2\right]$.

ii) If Z_t is independent with both (x_t, y_t) and $\mathbf{w}_t$ for all t (i.e., the decision of approximation is independent with the current hyperplane and the data distribution) then

$$\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T^\alpha\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{\lambda\left(1-\alpha\right)}{2\alpha}W_T^\alpha + \frac{\delta_\Phi M^{1/2}W^{1/2}}{\alpha T}\sum_{t=(1-\alpha)T+1}^{T}\mathbb{P}\left(Z_t = 1\right) + \frac{H\log\left(1/\left(1-\alpha\right)\right)}{2\lambda\alpha T}$$

iii) In general, we always have

$$\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T^\alpha\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{\lambda\left(1-\alpha\right)}{2\alpha}W_T^\alpha + \delta_\Phi M^{1/2}W^{1/2} + \frac{H\log\left(1/\left(1-\alpha\right)\right)}{2\lambda\alpha T}$$

Theorem 10. *Let us once again define the induced gap by d_T, which is respectively*

$$\frac{\lambda\left(1-\alpha\right)}{2\alpha}W_T^\alpha + \frac{\delta_\Phi M^{1/2}W^{1/2}}{\alpha T}\sum_{t=(1-\alpha)T+1}^{T}\mathbb{P}\left(Z_t = 1\right)^{1/2} \quad \textit{(if Z_t is independent with $\mathbf{w}_t$)},$$

$$\frac{\lambda\left(1-\alpha\right)}{2\alpha}W_T^\alpha + \frac{\delta_\Phi M^{1/2}W^{1/2}}{\alpha T}\sum_{t=(1-\alpha)T+1}^{T}\mathbb{P}\left(Z_t = 1\right) \quad \textit{(if Z_t is independent with (x_t, y_t) and $\mathbf{w}_t$)},$$

or $\frac{\lambda(1-\alpha)}{2\alpha}W_T^\alpha + \delta_\Phi M^{1/2}W^{1/2}$. Let r be any number randomly picked from $\{(1-\alpha)T+1, 2, \ldots, T\}$. With the probability at least $1 - \delta$, the following statement holds

$$f\left(\mathbf{w}_r\right) - f\left(\mathbf{w}^*\right) \leq \frac{H\log\left(1/\left(1-\alpha\right)\right)}{2\lambda\alpha T} + d_T + \Delta_T^\alpha\sqrt{\frac{1}{2}\log\frac{1}{\delta}}$$

where $\Delta_T^\alpha = \max_{(1-\alpha)T+1 \leq t \leq T}\left(f\left(\mathbf{w}_t\right) - f\left(\mathbf{w}^*\right)\right)$.

Remark 11. Theorems 8 and 10 concern with the theoretical warranty if rendering any single-point output $\mathbf{w}_r$ rather than the average outputs. The upper bound gained in Theorem 10 is tighter than that gained in Theorem 8 in the sense that the quantity $\frac{H\log(1/(1-\alpha))}{2\lambda\alpha T} + \Delta_T^\alpha\sqrt{\frac{1}{2}\log\frac{1}{\delta}}$ decreases faster and may decrease to 0 when $T \to +\infty$ given a confidence level $1 - \delta$.

6.2.2 ANALYSIS FOR SMOOTH LOSS FUNCTION

Definition 12. A loss function $l(\mathbf{w}; x, y)$ is said to be μ-strongly smooth w.r.t a norm $\|.\|$ iff for all $\mathbf{u}, \mathbf{v}$ and (x, y) the following condition satisfies

$$l(\mathbf{v}; x, y) \leq l(\mathbf{u}; x, y) + l'(\mathbf{u}; x, y)^\mathsf{T}(\mathbf{v} - \mathbf{u}) + \frac{\mu}{2}\|\mathbf{v} - \mathbf{u}\|^2$$

Another equivalent definition of μ-strongly smooth function is

$$\left\|l'(\mathbf{u}; x, y) - l'(\mathbf{v}; x, y)\right\|_* \leq \mu\|\mathbf{v} - \mathbf{u}\|$$

where $\|.\|_*$ is used to represent the dual norm of the norm $\|.\|$.

It is well-known that

- ℓ_2 loss is 1-strongly smooth w.r.t $\|.\|_2$.

- Logistic loss is 1-strongly smooth w.r.t $\|.\|_2$.

- τ-smooth Hinge loss (Shalev-Shwartz and Zhang, 2013) is $\frac{1}{\tau}$-strongly smooth w.r.t $\|.\|_2$.

Theorem 13. *Assume that ℓ_2, Logistic, or τ-smooth Hinge loss is used, let us denote $L = \frac{\lambda}{2} + 1$, $\frac{\lambda}{2} + 1$, or $\frac{\lambda}{2} + \tau^{-1}$ respectively. Let us define the gap by d_T as in Theorem 10. Let r be any number randomly picked from $\{(1 - \alpha) T + 1, 2, \dots, T\}$. With the probability at least $(1 - \delta)$, the following statement holds*

$$f(\mathbf{w}_r) - f(\mathbf{w}^*) \leq \frac{H\log(1/(1-\alpha))}{2\lambda\alpha T} + d_T + \frac{LM_T^\alpha}{2}\sqrt{\frac{1}{2}\log\frac{1}{\delta}}$$

where $M_T^\alpha = \max\limits_{(1-\alpha)T+1 \leq t \leq T} \|\mathbf{w}_t - \mathbf{w}^*\|$.

Remark 14. Theorem 13 extends Theorem 10 for the case of smooth loss function. This allows the gap $\frac{H\log(1/(1-\alpha))}{2\lambda\alpha T} + \frac{LM_T^\alpha}{2}\sqrt{\frac{1}{2}\log\frac{1}{\delta}}$ to be quantified more precisely regarding the discrepancy in the model itself rather than that in the objective function. The gap $\frac{H\log(1/(1-\alpha))}{2\lambda\alpha T} + \frac{LM_T^\alpha}{2}\sqrt{\frac{1}{2}\log\frac{1}{\delta}}$ could possibly decrease rapidly when T approaches $+\infty$.

Algorithms	Regret	Budget
Forgetron (Dekel et al., 2005)	NA	MB
PA-I, II (Crammer et al., 2006)	NA	NB
Randomized Budget Perceptron (Cavallanti et al., 2007)	NA	NB
Projection (Orabona et al., 2009)	NA	AB
Kernelized Pegasos (Shalev-Shwartz et al., 2011)	$O\left(\frac{\log(T)}{T}\right)$	NB
Budgeted SGD (Wang et al., 2012)	$O\left(\frac{\log(T)}{T}\right)$	MB
Fourier OGD (Lu et al., 2015)	$O\left(\frac{1}{\sqrt{T}}\right)$	MB
Nystrom OGD (Lu et al., 2015)	$O\left(\frac{1}{\sqrt{T}}\right)$	MB
AVM (average output)	$O\left(\frac{\log(T)}{T}\right)$	AB
AVM (α-suffix average output)	$O\left(\frac{1}{T}\right)$	AB

Table 1: Comparison on the regret bounds and the budget sizes of the kernel online algorithms. On the column of budget size, NB stands for *Not Bound* (i.e., the model size is not bounded and learning method is vulnerable to the curse of kernelization), MB stands for *Manual Bound* (i.e., the model size is manually bounded by a predefined budget), and AB is an abbreviation of *Automatic Bound* (i.e., the model size is automatically bounded and this model size is automatically inferred).

To end this section, we present the regret bound and the obtained budget size for our AVM(s) together with those of algorithms listed in Table 1. We note that some early works on online kernel learning mainly focused on the mistake rate and did not present any theoretical results regarding the regret bounds.

6.2.3 UPPER BOUND OF MODEL SIZE

In what follows, we present the theoretical results regarding the model size and sparsity level of our proposed AVM. Theorem 15 shows that AVM offers a high level of freedom to control the model size. Especially, if we use the always-on setting (i.e., $\mathbb{P}(Z_t = 1) = 1, \forall t$), the model size is bounded regardless of the data distribution and data arrival order.

Theorem 15. *Let us denote $\mathbb{P}(Z_t = 1) = p_t$, $\mathbb{P}(Z_t = 0) = q_t$, and the number of cells generated after the iteration t by M_t. If we define the model size, i.e., the size of support set, after the iteration t by S_t, the following statement holds*

$$\mathbb{E}[S_T] \leq \sum_{t=1}^{T} q_t + \sum_{t=1}^{T} p_t \mathbb{E}[M_t - M_{t-1}] \leq \sum_{t=1}^{T} q_t + \mathbb{E}[M_T]$$

Specially, if we use some specific settings for p_t, we can bound the model size $\mathbb{E}[S_t]$ accordingly as follows
i) If $p_t = 1, \forall t$ then $\mathbb{E}[S_T] \leq \mathbb{E}[M_T] \leq |\mathcal{P}|$, where $|\mathcal{P}|$ specifies the size of the partition $\mathcal{P}$, i.e., its number of cells.
ii) If $p_t = \max\left(0, 1 - \frac{\beta}{t}\right), \forall t$ then $\mathbb{E}[S_T] \leq \beta(\log(T) + 1) + \mathbb{E}[M_T]$.
iii) If $p_t = \max\left(0, 1 - \frac{\beta}{t^\rho}\right), \forall t$, where $0 < \rho < 1$, then $\mathbb{E}[S_T] \leq \frac{\beta T^{1-\rho}}{1-\rho} + \mathbb{E}[M_T]$.

iv) If $p_t = \max\left(0, 1 - \frac{\beta}{t^\rho}\right)$, $\forall t$, where $\rho > 1$, then $\mathbb{E}\left[S_T\right] \leq \beta\zeta\left(\rho\right) + \mathbb{E}\left[M_T\right] \leq \beta\zeta\left(\rho\right) + |\mathcal{P}|$, where $\zeta\left(.\right)$ is ζ- Riemann function defined by the integral $\zeta\left(s\right) = \frac{1}{\Gamma(s)} \int_0^{+\infty} \frac{t^{s-1}}{e^s-1} dt$.

Remark 16. We use two parameters β and ρ to flexibly control the rate of approximation p_t. It is evident that when β increases, the rate of approximation decreases and consequently the model size and accuracy increase. On the other hand, when ρ increases, the rate of approximation increases as well and it follows that the model size and accuracy decreases. We conducted experiment to investigate how the variation of these two parameters influence the model size and accuracy (cf. Section 9.2).

Remark 17. The items i) and iv) in Theorem 15 indicate that if $\mathbb{P}\left(Z_t = 1\right) = p_t = \max\left(0, 1 - \frac{\beta}{t^\rho}\right)$, where $\rho > 1$ or $\rho = +\infty$, then the model size is bounded by $\beta\zeta\left(\rho\right) + |\mathcal{P}|$ (by convention we define $\zeta\left(+\infty\right) = 0$). In fact, the tight upper bound is $\beta\zeta\left(\rho\right) + \mathbb{E}\left[M_T\right]$, where M_T is the number of unique cells used so far. It is empirically proven that M_T could be very small comparing with T and $|\mathcal{P}|$. In addition, since all support sets of $\mathbf{w}_t$ $(1 \leq t \leq T)$ are all lain in the core set, if we output the average $\overline{\mathbf{w}}_T = \frac{\sum_{t=1}^T \mathbf{w}_t}{T}$ or α-suffix average $\overline{\mathbf{w}}_T^\alpha = \frac{1}{\alpha T} \sum_{t=(1-\alpha)T+1}^T \mathbf{w}_T$, the model size is still bounded.

Remark 18. The items ii) and iii) in Theorem 15 indicate that if $\mathbb{P}\left(Z_t = 1\right) = p_t = \max\left(0, 1 - \frac{\beta}{t^\rho}\right)$, where $0 < \rho \leq 1$ then although the model size is not bounded, it would slowly increase comparing with T, i.e., $\log\left(T\right)$ or $T^{1-\rho}$ when ρ is around 1.

6.3 Construction of δ-Coverage

In this section, we return to the construction of δ-coverage defined in Section 6.1 and present two methods to construct a finite δ-coverage. The first method employs hypersphere cells (cf. Algorithm 3) whereas the second method utilizes the hyperrectangle cells (cf. Algorithm 4). In these two methods, the cells in coverage are constructed on the fly when the incoming instances arrive. Both are theoretically proven to be a finite coverage.

Algorithm 3 Constructing hypersphere δ-coverage.

1: $\mathcal{P} = \emptyset$
2: $n = 0$
3: **for** $t = 1, 2, \ldots$ **do**
4:     Receive (x_t, y_t)
5:     $i_t = \operatorname{argmin}_{k \leq n} \|x_t - c_k\|$
6:     **if** $\|x_t - c_{i_t}\| \geq \delta/2$ **then**
7:         $n = n + 1$
8:         $c_n = x_t$
9:         $i_t = n$
10:        $\mathcal{P} = \mathcal{P} \cup \left[\mathcal{B}\left(c_n, \delta/2\right)\right]$
11:     **end if**
12: **end for**

Algorithm 4 Constructing hyperrectangle δ-coverage.

1: $\mathcal{P} = \emptyset$
2: $a = \delta/\sqrt{d}$
3: $n = 0$
4: **for** $t = 1, 2, \ldots$ **do**
5:    Receive (x_t, y_t)
6:    $i_t = 0$
7:    **for** $i = 1$ to n **do**
8:      **if** $\|x_t - c_i\|_\infty < a$ **then**
9:        $i_t = i$
10:        **break**
11:      **end if**
12:    **end for**
13:    **if** $i_t = 0$ **then**
14:      $n = n + 1$
15:      $c_n = x_t$
16:      $i_t = n$
17:      $\mathcal{P} = \mathcal{P} \cup [\mathcal{R}(c_n, a)]$
18:    **end if**
19: **end for**

Algorithm 3 employs a collection of open hypersphere cell $\mathcal{B}(c, R)$, which is defined as $\mathcal{B}(c, R) = \{x \in \mathbb{R}^d : \|x - c\| < R\}$, to cover the data domain. Similar to Algorithm 3, Algorithm 4 uses a collection of open hyperrectangle $\mathcal{R}(c, a)$, which is given by $\mathcal{R}(c, a) = \{x \in \mathbb{R}^d : \|x - c\|_\infty < a\}$, to cover the data domain.

Both Algorithms 3 and 4 are constructed in the common spirit: if the incoming instance (x_t, y_t) is outside all current cells, a new cell whose centre or vertex is this instance is generated. It is noteworthy that the variable i_t in these two algorithms specifies the cell that contains the new incoming instance and is the same as itself in Algorithm 2.

Theorem 19 establishes that regardless of the data distribution and data arrival order, Algorithms 3 and 4 always generate a finite δ-coverage which implies a bound on the model size of AVM. It is noteworthy at this point that in some scenarios of data arrival, Algorithms 3 and 4 might not generate a coverage for the entire space $\mathcal{X}$. However, since the generated sequence $\{x_t\}_t$ cannot be outside the set $\cup_i \mathcal{B}(c_i, \delta)$ and $\cup_i \mathcal{R}(c_i, \delta)$, without loss of generality we can restrict $\mathcal{X}$ to $\cup_i \mathcal{B}(c_i, \delta)$ or $\cup_i \mathcal{R}(c_i, \delta)$ by assuming that $\mathcal{X} = \cup_i \mathcal{B}(c_i, \delta)$ or $\mathcal{X} = \cup_i \mathcal{R}(c_i, \delta)$.

Theorem 19. *Let us consider the coverages formed by the running of Algorithms 3 and 4. If the data domain $\mathcal{X}$ is compact (i.e., close and bounded) then these coverages are all finite δ-coverages whose sizes are all dependent on the data domain $\mathcal{X}$ and independent with the sequence of incoming data instances (x_t, y_t) received.*

Remark 20. Theorem 19 also reveals that regardless of the data arrival order, the model size of AVM is always bounded (cf. Remark 17). Referring to the work of (Cucker and Smale, 2002), it is known that this model size cannot exceed $\left(\frac{4D(\mathcal{X})}{\delta}\right)^d$. However with

many possible data arrival orders, the number of active cells or the model size of AVM is significantly smaller than the aforementioned theoretical bound.

6.4 Complexity Analysis

We now present the computational complexity of our AVM(s) with the hypersphere δ-coverage at the iteration t. The cost to find the hypersphere cell in Step 5 of Algorithm 2 is $O\left(d^2 M_t\right)$. The cost to calculate α_t in Step 6 of Algorithm 2 is $O\left(S_t\right)$ if we consider the kernel operation as a unit operation. If ℓ_2 loss is used and $\lambda \leq 1$, we need to do a projection onto the hypersphere $\mathcal{B}\left(\mathbf{0}, y_{\max}\lambda^{-1/2}\right)$ which requires the evaluation of the length of the vector $\mathbf{w}_t - \eta_t h_t$ (i.e., $\|\mathbf{w}_t - \eta_t h_t\|$) which costs S_t unit operations using incremental implementation. Therefore, the computational operation at the iteration t of AVM(s) is either $O\left(d^2 M_t + S_t\right) = O\left(\left(d^2 + 1\right)S_t\right)$ or $O\left(d^2 M_t + S_t + S_t\right) = O\left(\left(d^2 + 2\right)S_t\right)$ (since $M_t \leq S_t$).

7. Suitability of Loss Functions

We introduce six types of loss functions that can be used in our proposed algorithm, namely Hinge, Logistic, ℓ_2, ℓ_1, $\varepsilon-$insensitive, and τ-smooth Hinge. We verify that these loss functions satisfying the necessary condition, that is, $\left\|l'\left(\mathbf{w}; x, y\right)\right\| \leq A\|\mathbf{w}\|^{1/2} + B$ for some appropriate positive numbers A, B (this is required for our problem formulation presented in Section 4).

For comprehensibility, without loss of generality, we assume that $\|\Phi\left(x\right)\| = K\left(x, x\right)^{1/2} = 1$, $\forall x \in \mathcal{X}$. At the outset of this section, it is noteworthy that for classification task (i.e., Hinge, Logistic, and τ-smooth Hinge cases), the label y is either -1 or 1 which instantly implies $|y| = y^2 = 1$.

- **Hinge loss**

$$l\left(\mathbf{w}; x, y\right) = \max\left\{0, 1 - y\mathbf{w}^\mathsf{T}\Phi\left(x\right)\right\}$$

$$l'\left(\mathbf{w}; x, y\right) = -\mathbb{I}_{\left\{y\mathbf{w}^\mathsf{T}\Phi(x)\leq 1\right\}}y\Phi\left(x\right)$$

where $\mathbb{I}_S$ is the indicator function which renders 1 if the logical statement S is true and 0 otherwise.

Therefore, by choosing $A = 0$, $B = 1$ we have

$$\left\|l'\left(\mathbf{w}; x, y\right)\right\| \leq \|\Phi\left(x\right)\| \leq 1 = A\|\mathbf{w}\|^{1/2} + B$$

- **ℓ_2 loss**

In this case, at the outset we cannot verify that $\left\|l'\left(\mathbf{w}; x, y\right)\right\| \leq A\|\mathbf{w}\|^{1/2} + B$ for all $\mathbf{w}, x, y$. However, to support the proposed theory, we only need to check that $\left\|l'\left(\mathbf{w}_t; x, y\right)\right\| \leq A\|\mathbf{w}_t\|^{1/2} + B$ for all $t \geq 1$. We derive as follows

$$l\left(\mathbf{w}; x, y\right) = \frac{1}{2}\left(y - \mathbf{w}^\mathsf{T}\Phi\left(x\right)\right)^2$$

$$l'\left(\mathbf{w}; x, y\right) = \left(\mathbf{w}^\mathsf{T}\Phi\left(x\right) - y\right)\Phi\left(x\right)$$

17

$$\left\| l'\left(\mathbf{w}_t; x, y\right)\right\| = |\mathbf{w}_t^\mathsf{T}\Phi\left(x\right) + y|\left\|\Phi\left(x\right)\right\| \leq |\mathbf{w}_t^\mathsf{T}\Phi\left(x\right)| + y_{\max}$$

$$\leq \left\|\Phi\left(x\right)\right\|\left\|\mathbf{w}_t\right\| + y_{\max} \leq A\left\|\mathbf{w}_t\right\|^{1/2} + B$$

where $B = y_{\max}$ and $A = \begin{cases} y_{\max}^{1/2}\lambda^{-1/4} & \text{if } \lambda \leq 1 \\ y_{\max}^{1/2}\left(\lambda - 1\right)^{-1/2} & \text{otherwise} \end{cases}$.

Here we note that we make use of the fact that $\left\|\mathbf{w}_t\right\| \leq y_{\max}\left(\lambda - 1\right)^{-1}$ if $\lambda > 1$ (cf. Theorem 25 in Appendix 12) and $\left\|\mathbf{w}_t\right\| \leq y_{\max}\lambda^{-1/2}$ otherwise (cf. Line 12 in Algorithm 2).

- ℓ_1 **loss**

$$l\left(\mathbf{w}; x, y\right) = |y - \mathbf{w}^\mathsf{T}\Phi\left(x\right)|$$

$$l'\left(\mathbf{w}; x, y\right) = \text{sign}\left(\mathbf{w}^\mathsf{T}\Phi\left(x\right) - y\right)\Phi\left(x\right)$$

Therefore, by choosing $A = 0$, $B = 1$ we have

$$\left\| l'\left(\mathbf{w}; x, y\right)\right\| = \left\|\Phi\left(x\right)\right\| \leq 1 = A\left\|\mathbf{w}\right\|^{1/2} + B$$

- **Logistic loss**

$$l\left(\mathbf{w}; x, y\right) = \log\left(1 + \exp\left(-y\mathbf{w}^\mathsf{T}\Phi\left(x\right)\right)\right)$$

$$l'\left(\mathbf{w}; x, y\right) = \frac{-y\exp\left(-y\mathbf{w}^\mathsf{T}\Phi\left(x\right)\right)\Phi\left(x\right)}{\exp\left(-y\mathbf{w}^\mathsf{T}\Phi\left(x\right)\right) + 1}$$

Therefore, by choosing $A = 0$, $B = 1$ we have

$$\left\| l'\left(\mathbf{w}; x, y\right)\right\| < \left\|\Phi\left(x\right)\right\| \leq 1 = A\left\|\mathbf{w}\right\|^{1/2} + B$$

- ε-**insensitive loss**

$$l\left(\mathbf{w}; x, y\right) = \max\left\{0, |y - \mathbf{w}^\mathsf{T}\Phi\left(x\right)| - \varepsilon\right\}$$

$$l'\left(\mathbf{w}; x, y\right) = \mathbb{I}_{\left\{|y - \mathbf{w}^\mathsf{T}\Phi(x)| > \varepsilon\right\}}\text{sign}\left(\mathbf{w}^\mathsf{T}\Phi\left(x\right) - y\right)\Phi\left(x\right)$$

Therefore, by choosing $A = 0$, $B = 1$ we have

$$\left\| l'\left(\mathbf{w}; x, y\right)\right\| \leq \left\|\Phi\left(x\right)\right\| \leq 1 = A\left\|\mathbf{w}\right\|^{1/2} + B$$

- τ-**smooth Hinge loss** Shalev-Shwartz and Zhang (2013)

$$l\left(\mathbf{w}; x, y\right) = \begin{cases} 0 & \text{if } y\mathbf{w}^\mathsf{T}\Phi\left(x\right) > 1 \\ 1 - y\mathbf{w}^\mathsf{T}\Phi\left(x\right) - \frac{\tau}{2} & \text{if } y\mathbf{w}^\mathsf{T}\Phi\left(x\right) < 1 - \tau \\ \frac{1}{2\tau}\left(1 - y\mathbf{w}^\mathsf{T}\Phi\left(x\right)\right)^2 & \text{otherwise} \end{cases}$$

$$l'\left(\mathbf{w}; x, y\right) = -\mathbb{I}_{\left\{y\mathbf{w}^\mathsf{T}\Phi(x) < 1 - \tau\right\}}y\Phi\left(x\right)$$

$$+ \tau^{-1}\mathbb{I}_{1 - \tau \leq y\mathbf{w}^\mathsf{T}\Phi(x) \leq 1}\left(y\mathbf{w}^\mathsf{T}\Phi\left(x\right) - 1\right)y\Phi\left(x\right)$$

Therefore, by choosing $A = 0$, $B = 2$, we have

$$\left\| l'(\mathbf{w}; x, y) \right\| \leq |y| \left\| \Phi(x) \right\| + \tau^{-1} |y| \left\| \Phi(x) \right\| \tau \leq 2$$
$$= A \left\| \mathbf{w} \right\|^{1/2} + B$$

8. Multiclass Setting

In this section, we show that our proposed framework could also easily extend to the multi-class setting. We base on the work of Crammer and Singer (2002) for multiclass classification to formulate the optimization problem in multi-class setting as

$$\min_{W} \left(f(W) \triangleq \frac{\lambda}{2} \|W\|_{2,2}^2 + \frac{1}{N} \sum_{i=1}^{N} l\left(\mathbf{w}_{y_i}^\mathsf{T} \Phi(x_i) - \mathbf{w}_{z_i}^\mathsf{T} \Phi(x_i) \right) \right)$$

where we have defined

$$z_i = \underset{j \neq y_i}{\operatorname{argmax}} \, \mathbf{w}_j^\mathsf{T} \Phi(x_i),$$

$$W = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m], \, \|W\|_{2,2}^2 = \sum_{j=1}^{m} \|\mathbf{w}_j\|^2,$$

$$l(a) = \begin{cases} \max(0, 1-a) & \text{Hinge loss} \\ \log(1 + e^{-a}) & \text{Logistic loss} \end{cases}$$

For the exact update, at the t-th iteration, we receive the instance (x_t, y_t) and modify W as follows

$$\mathbf{w}_j^{(t+1)} = \begin{cases} \frac{t-1}{t} \mathbf{w}_j^{(t)} - \eta_t l'(a) \Phi(x_t) & \text{if } j = y_t \\ \frac{t-1}{t} \mathbf{w}_j^{(t)} + \eta_t l'(a) \Phi(x_t) & \text{if } j = z_t \\ \frac{t-1}{t} \mathbf{w}_j^{(t)} & \text{otherwise} \end{cases}$$

where $a = \mathbf{w}_{y_t}^\mathsf{T} \Phi(x_t) - \mathbf{w}_{z_t}^\mathsf{T} \Phi(x_t)$ and $l'(a) = -\mathbb{I}_{\{a<1\}}$ or $-1/(1 + e^a)$.

The algorithm for Approximation Vector Machine with multiclass setting proceeds as in Algorithm 5.

Algorithm 5 Multiclass Approximation Vector Machine.

Input: λ, p.s.d. & iso. kernel $K\left(.,.\right)$, δ-coverage $\mathcal{P} = \left(P_i\right)_{i \in I}$

1: $W_1 = 0$
2: **for** $t = 1, \ldots, T$ **do**
3:      Receive (x_t, y_t) $//(x_t, y_t) \sim \mathbb{P}_{\mathcal{X},\mathcal{Y}}$ *or* $\mathbb{P}_N$
4:      $a = \mathbf{w}_{y_t}^{\mathsf{T}} \Phi\left(x_t\right) - \max_{j \neq y_t} \mathbf{w}_j^{\mathsf{T}} \Phi\left(x_t\right)$
5:      $W^{(t+1)} = \frac{t-1}{t} W^{(t)}$
6:      Sample a Bernoulli random variable Z_t
7:      **if** $Z_t = 1$ **then**
8:          Find $i_t \in I$ such that $x_t \in P_{i_t}$
9:          $\mathbf{w}_{y_t}^{(t+1)} = \mathbf{w}_{y_t}^{(t+1)} - \eta_t l'\left(a\right) \Phi\left(c_{i_t}\right)$ $//do\ approximation$
10:        $\mathbf{w}_{z_t}^{(t+1)} = \mathbf{w}_{z_t}^{(t+1)} + \eta_t l'\left(a\right) \Phi\left(c_{i_t}\right)$
11:      **else**
12:          $\mathbf{w}_{y_t}^{(t+1)} = \mathbf{w}_{y_t}^{(t+1)} - \eta_t l'\left(a\right) \Phi\left(x_t\right)$
13:          $\mathbf{w}_{z_t}^{(t+1)} = \mathbf{w}_{z_t}^{(t+1)} + \eta_t l'\left(a\right) \Phi\left(x_t\right)$
14:      **end if**
15: **end for**
Output $\overline{W}^{(T)} = \frac{\sum_{t=1}^{T} W^{(t)}}{T}$ or $W^{(t+1)}$

---

9. Experiments

In this section, we conduct comprehensive experiments to quantitatively evaluate the capacity and scalability of our proposed Approximation Vector Machine (AVM) on classification and regression tasks under three different settings:

- *Batch classification*[4]: the regular binary and multiclass classification tasks that follow a standard validation setup, wherein each dataset is partitioned into training set and testing set. The models are trained on the training part, and then their discriminative capabilities are verified on the testing part using classification accuracy measure. The computational costs are commonly measured based on the training time.

- *Online classification*: the binary and multiclass classification tasks that follow a purely online learning setup, wherein there is no division of training and testing sets as in batch setting. The algorithms sequentially receive and process a single data sample turn-by-turn. When an individual data point comes, the models perform prediction to compute the mistake rate first, then use the feature and label information of such data point to continue their learning procedures. Their predictive performances and computational costs are measured basing on the average of mistake rate and execution time, respectively, accumulated in the learning progress on the entire dataset.

- *Online regression*: the regression task that follows the same setting of online classification, except the predictive performances are measured based on the regression error rate accumulated in the learning progress on the entire dataset.

---

4. This setting is also known as offline classification.

Our main goal is to examine the scalability, classification and regression capabilities of AVMs by directly comparing with those of several recent state-of-the-art batch and online learning approaches using a number of datasets with a wide range of sizes. Our models are implemented in Python with Numpy package. The source code and experimental scripts are published for reproducibility[5]. In what follows, we present the data statistics, experimental setup, results and our observations.

9.1 Data Statistics and Experimental Setup

We use 11 datasets whose statistics are summarized in Table 2. The datasets are selected in a diverse array of sizes in order to clearly expose the differences among scalable capabilities of the models. Five of which (*year, covtype, poker, KDDCup99, airlines*) are large-scale datasets with hundreds of thousands and millions of data points, whilst the rest are ordinal-size databases. Except the *airlines*, all of the datasets can be downloaded from LIBSVM[6] and UCI[7] websites.

Dataset	#training	#testing	#features	#classes	Source
a9a	$32,561$	$16,281$	123	2	UCI
w8a	$49,749$	$14,951$	300	2	LIBSVM
cod-rna	$59,535$	$271,617$	8	2	LIBSVM
ijcnn1	$49,990$	$91,701$	22	2	LIBSVM
covtype	$522,911$	$58,101$	54	7	LIBSVM
poker	$25,010$	$1,000,000$	10	10	UCI
KDDCup99	$4,408,589$	$489,842$	41	23	UCI
airlines	$5,336,471$	$592,942$	8	2	ASA
Dataset	**#training**	**#testing**	**#features**	**value**	**Source**
casp	$45,730$	–	9	$[0,1]$	UCI
slice	$53,500$	–	384	$[0,1]$	UCI
year	$515,345$	–	90	$[0,1]$	UCI
airlines	$5,929,413$	–	8	$\mathbb{R}^+$	ASA

Table 2: Data statistics. #training: number of training samples; #testing: number of testing samples.

The airlines dataset is provided by American Statistical Association (ASA[8]). The dataset contains information of all commercial flights in the US from October 1987 to April 2008. The aim is to predict whether a flight will be delayed or not and how long in minutes the flight will be delayed in terms of departure time. The departure delay time is provided in the flight database. A flight is considered *delayed* if its delay time is above 15 minutes, and *non-delayed* otherwise. The average delay of a flight in 2008 was of 56.3 minutes. Following the procedure of (Hensman et al., 2013), we further process the data in two steps. First, we join the data with the information of individual planes basing on their tail numbers in order

---

5. `https://github.com/tund/avm`.

6. `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

7. `https://archive.ics.uci.edu/ml/datasets.html`.

8. The data can be downloaded from `http://stat-computing.org/dataexpo/2009/`.

to obtain the manufacture year. This additional information is provided as a supplemental data source on ASA website. We then extract 8 features of many available fields: the age of the aircraft (computed based on the manufacture year), journey distance, airtime, scheduled departure time, scheduled arrival time, month, day of week and month. All features are normalized into the range $[0, 1]$.

In batch classification experiments, we follow the original divisions of training and testing sets in LIBSVM and UCI sites wherever available. For *KDDCup99*, *covtype* and *airlines* datasets, we split the data into 90% for training and 10% for testing. In online classification and regression tasks, we either use the entire datasets or concatenate training and testing parts into one. The online learning algorithms are then trained in a single pass through the data. In both batch and online settings, for each dataset, the models perform 10 runs on different random permutations of the training data samples. Their prediction results and time costs are then reported by taking the average with the standard deviation of the results over these runs.

For comparison, we employ some baseline methods that will be described in the following sections. Their C++ implementations with Matlab interfaces are published as a part of LIBSVM, BudgetedSVM[9] and LSOKL[10] toolboxes. Throughout the experiments, we utilize RBF kernel, i.e., $K\left(x, x^{'}\right) = \exp\left(-\gamma \left\| x - x^{'} \right\|^{2}\right)$ for all algorithms including ours. We use hypersphere strategy to construct the δ-coverage (cf. Section 6.3), due to its better performance than that of hyperrectangle approach during model evaluation. All experiments are conducted using a Windows machine with 3.46GHz Xeon processor and 96GB RAM.

9.2 Model Evaluation on The Effect of Hyperparameters

In the first experiment, we investigate the effect of hyperparameters, i.e., δ-coverage diameter, sampling parameters β and ρ (cf. Section 6.2.3) on the performance of AVMs. Particularly, we conduct an initial analysis to quantitatively evaluate the sensitivity of these hyperparameters and their impact on the predictive accuracy and model size. This analysis provides a heuristic approach to find the best setting of hyperparameters. Here the AVM with Hinge loss is trained following the online classification scheme using two datasets *a9a* and *cod-rna*.

To find the plausible range of coverage diameter, we use a heuristic approach as follows. First we compute the mean and standard deviation of pairwise Euclidean distances between any two data samples. Treating the mean as the radius, the coverage diameter is then varied around twice of this mean bounded by twice of the standard deviation. Fig. 2a and Fig. 3a report the average mistake rates and model sizes of AVMs with respect to (w.r.t) these values for datasets *a9a* and *cod-rna*, respectively. Here we set $\beta = 0$ and $\rho = 1.0$. There is a consistent pattern in both figures: the classification errors increase for larger δ whilst the model sizes decrease. This represents the trade-off between model performance and model size via the model coverage. To balance the performance and model size, in these cases, we can choose $\delta = 7.0$ for *a9a* data and $\delta = 1.0$ for *cod-rna* data.
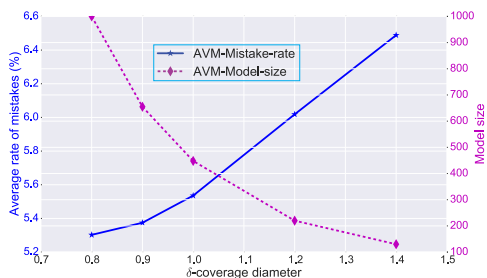
---

9. http://www.dabi.temple.edu/budgetedsvm/index.html.
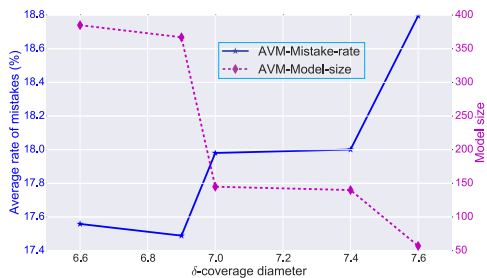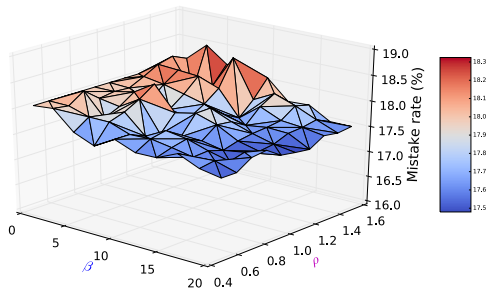
10. http://lsokl.stevenhoi.com/.

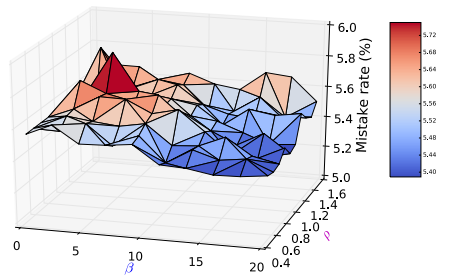(a) The effect of δ-coverage diameter on the mistake rate and model size.

(b) The effect of β and ρ on the classification mistake rate. $\beta = 0$ means always approximating.

Figure 2: Performance evaluation of AVM with Hinge loss trained using *a9a* dataset with different values of hyperparameters.

(a) The effect of δ-coverage diameter on the mistake rate and model size.

(b) The effect of β and ρ on the classification mistake rate. $\beta = 0$ means always approximating.

Figure 3: Performance evaluation of AVM with Hinge loss trained using *cod-rna* dataset with different values of hyperparameters.

Fixing the coverage diameters, we vary β and ρ in 10 values monotonically increasing from 0 to 10 and from 0.5 to 1.5, respectively, to evaluate the classification performance. The smaller β and larger ρ indicate that the machine approximates the new incoming data more frequently, resulting in less powerful prediction capability. This can be observed in Fig. 2b and Fig. 3b, which depict the average mistake rates in 3D as a function of these values for dataset *a9a* and *cod-rna*. Here $\beta = 0$ means that the model always performs approximation without respect to the value of ρ. From these visualizations, we found that the AVM with always-on approximation mode still can achieve fairly comparable classification results. Thus we set $\beta = 0$ for all following experiments.

9.3 Batch Classification

We now examine the performances of AVMs in classification task following batch mode. We use eight datasets: *a9a*, *w8a*, *cod-rna*, *KDDCup99*, *ijcnn1*, *covtype*, *poker* and *airlines*

(delayed and non-delayed labels). We create two versions of our approach: AVM with Hinge loss (AVM-Hinge) and AVM with Logistic loss (AVM-Logit). It is noteworthy that the Hinge loss is not a smooth function with undefined gradient at the point that the classification confidence $yf(x) = 1$. Following the sub-gradient definition, in our experiment, we compute the gradient given the condition that $yf(x) < 1$, and set it to 0 otherwise.

Baselines. For discriminative performance comparison, we recruit the following state-of-the-art baselines to train kernel SVMs for classification in batch mode:

- LIBSVM: one of the most widely-used and state-of-the-art implementations for batch kernel SVM solver (Chang and Lin, 2011). We use the one-vs-all approach as the default setting for the multiclass tasks;

- LLSVM: low-rank linearization SVM algorithm that approximates kernel SVM optimization by a linear SVM using low-rank decomposition of the kernel matrix (Zhang et al., 2012);

- BSGD-M: budgeted stochastic gradient descent algorithm which extends the Pegasos algorithm (Shalev-Shwartz et al., 2011) by introducing a merging strategy for support vector budget maintenance (Wang et al., 2012);

- BSGD-R: budgeted stochastic gradient descent algorithm which extends the Pegasos algorithm (Shalev-Shwartz et al., 2011) by introducing a removal strategy for support vector budget maintenance (Wang et al., 2012);

- FOGD: Fourier online gradient descent algorithm that applies the random Fourier features for approximating kernel functions (Lu et al., 2015);

- NOGD: Nystrom online gradient descent (NOGD) algorithm that applies the Nystrom method to approximate large kernel matrices (Lu et al., 2015).

Hyperparameters setting. There are a number of different hyperparameters for all methods. Each method requires a different set of hyperparameters, e.g., the regularization parameters (C in LIBSVM, λ in Pegasos and AVM), the learning rates (η in FOGD and NOGD), the coverage diameter (δ in AVM) and the RBF kernel width (γ in all methods). Thus, for a fair comparison, these hyperparameters are specified using cross-validation on training subset.

Particularly, we further partition the training set into 80% for learning and 20% for validation. For large-scale databases, we use only 1% of training set, so that the searching can finish within an acceptable time budget. The hyperparameters are varied in certain ranges and selected for the best performance on the validation set. The ranges are given as follows: $C \in \{2^{-5}, 2^{-3}, ..., 2^{15}\}$, $\lambda \in \{2^{-4}/N, 2^{-2}/N, ..., 2^{16}/N\}$, $\gamma \in \{2^{-8}, 2^{-4}, 2^{-2}, 2^{0}, 2^{2}, 2^{4}, 2^{8}\}$, $\eta \in \{16.0, 8.0, 4.0, 2.0, 0.2, 0.02, 0.002, 0.0002\}$ where N is the number of data points. The coverage diameter δ of AVM is selected following the approach described in Section 9.2. For the budget size B in NOGD and Pegasos algorithm, and the feature dimension D in FOGD for each dataset, we use identical values to those used in Section 7.1.1 of Lu et al. (2015).

Results.   The classification results, training and testing time costs are reported in Table 3. Overall, the batch algorithms achieve the highest classification accuracies whilst those of online algorithms are lower but fairly competitive. The online learning models, however, are much sparser, resulting in a substantial speed-up, in which the training time costs and model sizes of AVMs are smallest with orders of magnitude lower than those of the standard batch methods. More specifically, the LIBSVM outperforms the other approaches in most of datasets, on which its training phase finishes within the time limit (i.e., two hours), except for the *ijcnn1* data wherein its testing score is less accurate but very close to that of BSGD-M. The LLSVM achieves good results which are slightly lower than those of the state-of-the-art batch kernel algorithm. The method, however, does not support multiclass classification. These two batch algorithms – LIBSVM and LLSVM could not be trained within the allowable amount of time on large-scale datasets (e.g., *airlines*), thus are not scalable.

Furthermore, six online algorithms in general have significant advantages against the batch methods in computational efficiency, especially when running on large-scale datasets. Among these algorithms, the BSGD-M (Pegasos+merging) obtains the highest classification scores, but suffers from a high computational cost. This can be seen in almost all datasets, especially for the airlines dataset on which its learning exceeds the time limit. The slow training of BSGD-M is caused by the merging step with computational complexity $\mathcal{O}\left(B^2\right)$ (B is the budget size). By contrast, the BSGD-R (Pegasos+removal) runs faster than the merging approach, but suffers from very high inaccurate results due to its naive budget maintenance strategy, that simply discards the most redundant support vector which may contain important information.

In terms of predictive performance, our proposed methods outperform the recent advanced online learning algorithms – FOGD and NOGD in most scenarios. The AVM-based models are able to achieve slightly less accurate but fairly comparable results compared with those of the state-of-the-art LIBSVM algorithm. In terms of sparsity and speed, the AVMs are the fastest ones in the training and testing phases in all cases thanks to their remarkable smaller model sizes. The difference between the training speed of our AVMs and that of two approaches varies across datasets. The gap is more significant for datasets with higher dimensional feature spaces. This is expected because the procedure to compute random features for each data point of FOGD involves *sin* and *cos* operators which are costly. These facts indicate that our proposed online kernel learning algorithms are both efficient and effective in solving large-scale kernel classification problems. Thus we believe that the AVM is the fast alternative to the existing SVM solvers for large-scale classification tasks.

Finally, comparing two versions of AVMs, it can be seen that the discriminative performances of AVM with Logistic loss are better than those of AVM with Hinge loss in most of datasets. This is because the Logistic function is smoother than the Hinge function, whilst the Hinge loss encourages sparsity of the model. The AVM-Logit, however, contains additional exponential operators, resulting in worse training time.

Table 3: Classification performance of our AVMs and the baselines in batch mode. The notation $[\delta \mid S \mid B \mid D]$, next to the dataset name, denotes the diameter δ, the model size S of AVM-based models, the budget size B of budgeted algorithms, and the number of random features D of FOGD, respectively. The accuracy is reported in percent (%), the training time and testing time are in second. The best performance is in **bold**. It is noteworthy that the LLSVM does not support multiclass classification and we terminate all runs exceeding the limit of two hours, therefore some results are unavailable.

Dataset $[\delta \mid S \mid B \mid D]$	a9a $[7.0 \mid 135 \mid 1,000 \mid 4,000]$			w8a $[13.0 \mid 131 \mid 1,000 \mid 4,000]$		
Algorithm	Train	Test	Accuracy	Train	Test	Accuracy
LIBSVM	84.57	22.23	**84.92**	50.96	2.95	**99.06**
LLSVM	50.73	8.73	83.00	92.19	10.41	98.64
BSGD-M	232.59	2.88	84.76±0.16	264.70	5.16	98.17±0.07
BSGD-R	90.48	2.72	80.26±3.38	253.30	4.98	97.10±0.04
FOGD	15.99	2.87	81.15±5.05	32.16	3.55	97.92±0.38
NOGD	82.40	0.60	82.33±2.18	374.87	0.65	98.06±0.18
AVM-Hinge	**4.96**	**0.25**	83.55±0.50	**11.84**	**0.52**	96.87±0.28
AVM-Logit	5.35	**0.25**	83.83±0.34	12.54	**0.52**	96.96±0.00
Dataset $[\delta \mid S \mid B \mid D]$	cod-rna $[1.0 \mid 436 \mid 400 \mid 1,600]$			ijcnn1 $[1.0 \mid 500 \mid 1,000 \mid 4,000]$		
Algorithm	Train	Test	Accuracy	Train	Test	Accuracy
LIBSVM	114.90	85.34	**96.39**	38.63	11.17	97.35
LLSVM	20.17	19.38	94.16	40.62	54.22	96.99
BSGD-M	90.62	5.66	95.67±0.21	93.05	6.13	**97.69±0.11**
BSGD-R	19.31	5.48	66.83±0.11	41.70	7.07	90.90±0.18
FOGD	7.62	11.95	92.65±4.20	7.31	10.10	90.64±0.07
NOGD	9.81	3.24	91.83±3.35	21.58	3.68	90.43±1.22
AVM-Hinge	**6.52**	**2.69**	94.38±1.16	**6.47**	2.71	91.14±0.71
AVM-Logit	7.03	2.86	93.10±2.11	6.86	**2.67**	91.19±0.95
Dataset $[\delta \mid S \mid B \mid D]$	covtype $[3.0 \mid 59 \mid 400 \mid 1,600]$			poker $[12.0 \mid 393 \mid 1,000 \mid 4,000]$		
Algorithm	Train	Test	Accuracy	Train	Test	Accuracy
LIBSVM	–	–	–	40.03	932.58	**57.91**
LLSVM	–	–	–	–	–	–
BSGD-M	2,413.15	3.75	**72.26±0.16**	414.09	123.57	54.10±0.22
BSGD-R	418.68	3.02	61.09±1.69	35.76	102.84	52.14±1.05
FOGD	69.94	2.45	59.34±5.85	9.61	101.29	46.62±5.00
NOGD	679.50	0.76	68.20±2.96	118.54	36.84	54.65±0.27
AVM-Hinge	**60.27**	0.26	64.31±0.37	3.86	8.21	55.49±0.13
AVM-Logit	61.92	**0.22**	64.42±0.34	**3.36**	**7.54**	55.60±0.17
Dataset $[\delta \mid S \mid B \mid D]$	KDDCup99 $[3.0 \mid 115 \mid 200 \mid 400]$			airlines $[1.0 \mid 388 \mid 1,000 \mid 4,000]$		
Algorithm	Train	Test	Accuracy	Train	Test	Accuracy
LIBSVM	4,380.58	661.04	**99.91**	–	–	–
LLSVM	–	–	–	–	–	–
BSGD-M	2,680.58	21.25	99.73±0.00	–	–	–
BSGD-R	1,644.25	14.33	39.81±2.26	4,741.68	29.98	80.27±0.06
FOGD	706.20	22.73	99.75±0.11	1,085.73	861.52	80.37±0.21
NOGD	3,726.21	3.11	99.80±0.02	3,112.08	18.53	74.83±0.20
AVM-Hinge	**554.42**	**2.75**	99.82±0.05	**586.90**	6.55	**80.72±0.00**
AVM-Logit	576.76	2.80	99.72±0.06	642.23	**6.10**	**80.72±0.00**

9.4 Online Classification

The next experiment investigates the performance of the AVMs in online classification task where individual data point continuously come turn-by-turn in a stream. Here we also use eight datasets and two versions of our approach: AVM with Hinge loss (AVM-Hinge) and AVM with Logistic loss (AVM-Logit) which are used in batch classification setting (cf. Section 9.3).

Baselines. We recruit the two widely-used algorithms – Perceptron and OGD for regular online kernel classification without budget maintenance and 8 state-of-the-art budget online kernel learning methods as follows:

- Perceptron: the kernelized variant without budget of Perceptron algorithm (Freund and Schapire, 1999);

- OGD: the kernelized variant without budget of online gradient descent (Kivinen et al., 2004);

- RBP: a budgeted Perceptron algorithm using random support vector removal strategy (Cavallanti et al., 2007);

- Forgetron: a kernel-based Perceptron maintaining a fixed budget by discarding oldest support vectors (Dekel et al., 2005);

- Projectron: a Projectron algorithm using the projection strategy (Orabona et al., 2009);

- Projectron++: the aggressive version of Projectron algorithm (Orabona et al., 2009);

- BPAS: a budgeted variant of Passive-Aggressive algorithm with simple SV removal strategy (Wang and Vucetic, 2010);

- BOGD: a budgeted variant of online gradient descent algorithm using simple SV removal strategy (Zhao et al., 2012);

- FOGD and NOGD: described in Section 9.3.

Hyperparameters setting. For each method learning on each dataset, we follow the same hyperparameter setting which is optimized in the batch classification task. For time efficiency, we only include the fast algorithms FOGD, NOGD and AVMs for the experiments on large-scale datasets. The other methods would exceed the time limit when running on such data.

Results. Fig. 4 and Fig. 5 shows the relative performance convergence w.r.t classification error and computation cost of the AVMs in comparison with those of the baselines. Combining these two figures, we compare the average mistake rate and running time in Fig. 6. Table 4 reports the final average results in detailed numbers after the methods see all data samples. It is worthy to note that for the four biggest datasets (*KDDCup99*, *covtype*, *poker*, *airlines*) that consist of millions data points, we exclude the non-budgeted online learning algorithm because of their substantially expensive time costs. From these results, we can draw some observations as follows.

First of all, as can be seen from Fig. 4, there are three groups of algorithms that have different learning progresses in terms of classification mistake rate. The first group includes the BOGD, Projectron and Forgetron that have the error rates fluctuating at the beginning, but then being stable till the end. In the meantime, the rates of the models in the second group, including Perceptron, OGD, RBP, Projectron++ and BPAS, quickly saturate at a plateau after these methods see a few portions, i.e., one-tenth to two-tenth, of the data. By contrast, the last group includes the recent online learning approaches – FOGD, NOGD, and our proposed ones – AVM-Hinge, AVM-Logit, that regularly perform better as more data points come. Exceptionally, for the dataset *w8a*, the classification errors of the methods in the first group keep increasing after seeing four-tenth of the data, whilst those of the last group are unexpectedly worse.

Second, Fig. 6 plots average mistake rate against computational cost, which shows similar patterns as in the our first observation. In addition, it can be seen from Fig. 5 that all algorithms have normal learning pace in which the execution time is accumulated over the learning procedure. Only the Projectron++ is slow at the beginning but then performs faster after receiving more data.

According to final results summarized in Table 4, the budgeted online approaches show efficacies with substantially faster computation than the ones without budgets. This is more obvious for larger datasets wherein the execution time costs of our proposed models are several orders of magnitude lower than those of regular online algorithms. This is because the coverage scheme of AVMs impressively boost their model sparsities, e.g., using $\delta = 3$ resulting in 115 core points for dataset *KDDCup99* consisting of $4, 408, 589$ instances, and using $\delta = 1$ resulting in 388 core points for dataset *airlines* containing $5, 336, 471$ data samples.

For classification capability, the non-budgeted methods only surpass the budgeted ones for the smallest dataset, that is, the OGD obtains the best performance for *a9a* data. This again demonstrates the importance of exploring budget online kernel learning algorithms. Between the two non-budgeted algorithms, the OGD achieves considerably better error rates than the Perceptron. The method, however, must perform much more expensive updates, resulting in a significantly larger number of support vectors and significantly higher computational time costs. This represents the trade-off between classification accuracy and computational complexity of the OGD.

Furthermore, comparing the performance of different existing budgeted online kernel learning algorithms, the AVM-Hinge and AVM-Logit outperform others in both discriminative performance and computation efficiency for almost all datasets. In particular, the AVM-based methods achieve the best mistake rates – 5.61±0.17, 8.01±0.18, 43.85±0.09, 19.28±0.00 for the *cod-rna*, *ijcnn1*, *poker* and *airlines* data, that are, respectively, 27.5%, 17.5%, 2.4%, 8.8% lower than the error rates of the second best models – two recent approaches FOGD and NOGD. On the other hand, the computation costs of the AVMs are significantly lower with large margins of hundreds of percents for large-scale databases *covtype*, *poker*, and *airlines* as shown in Table 4.
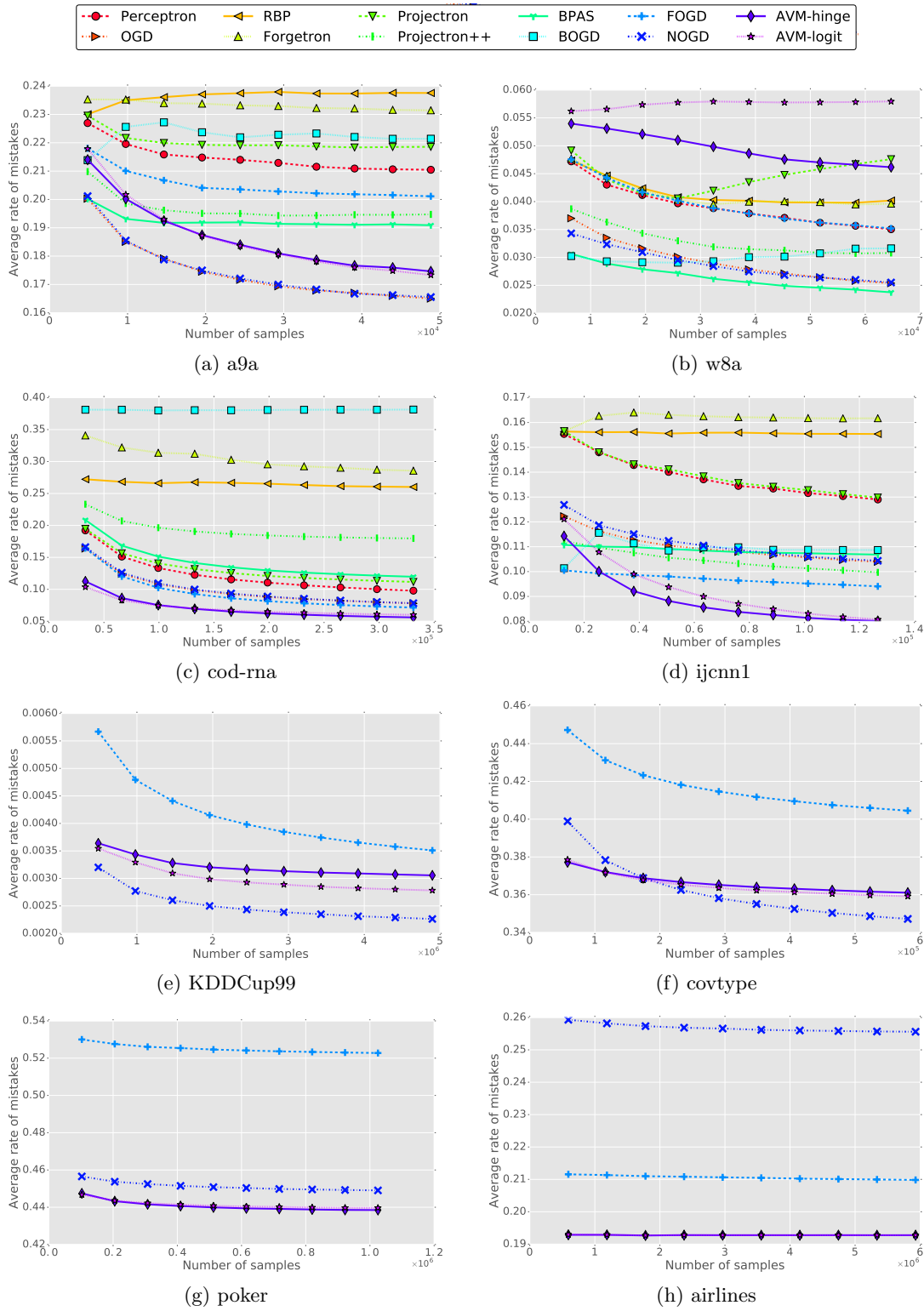
Figure 4: Convergence evaluation of online classification tasks: the average rate of mistakes as a function of the number of samples seen by the models. (Best viewed in colors).
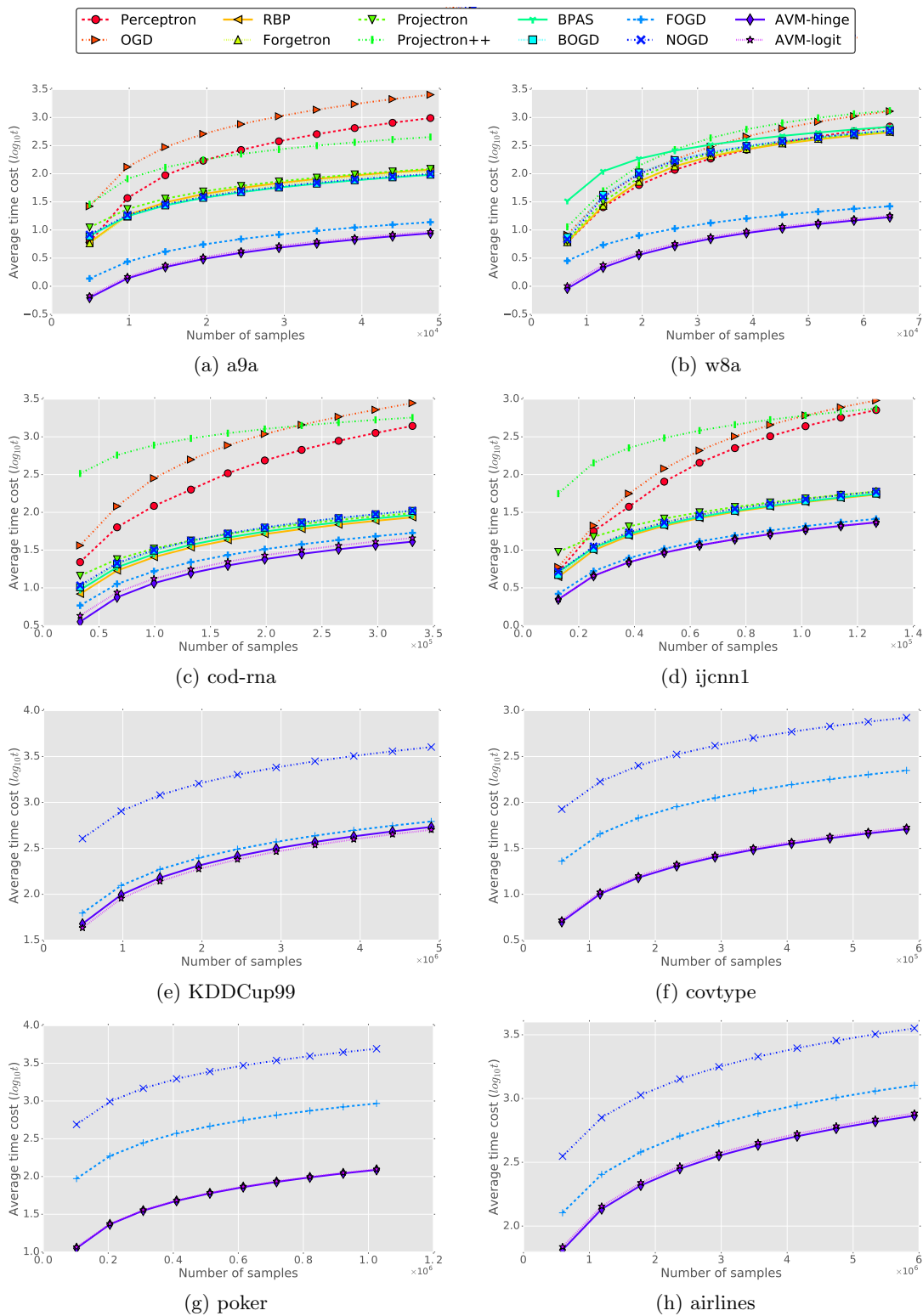
Figure 5: Convergence evaluation of online classification task: the average time costs (seconds shown in the logarithm with base 10) as a function of the number of samples seen by the models. (Best viewed in colors).
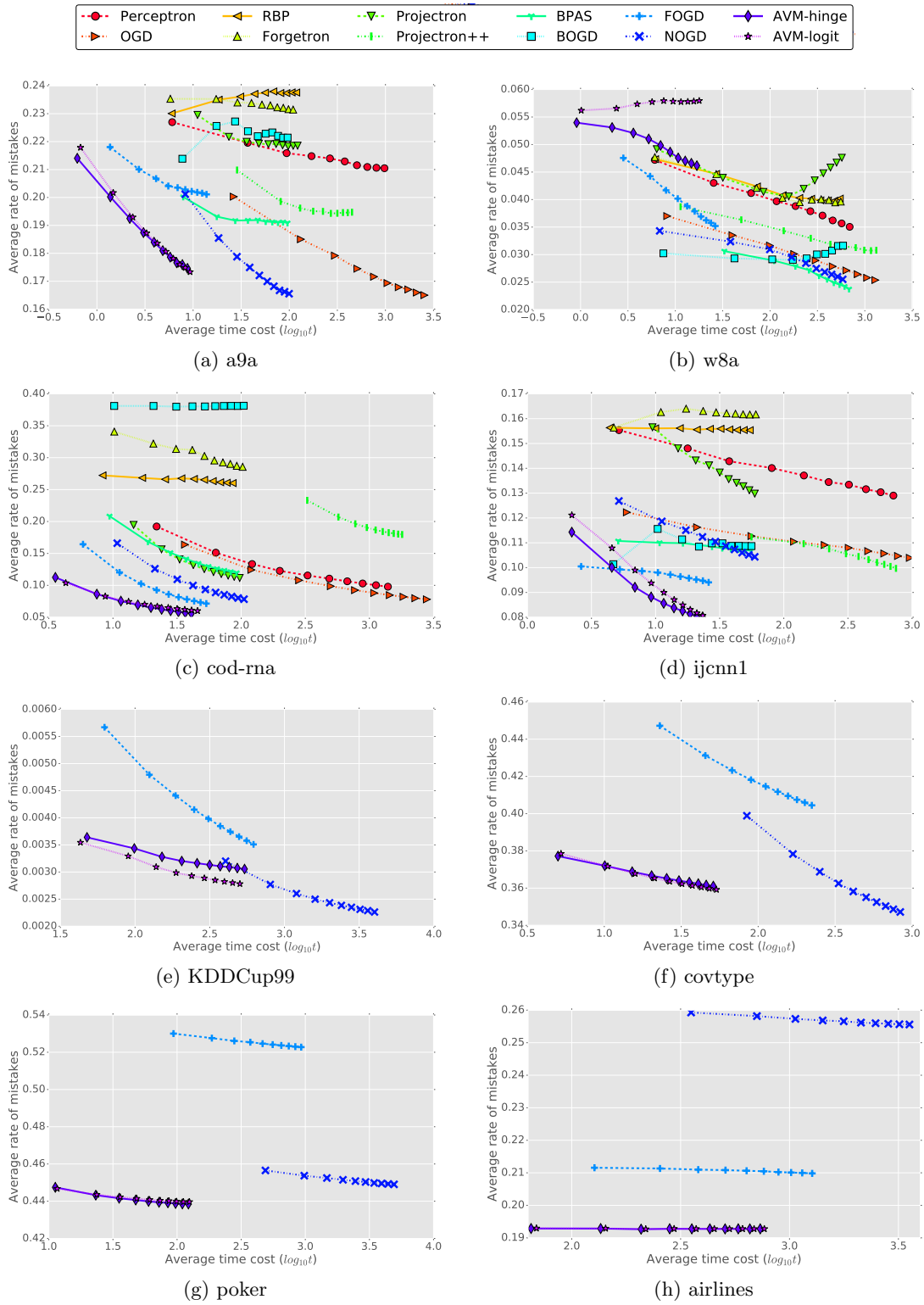
Figure 6: Average mistake rate vs. time cost for online classification. The average time (seconds) is shown in the logarithm with base 10. (Best viewed in colors).

Table 4: Classification performance of our proposed methods and the baselines in online mode. Note that δ, B and D are set to be the same as in batch classification tasks (cf., Section 9.3). The mistake rate is reported in percent (%) and the execution time is in second. The best performance is in **bold**.

Dataset [S]	a9a [142]		w8a [131]	
Algorithm	Mistake Rate	Time	Mistake Rate	Time
Perceptron	21.05±0.12	976.79	3.51±0.03	691.80
OGD	**16.50±0.06**	2,539.46	2.54±0.03	1,290.13
RBP	23.76±0.21	118.25	4.02±0.07	544.83
Forgetron	23.15±0.34	109.71	3.96±0.10	557.75
Projectron	21.86±1.73	122.08	4.76±1.13	572.20
Projectron++	19.47±2.22	449.20	3.08±0.63	1321.93
BPAS	19.09±0.17	95.81	**2.37±0.02**	681.46
BOGD	22.14±0.25	96.11	3.16±0.08	589.47
FOGD	20.11±0.10	13.79	3.52±0.05	26.40
NOGD	16.55±0.07	99.54	2.55±0.05	585.23
AVM-Hinge	17.46±0.12	**8.74**	4.62±0.78	**16.89**
AVM-Logit	17.33±0.16	9.31	5.80±0.02	17.86
Dataset [S]	cod-rna [436]		ijcnn1 [500]	
Algorithm	Mistake Rate	Time	Mistake Rate	Time
Perceptron	9.79±0.04	1,393.56	12.85±0.09	727.90
OGD	7.81±0.03	2,804.01	10.39±0.06	960.44
RBP	26.02±0.39	85.84	15.54±0.21	54.29
Forgetron	28.56±2.22	102.64	16.17±0.26	60.54
Projectron	11.16±3.61	97.38	12.98±0.23	59.37
Projectron++	17.97±15.60	1,799.93	9.97±0.09	749.70
BPAS	11.97±0.09	92.08	10.68±0.05	55.44
BOGD	38.13±0.11	104.60	10.87±0.18	55.99
FOGD	7.15±0.03	53.45	9.41±0.03	25.93
NOGD	7.83±0.06	105.18	10.43±0.08	59.36
AVM-Hinge	**5.61±0.17**	**40.89**	**8.01±0.18**	**23.26**
AVM-Logit	6.01±0.20	45.67	8.07±0.20	23.36
Dataset [S]	KDDCup99 [115]		covtype [59]	
Algorithm	Mistake rate	Time	Mistake rate	Time
FOGD	0.35±0.00	620.95	40.45±0.05	223.20
NOGD	**0.23±0.00**	4,009.03	**34.72±0.07**	838.47
AVM-Hinge	0.31±0.07	540.65	36.11±0.16	**51.12**
AVM-Logit	0.28±0.03	**503.34**	35.92±0.16	53.51
Dataset [S]	poker [393]		airlines [388]	
Algorithm	Mistake Rate	Time	Mistake Rate	Time
FOGD	52.28±0.04	928.89	20.98±0.01	1,270.75
NOGD	44.90±0.16	4,920.33	25.56±0.01	3,553.50
AVM-Hinge	**43.85±0.09**	**122.59**	**19.28±0.00**	**733.72**
AVM-Logit	43.97±0.07	124.86	**19.28±0.00**	766.19

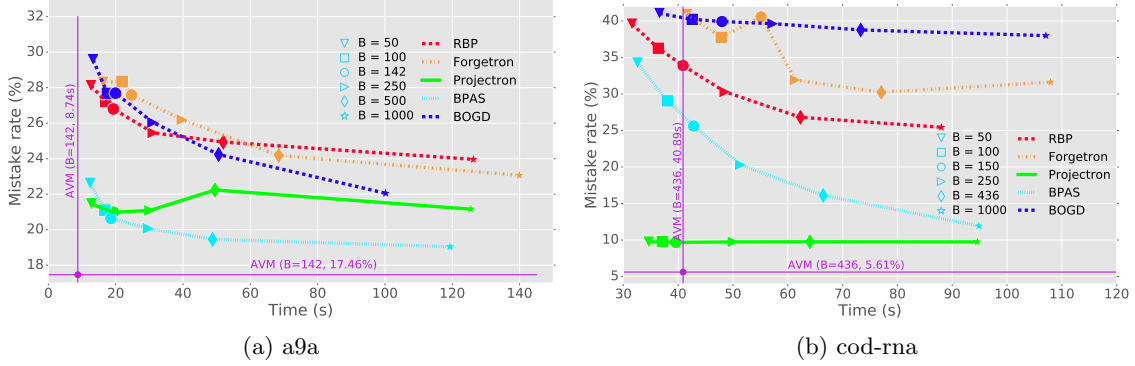(a) a9a   (b) cod-rna

Figure 7: Predictive and wall-clock performance on two datasets: *a9a* and *cod-rna* of budgeted methods when the budget size B is varied. (Best viewed in colors).

Table 5: Online regression results of 6 baselines and 3 versions of our AVMs. The notation $[\delta; S; B; D]$ denotes the same meanings as those in Table 3. The regression loss is measured using root mean squared error (RMSE) and the execution time is reported in second. The best performance is in **bold**.

| Dataset	casp		slice	
$[\delta \mid S \mid B \mid D]$	$[4.0 \mid 166 \mid 400 \mid 2,000]$		$[16.0 \mid 27 \mid 1,000 \mid 3,000]$	
Algorithm	RMSE	Time	RMSE	Time
RBP	0.3195±0.0012	7.15	0.1154±0.0006	810.14
Forgetron	0.3174±0.0008	10.14	0.1131±0.0004	1,069.15
Projectron	0.2688±0.0002	8.48	**0.0770±0.0002**	814.37
BOGD	0.2858±0.0002	6.20	0.1723±0.0001	816.16
FOGD	0.3775±0.0014	5.83	0.1440±0.0009	20.65
NOGD	**0.2512±0.0001**	6.99	0.0873±0.0002	812.69
AVM-ε	0.3165±0.0329	3.53	0.2013±0.0137	7.07
AVM-ℓ_1	0.3166±0.0330	3.44	0.2013±0.0138	7.13
AVM-ℓ_2	0.3274±0.0280	**3.31**	0.2590±0.0002	**6.88**
Dataset	year		airlines	
$[\delta \mid S \mid B \mid D]$	$[60.0 \mid 67 \mid 400 \mid 1,600]$		$[1.0 \mid 388 \mid 1,000 \mid 2,000]$	
Algorithm	RMSE	Time	RMSE	Time
RBP	0.1881±0.0002	605.42	36.5068±0.0010	3,418.89
Forgetron	0.1877±0.0004	904.09	36.5065±0.0003	5,774.47
Projectron	0.1390±0.0003	605.19	36.1365±0.0009	3,834.19
BOGD	0.2009±0.0000	596.10	35.7346±0.0010	3,058.96
FOGD	0.1581±0.0002	76.70	53.1638±0.0120	646.15
NOGD	0.1375±0.0005	607.37	**34.7421±0.0013**	3,324.38
AVM-ε	0.1286±0.0002	48.01	36.0901±0.0914	638.60
AVM-ℓ_1	**0.1232±0.0003**	47.29	36.3632±0.0192	**621.57**
AVM-ℓ_2	0.2420±0.0001	**46.63**	35.1128±0.0192	633.27
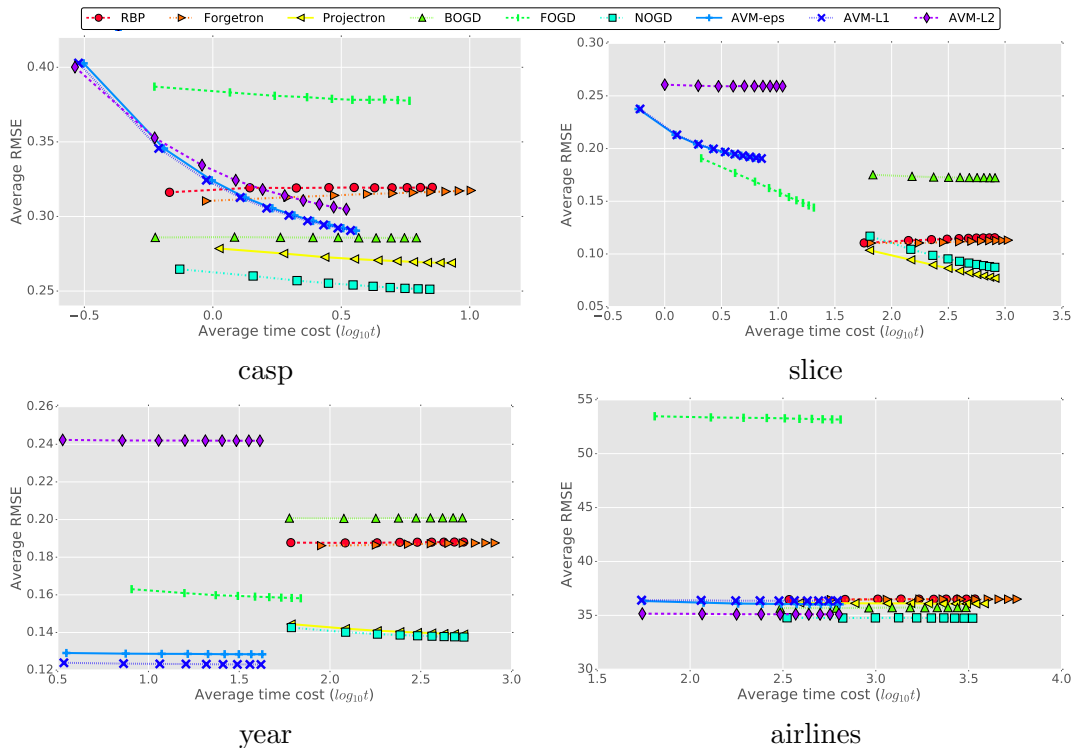
Figure 8: Average RMSE vs. time cost for online regression. The average time (seconds) is shown in the logarithm with base 10. (Best viewed in colors).

In all experiments, our proposed method produces the model sizes that are much smaller than the budget sizes of baseline methods. Thus we further investigate the performance of the budgeted baselines by varying the budget size B, and compare with our AVM with Hinge loss. Fig. 7 shows our analysis on two datasets *a9a* and *cod-rna*. It can be seen that the larger B helps model obtain better classification results, but hurts their running speed. For both datasets, the budgeted baselines with larger budget sizes still fail to beat the predictive performance of AVM. On the other hand, the baselines with smaller budget sizes run faster than the AVM on *cod-rna* dataset, but slower on *a9a* dataset.

Finally, two versions of AVMs demonstrate similar discriminative performances and computational complexities wherein the AVM-Logit is slightly slower due to the additional exponential operators as also seen in batch classification task. All aforementioned observations validate the effectiveness and efficiency of our proposed technique. Thus, we believe that our approximation machine is a promising technique for building scalable online kernel learning algorithms for large-scale classification tasks.

9.5 Online Regression

The last experiment addresses the online regression problem to evaluate the capabilities of our approach with three proposed loss functions – ℓ_1, ℓ_2 and ε-insensitive losses as described in Section 7. Incorporating these loss functions creates three versions: AVM-ε, AVM-ℓ_1 and AVM-ℓ_2. We use four datasets: *casp*, *slice*, *year* and *airlines* (delay minutes) with a wide

34

range of sizes for this task. We recruit six baselines: RBP, Forgetron, Projectron, BOGD, FOGD and NOGD (cf. more detailed description in Section 9.4).

Hyperparameters setting. We adopt the same hyperparameter searching procedure for batch classification task as in Section 9.3. Furthermore, for the budget size B and the feature dimension D in FOGD, we follow the same strategy used in Section 7.1.1 of Lu et al. (2015). More specifically, these hyperparameters are separately set for different datasets as reported in Table 5. They are chosen such that they are roughly proportional to the number of support vectors produced by the batch SVM algorithm in LIBSVM running on a small subset. The aim is to achieve competitive accuracy using a relatively larger budget size for tackling more challenging regression tasks.

Results. Fig. 9a and Fig. 9b shows the relative performance convergence w.r.t regression error (root mean square root - RMSE) and computation cost (seconds) of the AVMs in comparison with those of the baselines. Combining these two figures, we compare the average error and running time in Fig. 8. Table 5 reports the final average results in detailed numbers after the methods traverse all data samples. From these results, we can draw some observations as follows.

First of all, as can be seen from Fig. 9a, there are several different learning behaviors w.r.t regression loss, of the methods training on individual datasets. All algorithms, in general, reach their regression error plateaus very quickly as observed in the datasets *year* and *airlines* where they converge at certain points from the initiation of the learning. On the other hand, for *casp* and *slice* databases, the AVM-based models regularly obtain better performance, that is, their average RMSE scores keep reducing when receiving more data, except in *slice* data, the regression performance of AVM-ℓ_2 are almost unchanged during the learning. Note that, for these two datasets, the learning curve of AVM-ε coincides, thus is overplotted by that of AVM-ℓ_1, resulting in its no-show in the figure. Interestingly, the errors of RBP and Forgetron slightly increase throughout their online learning in these two cases.

Second, Fig. 8 plots average error against computational cost, which shows similar learning behaviors as in the our first observation. The computational cost progresses are simple and more obvious to comprehend than the regression progresses. As illustrated in Fig. 9b, all algorithms have nicely plausible execution time curves in which the time is accumulated over the learning procedure.

According to final results summarized in Table 5, our proposed models enjoy a significant advantage in computational efficacy whilst achieve better (for *year* dataset) or competitive regression results with other methods. The AVM, again, secures the best performance in terms of model sparsity. Among the baselines, the FOGD is the fastest, that is, its time costs can be considered to compare with those of our methods, but its regression performances are worse. The remaining algorithms usually obtain better results, but is traded off by the sacrifice of scalability. This, once again, verifies the effectiveness and efficiency of our proposed techniques. We believe that the AVM is a promising machine to perform online regression task for large-scale datasets.
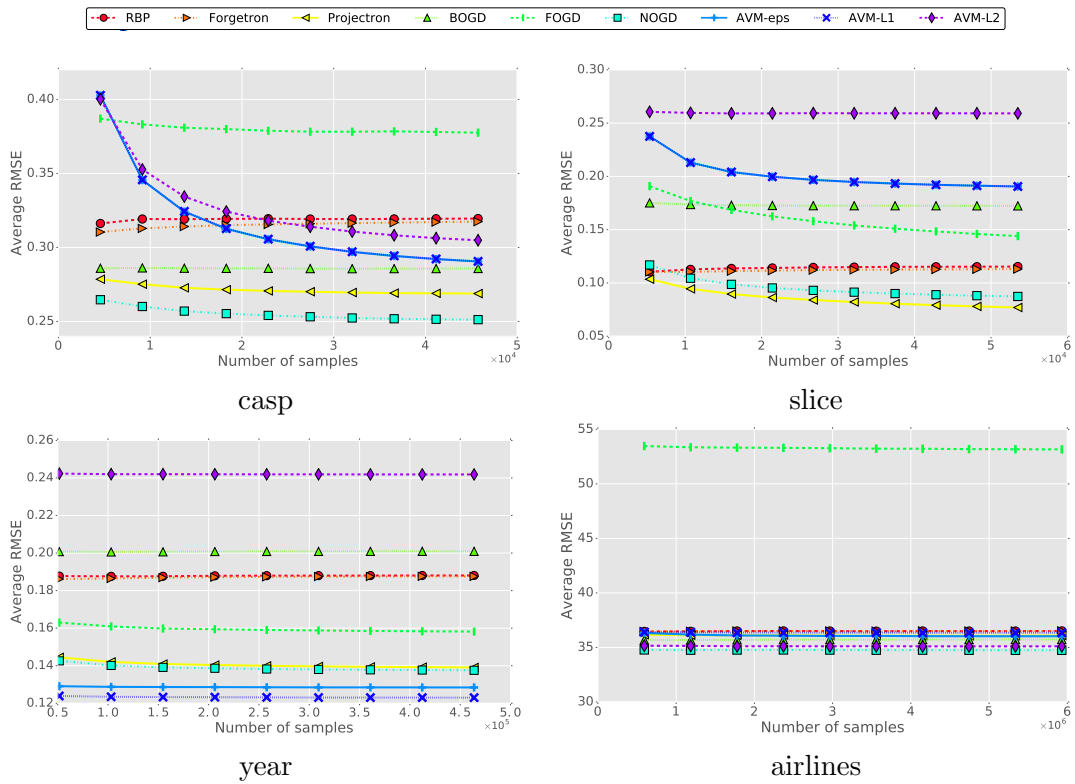
(a) The average RMSE as a function of the number of samples seen by the models.
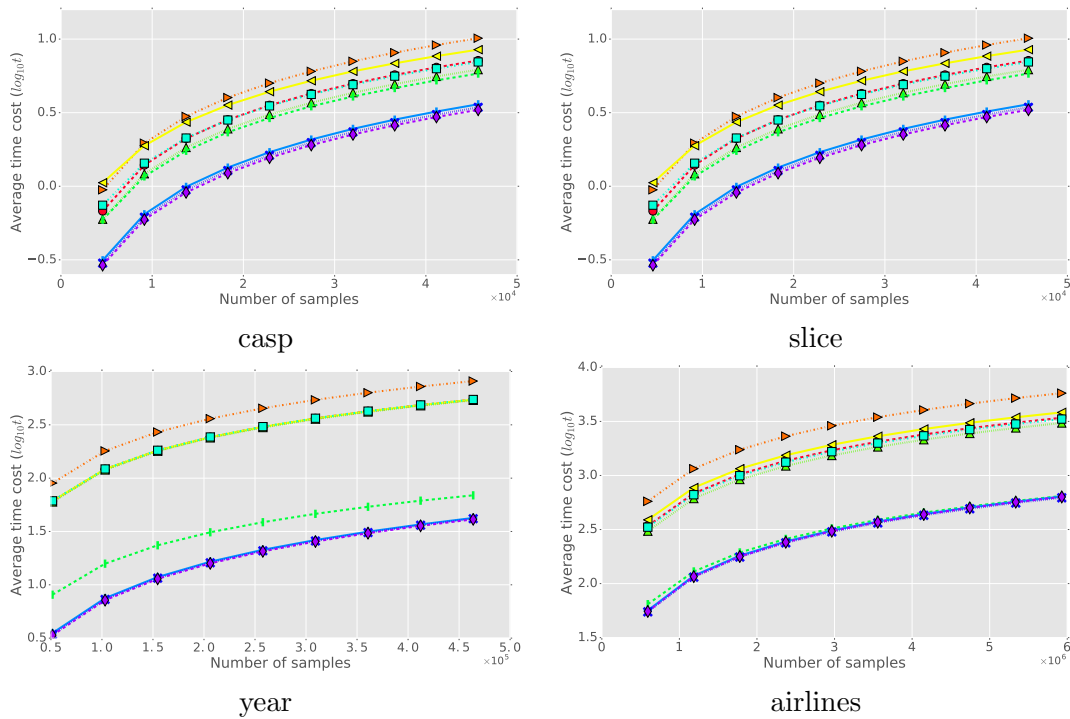
(b) The average time costs (seconds in logarithm of 10) as a function of the number of samples seen by the models.

Figure 9: Convergence evaluation of online regresion. (Best viewed in colors).

Finally, comparing the capability of three AVM's variants, all models demonstrate similar computational complexities wherein the AVM-ℓ_2 is slightly faster due to its simpler operator in computing the gradient as derived in Section 7. However, its regression errors are higher than two other methods – AVM-ε and AVM-ℓ_1.

10. Conclusion

In this paper, we have proposed Approximation Vector Machine (AVM) for large-scale online learning. The AVM is theoretically proven to have bounded and sparse model size while not hurting the predictive performance. We have validated our proposed method on several benchmark datasets. The experimental results show that the proposed AVM obtains a comparable predictive performance while simultaneously achieving an impressive model size and a computational speed-up compared with those of the baselines. Our future works are to apply AVM to the context of semi-supervised learning, anomaly detection, and support vector clustering.

Acknowledgment

11. Proofs Regarding δ-Coverage

Proof of Theorem 4

Assume that $\left\|x - x^{'}\right\| \leq \delta$, then we have

$$\left\|\Phi\left(x\right) - \Phi\left(x^{'}\right)\right\|^2 = K\left(x, x\right) + K\left(x^{'}, x^{'}\right) - 2K\left(x, x^{'}\right) = 2\left(1 - k\left(\left\|x - x^{'}\right\|^2\right)\right)$$

$$\leq 2\left(1 - k\left(\delta^2\right)\right) = \delta_\Phi^2$$

Furthermore, we have

$$\lim_{\delta \to 0} \delta_\Phi = 2^{1/2} \lim_{\delta \to 0}\left(1 - k\left(\delta^2\right)\right)^{1/2} = 2^{1/2}\left(1 - k\left(0\right)\right)^{1/2} = 0$$

Finally, since Gaussian kernel is a special radial kernel with $k\left(t\right) = \exp\left(-\gamma t\right)$, we obtain the final conclusion. ∎

Proof of Theorem 19

Since the proof is similar for the hyperrectangle cell case, we present the proof for the hypersphere case. Let us consider the open coverage $\mathcal{U} = \left\{\mathcal{B}\left(z, \frac{\delta}{2}\right)\right\}_{z \in \mathcal{X}}$. From the compactness of the data domain $\mathcal{X}$, it apparent that from $\mathcal{U}$ we must be able to extract a finite subcoverage of size m, that is, $\mathcal{U}_m = \left\{\mathcal{B}\left(z_i, \frac{\delta}{2}\right)\right\}_{i=1}^{m} \subset \mathcal{U}$. From the construction of the coverage $\mathcal{P}$ in Algorithm 3, we know that

$$\|c_i - c_j\| > \delta/2 \text{ if } i \neq j$$

Hence, each open sphere in the finite coverage $\mathcal{U}_m$ is able to contain at most one core point of $\mathcal{P}$. It means that the cardinality of $\mathcal{P}$ must be less than or equal m, that is, $|\mathcal{P}| \leq m$. ∎

Proofs Regarding Convergence Analysis

Given a finite δ-coverage $\mathcal{P} = (P_i)_{i \in I}$ with the core set $\mathcal{C} = (c_i)_{i \in I}$, when receiving an incoming instance (x_t, y_t) we approximate (x_t, y_t) by (c_{i_t}, y_t) with c_{i_t} is a core point whose cell contains x_t, that is, $x_t \in P_{i_t}$. We use a Bernoulli random variable Z_t to control if the approximation is performed or not, that is, $Z_t = 1$ indicates the approximation is performed.

Let us define $g_t = \lambda \mathbf{w}_t + l'(\mathbf{w}_t; x_t, y_t) = \lambda \mathbf{w}_t + \alpha_t \Phi(x_t)$. We have the following

$$h_t = g_t + Z_t \Delta_t$$

where $\Delta_t = \alpha_t (\Phi(c_{i_t}) - \Phi(x_t))$.

The update rule becomes

$$\mathbf{w}_{t+1} = \prod_S (\mathbf{w}_t - \eta_t h_t)$$

where $S = \mathbb{R}^D$ (i.e., the feature space) or $\mathcal{B}\left(\mathbf{0}, y_{\max} \lambda^{-1/2}\right)$.

Lemma 21. *The following statements hold*

i) There exist two positive constants P and M such that $\mathbb{E}\left[\|\mathbf{w}_t\|^2\right] \leq P^2$ and $\mathbb{E}\left[\alpha_t^2\right] \leq M$ for all t.

ii) $\mathbb{E}\left[\left\|l'(\mathbf{w}_t; x_t, y_t)\right\|^2\right] \leq L = \left(A\sqrt{P} + B\right)^2$ for all t.

iii) $\mathbb{E}\left[\|g_t\|^2\right] \leq G = \left(\lambda P + A\sqrt{P} + B\right)^2$ for all t.

iv) $\mathbb{E}\left[\|h_t\|^2\right] \leq H = \left(\sqrt{G} + \delta_\Phi\left(A\sqrt{P} + B\right)\right)^2$ for all t.

Proof i) We prove by induction that $\mathbb{E}\left[\|\mathbf{w}_t\|^2\right] \leq P^2$ where $P = \left(\frac{(\delta_\Phi+1)A + \sqrt{(\delta_\Phi+1)^2 A^2 + 4B\lambda(\delta_\Phi+1)}}{2\lambda}\right)^2$ for all t. Assume that the claim is holding for t, using Minkowski inequality, we have

$$\sqrt{\mathbb{E}\left[\|\mathbf{w}_{t+1}\|^2\right]} \leq \sqrt{\mathbb{E}\left[\left\|\prod_S (\mathbf{w}_t - \eta_t h_t)\right\|^2\right]} \leq \sqrt{\mathbb{E}\left[\|\mathbf{w}_t - \eta_t h_t\|^2\right]}$$

$$\leq \frac{t-1}{t}\sqrt{\mathbb{E}\left[\|\mathbf{w}_t\|^2\right]} + \eta_t \sqrt{\mathbb{E}\left[\|l'(\mathbf{w}_t; x_t, y_t)\|^2\right]} + \eta_t \sqrt{\mathbb{E}\left[\|\Delta_t\|^2\right]}$$

$$\leq \frac{t-1}{t}\sqrt{\mathbb{E}\left[\|\mathbf{w}_t\|^2\right]} + \frac{A\sqrt{\mathbb{E}[\|\mathbf{w}_t\|]} + B}{\lambda t} + \frac{\delta_\Phi \sqrt{\mathbb{E}\left[\alpha_t^2\right]}}{\lambda t}$$

$$\leq \frac{(t-1)P}{t} + \frac{A\sqrt{P} + B}{\lambda t} + \frac{\delta_\Phi \sqrt{\mathbb{E}\left[\|l'(\mathbf{w}_t; x_t, y_t)\|^2\right]}}{\lambda t}$$

$$\leq \frac{(t-1)P}{t} + \frac{(\delta_\Phi + 1)\left(A\sqrt{P} + B\right)}{\lambda t} = P$$

Note that we have used $\left\| l'\left(\mathbf{w}_t; x_t, y_t\right)\right\|^2 = \alpha_t^2 K\left(x_t, x_t\right) = \alpha_t^2$, $\mathbb{E}\left[\|\mathbf{w}_t\|\right] \le \sqrt{\mathbb{E}\left[\|\mathbf{w}_t\|^2\right]} \le$ P, and $u = P^{1/2} = \frac{(\delta_\Phi+1)A+\sqrt{(\delta_\Phi+1)^2 A^2 + 4B\lambda(\delta_\Phi+1)}}{2\lambda}$ is the solution of the quadratic equation

$$u^2 - \frac{\left(\delta_\Phi + 1\right)Au}{\lambda} - \frac{\left(\delta_\Phi + 1\right)B}{\lambda} = 0$$

The proof of $\mathbb{E}\left[\alpha_t^2\right] \le M$ is trivial for the case of Hinge, ℓ_1, Logistic, $\varepsilon-$insensitive losses. In these cases, we simply choose $M = \max\left(y_{\max}, 1\right)^2$. We only need to consider the ℓ_2-loss case. In particular, we have

$$\alpha_t^2 = \left(\mathbf{w}_t^\mathsf{T}\Phi\left(x_t\right) - y_t\right)^2 \le 2\left(\left(\mathbf{w}_t^\mathsf{T}\Phi\left(x_t\right)\right)^2 + y_{\max}^2\right)$$
$$\le 2\left(\|\mathbf{w}_t\|^2 \|\Phi\left(x_t\right)\|^2 + y_{\max}^2\right) \le 2\left(\|\mathbf{w}_t\|^2 + y_{\max}^2\right)$$

$$\mathbb{E}\left[\alpha_t^2\right] \le 2\left(P^2 + y_{\max}^2\right) = M$$

ii) We have the following

$$\sqrt{\mathbb{E}\left[\|l'\left(\mathbf{w}_t; x_t, y_t\right)\|^2\right]} \le \sqrt{\mathbb{E}\left[\left(A\|\mathbf{w}_t\|^{1/2} + B\right)^2\right]} \le A\sqrt{\mathbb{E}\left[\|\mathbf{w}_t\|\right]} + B \le A\sqrt{P} + B$$

Note that we have used the inequality $\mathbb{E}\left[\|\mathbf{w}_t\|\right] \le \sqrt{\mathbb{E}\left[\|\mathbf{w}_t\|^2\right]} \le P$.

iii) Using Minkowski inequality, we yield

$$\sqrt{\mathbb{E}\left[\|g_t\|^2\right]} \le \lambda\sqrt{\mathbb{E}\left[\|\mathbf{w}_t\|^2\right]} + \sqrt{\mathbb{E}\left[\|l'\left(\mathbf{w}_t; x_t, y_t\right)\|^2\right]} \le \lambda P + A\sqrt{P} + B$$

iv) We have the following

$$\sqrt{\mathbb{E}\left[\|h_t\|^2\right]} \le \sqrt{\mathbb{E}\left[\|g_t\|^2\right]} + \delta_\Phi\sqrt{\mathbb{E}\left[\alpha_t^2\right]}$$
$$\le \sqrt{G} + \delta_\Phi\sqrt{\mathbb{E}\left[\|l'\left(\mathbf{w}_t; x_t, y_t\right)\|^2\right]} = \sqrt{G} + \delta_\Phi\left(A\sqrt{P} + B\right)$$

$\blacksquare$

Lemma 22. *There exists a positive constant W such that $\mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right] \le W$ for all t.*

Proof We first remind the definitions of the relevant quantities

$$g_t = \lambda\mathbf{w}_t + l'\left(\mathbf{w}_t; x_t, y_t\right) = \lambda\mathbf{w}_t + \alpha_t\Phi\left(x_t\right)$$
$$h_t = g_t + Z_t\Delta_t \text{ where } \Delta_t = \alpha_t\left(\Phi\left(c_{i_t}\right) - \Phi\left(x_t\right)\right)$$

We now prove by induction in t. We derive as follows

$$\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2 = \left\|\prod_S (\mathbf{w}_t - \eta_t h_t) - \mathbf{w}^*\right\|^2 \leq \|\mathbf{w}_t - \eta_t h_t - \mathbf{w}^*\|^2$$
$$= \|\mathbf{w}_t - \mathbf{w}^*\| + \eta_t^2 \|h_t\|^2 - 2\eta_t \langle \mathbf{w}_t - \mathbf{w}^*, g_t + Z_t \Delta_t \rangle$$

where $S = \mathbb{R}^D$ or $\mathcal{B}\left(\mathbf{0}, y_{\max} \lambda^{-1/2}\right)$.

Taking conditional expectation w.r.t $\mathbf{w}_t$, we gain

$$\mathbb{E}\left[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2\right] \leq \mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right]$$
$$+ \eta_t^2 \mathbb{E}\left[\|h_t\|^2\right] - 2\eta_t \left\langle \mathbf{w}_t - \mathbf{w}^*, f'(\mathbf{w}_t)\right\rangle - 2\eta_t \langle \mathbf{w}_t - \mathbf{w}^*, Z_t \Delta_t \rangle$$
$$\leq \mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right] - 2\eta_t \lambda \|\mathbf{w}_t - \mathbf{w}^*\|^2 + \eta_t^2 \mathbb{E}\left[\|h_t\|^2\right] - 2\eta_t \langle \mathbf{w}_t - \mathbf{w}^*, Z_t \Delta_t \rangle$$

Here we note that we have used $\left\langle \mathbf{w}_t - \mathbf{w}^*, f'(\mathbf{w}_t)\right\rangle \geq \lambda \|\mathbf{w}_t - \mathbf{w}^*\|^2$. It comes from the following derivation

$$f(\mathbf{w}^*) - f(\mathbf{w}_t) \geq \left\langle f'(\mathbf{w}_t), \mathbf{w}^* - \mathbf{w}_t\right\rangle + \frac{\lambda}{2} \|\mathbf{w}_t - \mathbf{w}^*\|^2$$

$$\left\langle f'(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^*\right\rangle \geq f(\mathbf{w}_t) - f(\mathbf{w}^*) + \frac{\lambda}{2} \|\mathbf{w}_t - \mathbf{w}^*\|^2 \geq \left\langle f'(\mathbf{w}^*), \mathbf{w}_t - \mathbf{w}^*\right\rangle + \lambda \|\mathbf{w}_t - \mathbf{w}^*\|^2$$
$$\geq \lambda \|\mathbf{w}_t - \mathbf{w}^*\|^2 \qquad \text{thanks to } \left\langle f'(\mathbf{w}^*), \mathbf{w}_t - \mathbf{w}^*\right\rangle \geq 0$$

Taking expectation again, we gain

$$\mathbb{E}\left[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2\right] \leq \frac{t-2}{t} \mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right] + \frac{H}{\lambda^2 t^2} + \frac{2W^{1/2} M^{1/2} \delta_\Phi}{\lambda t}$$

Choosing $W = \left(\frac{M^{1/2} \delta_\phi + \left(M \delta_\phi^2 + 2H\right)^{1/2}}{2\lambda}\right)^2$, we gain if $\mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right] \leq W$ then $\mathbb{E}\left[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2\right] \leq$

W. The reason is that $W = \left(\frac{M^{1/2} \delta_\phi + \left(M \delta_\phi^2 + 2H\right)^{1/2}}{2\lambda}\right)^2$ is the solution of the equation

$$W = \frac{t-2}{t} W + \frac{H}{\lambda^2 t} + \frac{2W^{1/2} M^{1/2} \delta_\Phi}{\lambda t} \text{ or } 2W - \frac{2W^{1/2} M^{1/2} \delta_\Phi}{\lambda} - \frac{H}{\lambda^2} = 0$$

. Hence, if $\mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right] \leq W$, we arrive at

$$\mathbb{E}\left[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2\right] \leq \frac{t-2}{t} W + \frac{H}{\lambda^2 t} + \frac{2W^{1/2} M^{1/2} \delta_\Phi}{\lambda t} = W$$

$\blacksquare$

We now show the proof of Theorem 5.

Proof of Theorem 5

We first remind the definitions of the relevant quantities

$$g_t = \lambda \mathbf{w}_t + l'(\mathbf{w}_t; x_t, y_t) = \lambda \mathbf{w}_t + \alpha_t \Phi(x_t)$$
$$h_t = g_t + Z_t \Delta_t \text{ where } \Delta_t = \alpha_t (\Phi(c_{i_t}) - \Phi(x_t))$$

We then derive as follows

$$\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2 = \left\| \prod_S (\mathbf{w}_t - \eta_t h_t) - \mathbf{w}^* \right\|^2 \leq \|\mathbf{w}_t - \eta_t h_t - \mathbf{w}^*\|^2$$
$$= \|\mathbf{w}_t - \mathbf{w}^*\| + \eta_t^2 \|h_t\|^2 - 2\eta_t \langle \mathbf{w}_t - \mathbf{w}^*, g_t + Z_t \Delta_t \rangle$$

where $S = \mathbb{R}^D$ or $\mathcal{B}\left(\mathbf{0}, y_{\max}\lambda^{-1/2}\right)$.

$$\langle \mathbf{w}_t - \mathbf{w}^*, g_t \rangle \leq \frac{\|\mathbf{w}_t - \mathbf{w}^*\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2}{2\eta_t} + \frac{\eta_t}{2} \|h_t\|^2 - \langle \mathbf{w}_t - \mathbf{w}^*, Z_t \Delta_t \rangle$$

Taking conditional expectation w.r.t $\mathbf{w}_t$, we obtain

$$\left\langle \mathbf{w}_t - \mathbf{w}^*, f'(\mathbf{w}_t) \right\rangle \leq \frac{\mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right] - \mathbb{E}\left[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2\right]}{2\eta_t}$$
$$+ \frac{\eta_t}{2} \mathbb{E}\left[\|h_t\|^2\right] - \langle \mathbf{w}_t - \mathbf{w}^*, \mathbb{E}[Z_t \Delta_t] \rangle$$

$$f(\mathbf{w}_t) - f(\mathbf{w}^*) + \frac{\lambda}{2} \|\mathbf{w}_t - \mathbf{w}^*\|^2 \leq \frac{\mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right] - \mathbb{E}\left[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2\right]}{2\eta_t}$$
$$+ \frac{\eta_t}{2} \mathbb{E}\left[\|h_t\|^2\right] - \langle \mathbf{w}_t - \mathbf{w}^*, \mathbb{E}[Z_t \Delta_t] \rangle$$

Taking expectation again, we achieve

$$\mathbb{E}[f(\mathbf{w}_t) - f(\mathbf{w}^*)] \leq \frac{\lambda}{2}(t-1) \mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right] - \frac{\lambda}{2} t \mathbb{E}\left[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2\right]$$
$$+ \frac{\eta_t}{2} \mathbb{E}\left[\|h_t\|^2\right] - \mathbb{E}[\langle \mathbf{w}_t - \mathbf{w}^*, Z_t \Delta_t \rangle]$$

i) If Z_t is independent with $\mathbf{w}_t$, we derive as

$$
\begin{aligned}
\mathbb{E}\left[f\left(\mathbf{w}_t\right)-f\left(\mathbf{w}^*\right)\right] \leq{} & \frac{\lambda}{2}(t-1)\,\mathbb{E}\left[\left\|\mathbf{w}_t-\mathbf{w}^*\right\|^2\right]-\frac{\lambda}{2}t\mathbb{E}\left[\left\|\mathbf{w}_{t+1}-\mathbf{w}^*\right\|^2\right] \\
& +\frac{\eta_t}{2}\mathbb{E}\left[\left\|h_t\right\|^2\right]-\mathbb{E}\left[\left\langle Z_t\left(\mathbf{w}_t-\mathbf{w}^*\right),\Delta_t\right\rangle\right] \\
\leq{} & \frac{\lambda}{2}(t-1)\,\mathbb{E}\left[\left\|\mathbf{w}_t-\mathbf{w}^*\right\|^2\right]-\frac{\lambda}{2}t\mathbb{E}\left[\left\|\mathbf{w}_{t+1}-\mathbf{w}^*\right\|^2\right]+\frac{\eta_t}{2}\mathbb{E}\left[\left\|h_t\right\|^2\right] \\
& +\mathbb{E}\left[Z_t^2\left\|\mathbf{w}_t-\mathbf{w}^*\right\|^2\right]^{1/2}\mathbb{E}\left[\left\|\Delta_t\right\|^2\right]^{1/2} \\
\leq{} & \frac{\lambda}{2}(t-1)\,\mathbb{E}\left[\left\|\mathbf{w}_t-\mathbf{w}^*\right\|^2\right]-\frac{\lambda}{2}t\mathbb{E}\left[\left\|\mathbf{w}_{t+1}-\mathbf{w}^*\right\|^2\right]+\frac{\eta_t}{2}\mathbb{E}\left[\left\|h_t\right\|^2\right] \\
& +\mathbb{E}\left[Z_t^2\right]^{1/2}\mathbb{E}\left[\left\|\mathbf{w}_t-\mathbf{w}^*\right\|^2\right]^{1/2}\mathbb{E}\left[\left\|\Delta_t\right\|^2\right]^{1/2} \\
\leq{} & \frac{\lambda}{2}(t-1)\,\mathbb{E}\left[\left\|\mathbf{w}_t-\mathbf{w}^*\right\|^2\right]-\frac{\lambda}{2}t\mathbb{E}\left[\left\|\mathbf{w}_{t+1}-\mathbf{w}^*\right\|^2\right]+\frac{\eta_t}{2}H \qquad (5)\\
& +\mathbb{P}\left(Z_t=1\right)^{1/2}\mathbb{E}\left[\left\|\mathbf{w}_t-\mathbf{w}^*\right\|^2\right]^{1/2}\mathbb{E}\left[\left\|\Delta_t\right\|^2\right]^{1/2}
\end{aligned}
$$

Taking sum over $1,2,\ldots,T$ and using the inequalities in Lemmas 21 and 22, we yield

$$
\sum_{t=1}^T\mathbb{E}\left[f\left(\mathbf{w}_t\right)\right]-Tf\left(\mathbf{w}^*\right)\leq\frac{H}{2\lambda}\sum_{t=1}^T\frac{1}{t}+\sum_{t=1}^T\mathbb{P}\left(Z_t=1\right)^{1/2}\mathbb{E}\left[\left\|\mathbf{w}_t-\mathbf{w}^*\right\|^2\right]^{1/2}\mathbb{E}\left[\left\|\Delta_t\right\|^2\right]^{1/2}\quad(6)
$$

$$
T\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T\right)-f\left(\mathbf{w}^*\right)\right]\leq\frac{H}{2\lambda}\sum_{t=1}^T\frac{1}{t}+\sum_{t=1}^T\mathbb{P}\left(Z_t=1\right)^{1/2}\mathbb{E}\left[\left\|\mathbf{w}_t-\mathbf{w}^*\right\|^2\right]^{1/2}\mathbb{E}\left[\left\|\Delta_t\right\|^2\right]^{1/2}
$$

$$
\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T\right)-f\left(\mathbf{w}^*\right)\right]\leq\frac{H\left(\log\left(T\right)+1\right)}{2\lambda T}+\frac{\delta_\Phi M^{1/2}W^{1/2}}{T}\sum_{t=1}^T\mathbb{P}\left(Z_t=1\right)^{1/2}
$$

ii) If Z_t is independent with $\mathbf{w}_t$ and (x_t,y_t), we derive as

$$
\begin{aligned}
\mathbb{E}\left[f\left(\mathbf{w}_t\right)-f\left(\mathbf{w}^*\right)\right]\leq{} & \frac{\lambda}{2}(t-1)\,\mathbb{E}\left[\left\|\mathbf{w}_t-\mathbf{w}^*\right\|^2\right]-\frac{\lambda}{2}t\mathbb{E}\left[\left\|\mathbf{w}_{t+1}-\mathbf{w}^*\right\|^2\right] \\
& +\frac{\eta_t}{2}H-\mathbb{E}\left[Z_t\left\langle\mathbf{w}_t-\mathbf{w}^*,\Delta_t\right\rangle\right] \\
\leq{} & \frac{\lambda}{2}(t-1)\,\mathbb{E}\left[\left\|\mathbf{w}_t-\mathbf{w}^*\right\|^2\right]-\frac{\lambda}{2}t\mathbb{E}\left[\left\|\mathbf{w}_{t+1}-\mathbf{w}^*\right\|^2\right] \\
& +\frac{\eta_t}{2}H-\mathbb{E}\left[Z_t\right]\mathbb{E}\left[\left\langle\mathbf{w}_t-\mathbf{w}^*,\Delta_t\right\rangle\right] \\
\leq{} & \frac{\lambda}{2}(t-1)\,\mathbb{E}\left[\left\|\mathbf{w}_t-\mathbf{w}^*\right\|^2\right]-\frac{\lambda}{2}t\mathbb{E}\left[\left\|\mathbf{w}_{t+1}-\mathbf{w}^*\right\|^2\right]+\frac{\eta_t}{2}H \qquad(7)\\
& +\mathbb{P}\left(Z_t=1\right)\mathbb{E}\left[\left\|\mathbf{w}_t-\mathbf{w}^*\right\|^2\right]^{1/2}\mathbb{E}\left[\left\|\Delta_t\right\|^2\right]^{1/2}
\end{aligned}
$$

Taking sum over $1,2,\ldots,T$ and using the inequalities in Lemmas 21 and 22, we yield

$$\sum_{t=1}^{T} \mathbb{E}\left[f\left(\mathbf{w}_t\right)\right] - Tf\left(\mathbf{w}^*\right) \leq \frac{H}{2\lambda} \sum_{t=1}^{T} \frac{1}{t} + \sum_{t=1}^{T} P\left(Z_t = 1\right) \mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right]^{1/2} \mathbb{E}\left[\|\Delta_t\|^2\right]^{1/2} \quad (8)$$

$$T\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{H}{2\lambda} \sum_{t=1}^{T} \frac{1}{t} + \sum_{t=1}^{T} P\left(Z_t = 1\right) \mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right]^{1/2} \mathbb{E}\left[\|\Delta_t\|^2\right]^{1/2}$$

$$\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{H\left(\log\left(T\right) + 1\right)}{2\lambda T} + \frac{\delta_\Phi M^{1/2} W^{1/2}}{T} \sum_{t=1}^{T} P\left(Z_t = 1\right)$$

$$\leq \frac{H\left(\log\left(T\right) + 1\right)}{2\lambda T} + \delta_\Phi M^{1/2} W^{1/2}$$

iii) In general, we derive as

$$\mathbb{E}\left[f\left(\mathbf{w}_t\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{\lambda}{2}\left(t - 1\right)\mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right] - \frac{\lambda}{2}t\mathbb{E}\left[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2\right]$$

$$+ \frac{\eta_t}{2}H - \mathbb{E}\left[Z_t \langle \mathbf{w}_t - \mathbf{w}^*, \Delta_t \rangle\right]$$

$$\leq \frac{\lambda}{2}\left(t - 1\right)\mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right] - \frac{\lambda}{2}t\mathbb{E}\left[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2\right] \quad (9)$$

$$+ \frac{\eta_t}{2}H + \mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right]^{1/2} \mathbb{E}\left[\|\Delta_t\|^2\right]^{1/2}$$

Taking sum over $1, 2, \ldots, T$ and using the inequalities in Lemmas 21 and 22, we yield

$$\sum_{t=1}^{T} \mathbb{E}\left[f\left(\mathbf{w}_t\right)\right] - Tf\left(\mathbf{w}^*\right) \leq \frac{H}{2\lambda} \sum_{t=1}^{T} \frac{1}{t} + \sum_{t=1}^{T} \mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right]^{1/2} \mathbb{E}\left[\|\Delta_t\|^2\right]^{1/2} \quad (10)$$

$$T\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{H}{2\lambda} \sum_{t=1}^{T} \frac{1}{t} + \sum_{t=1}^{T} \mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right]^{1/2} \mathbb{E}\left[\|\Delta_t\|^2\right]^{1/2}$$

$$\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{H\left(\log\left(T\right) + 1\right)}{2\lambda T} + \delta_\Phi M^{1/2} W^{1/2}$$

■

Proof of Theorem 15

Let us denote the model size, i.e., the number of vectors in support set, after the iteration t by S_t. We also define N_t by the binary random variable which specifies whether the incoming instance (x_t, y_t) locates in a new cell of the coverage, that is, $N_t = 1$ indicating the current cell P_{i_t} is a new cell. We assume that Z_t is independent with (x_t, y_t) and so does with N_t. Since a new instance is added to the support set if either a new cell is discovered or the old cell is found but approximation is not performed, we reach the following

$$S_t \leq S_{t-1} + N_t + \left(1 - Z_t\right)\left(1 - N_t\right)$$

Taking expectation, we obtain

$$\mathbb{E}\left[S_t\right] \leq \mathbb{E}\left[S_{t-1}\right] + \mathbb{E}\left[N_t\right] + \left(1 - \mathbb{E}\left[Z_t\right]\right)\left(1 - \mathbb{E}\left[N_t\right]\right)$$
$$\leq \mathbb{E}\left[S_{t-1}\right] + \mathbb{E}\left[N_t\right] + \left(1 - p_t\right)\left(1 - \mathbb{E}\left[N_t\right]\right)$$
$$\leq \mathbb{E}\left[S_{t-1}\right] + \mathbb{E}\left[N_t\right] + q_t\left(1 - \mathbb{E}\left[N_t\right]\right)$$

$$\mathbb{E}\left[S_t\right] - \mathbb{E}\left[S_{t-1}\right] \leq \mathbb{E}\left[N_t\right] + q_t\left(1 - \mathbb{E}\left[N_t\right]\right)$$

Summing over the above when $t = 1, \ldots, T$, we have

$$\mathbb{E}\left[S_T\right] \leq \sum_{t=1}^{T} \mathbb{E}\left[N_t\right] + \sum_{t=1}^{T} q_t\left(1 - \mathbb{E}\left[N_t\right]\right) = \sum_{t=1}^{T} q_t + \sum_{t=1}^{T} p_t \mathbb{E}\left[N_t\right]$$
$$\leq \sum_{t=1}^{T} q_t + \sum_{t=1}^{T} \mathbb{E}\left[N_t\right] \leq \sum_{t=1}^{T} q_t + \mathbb{E}\left[M_T\right] \tag{11}$$

where we have denoted $\mathbb{P}\left(Z_t = 1\right) = p_t$, $\mathbb{P}\left(Z_t = 0\right) = q_t$, and $M_T = \sum_{t=1}^{T} N_t$ indicates the number of cells discovered so far.

We consider some specific cases and investigate the model size $\mathbb{E}\left[S_T\right]$ in these cases.

i) $p_t = \mathbb{P}\left(Z_t = 1\right) = 1, \forall t$, that is, we always do approximation. From Eq. (11), we obtain

$$\mathbb{E}\left[S_T\right] \leq \mathbb{E}\left[M_T\right] \leq |\mathcal{P}|$$

ii) $p_t = \mathbb{P}\left(Z_t = 1\right) = \max\left(0, 1 - \frac{\beta}{t}\right), \forall t$. It follows that

$$q_t = 1 - p_t \leq 1 - \left(1 - \frac{\beta}{t}\right) = \frac{\beta}{t}$$

From Eq. (11), we gain

$$\mathbb{E}\left[S_T\right] \leq \beta \sum_{t=1}^{T} \frac{1}{t} + \mathbb{E}\left[M_T\right] \leq \beta\left(1 + \int_1^T \frac{1}{t} dt\right) + \mathbb{E}\left[M_T\right]$$
$$\leq \beta\left(\log T + 1\right) + \mathbb{E}\left[M_T\right]$$

iii) $p_t = \mathbb{P}\left(Z_t = 1\right) = \max\left(0, 1 - \frac{\beta}{t^\rho}\right), \forall t$ where $0 < \rho < 1$. It follows that

$$q_t = 1 - p_t \leq 1 - \left(1 - \frac{\beta}{t^\rho}\right) = \frac{\beta}{t^\rho}$$

From Eq. (11), we gain

$$\mathbb{E}\left[S_T\right] \leq \beta \sum_{t=1}^{T} \frac{1}{t^\rho} + \mathbb{E}\left[M_T\right] \leq \beta\left(1 + \int_1^T t^{-\rho} dt\right) + \mathbb{E}\left[M_T\right] \leq \frac{\beta T^{1-\rho}}{1-\rho} + \mathbb{E}\left[M_T\right]$$

iv) $p_t = \mathbb{P}\left(Z_t = 1\right) = \max\left(0, 1 - \frac{\beta}{t^\rho}\right), \forall t$ where $\rho > 1$. It follows that

$$q_t = 1 - p_t \leq 1 - \left(1 - \frac{\beta}{t^\rho}\right) = \frac{\beta}{t^\rho}$$

From Eq. (11), we gain

$$\mathbb{E}\left[S_T\right] \leq \beta \sum_{t=1}^{T} \frac{1}{t^\rho} + \mathbb{E}\left[M_T\right] \leq \beta\zeta\left(\rho\right) + \mathbb{E}\left[M_T\right] \leq \beta\zeta\left(\rho\right) + |\mathcal{P}|$$

where $\zeta\left(.\right)$ is ζ- Riemann function defined by the integral $\zeta\left(s\right) = \frac{1}{\Gamma(s)}\int_0^{+\infty}\frac{t^{s-1}}{e^s-1}dt.$ ∎

We now show the proof of Theorem 8. To realize this proof, we use the famous inequality, namely Hoeffding which for completeness we state below.

Theorem. *(Hoeffding inequality) Let the independent variables $X_1, \ldots, X_n$ where $a_i \leq X_i \leq b_i$ for each $i \in [n]$. Let $S = \sum_{i=1}^{n} X_i$ and $\Delta_i = b_i - a_i$. The following hold*
i) $\mathbb{P}\left(S - \mathbb{E}\left[S\right] > \varepsilon\right) \leq \exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^{n}\Delta_i^2}\right).$
ii) $\mathbb{P}\left(|S - \mathbb{E}\left[S\right]| > \varepsilon\right) \leq 2\exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^{n}\Delta_i^2}\right)$

Proof of Theorem 8

From Eqs. (6, 8, 10), we achieve

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[f\left(\mathbf{w}_t\right)\right] - f\left(\mathbf{w}^*\right) \leq \frac{H\left(\log\left(T\right)+1\right)}{2\lambda T} + d_T$$

Let us denote $X = f\left(\mathbf{w}_r\right) - f\left(\mathbf{w}^*\right)$, where r is uniformly sampled from $\{1, 2, \ldots, T\}$. We have

$$\mathbb{E}_r\left[X\right] = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[f\left(\mathbf{w}_t\right)\right] - f\left(\mathbf{w}^*\right) \leq \frac{H\left(\log\left(T\right)+1\right)}{2\lambda T} + d_T$$

It follows that

$$\mathbb{E}\left[X\right] = \mathbb{E}_{(x_t,y_t)_{t=1}^T}\left[\mathbb{E}_r\left[X\right]\right] \leq \frac{H\left(\log\left(T\right)+1\right)}{2\lambda T} + d_T$$

Let us denote $\Delta_T = \max_{1\leq t \leq T}\left(f\left(\mathbf{w}_t\right) - f\left(\mathbf{w}^*\right)\right)$ which implies that $0 < f\left(\mathbf{w}_r\right) - f\left(\mathbf{w}^*\right) < \Delta_T$. Applying Hoeffding inequality for the random variable X, we gain

$$\mathbb{P}\left(X - \mathbb{E}\left[X\right] > \varepsilon\right) \leq \exp\left(-\frac{2\varepsilon^2}{\Delta_T^2}\right)$$

$$\mathbb{P}\left(X - \frac{H\left(\log\left(T\right)+1\right)}{2\lambda T} - d_T > \varepsilon\right) \leq \exp\left(-\frac{2\varepsilon^2}{\Delta_T^2}\right)$$

$$\mathbb{P}\left(X \leq \frac{H\left(\log\left(T\right)+1\right)}{2\lambda T} + d_T + \varepsilon\right) > 1 - \exp\left(-\frac{2\varepsilon^2}{\Delta_T^2}\right)$$

Choosing $\delta = \exp\left(-\frac{2\varepsilon^2}{\Delta_T^2}\right)$ or $\varepsilon = \Delta_T\sqrt{\frac{1}{2}\log\frac{1}{\delta}}$, then with the probability at least $1 - \delta$, we have

$$f\left(\mathbf{w}_r\right) - f\left(\mathbf{w}^*\right) \leq \frac{H\left(\log\left(T\right)+1\right)}{2\lambda T} + d_T + \Delta_T\sqrt{\frac{1}{2}\log\frac{1}{\delta}}$$

46

■

Proof of Theorem 9

We denote $W_T^\alpha = \mathbb{E}\left[\left\|\mathbf{w}_{(1-\alpha)T+1} - \mathbf{w}^*\right\|^2\right]$. Our proof proceeds as follows.

i) If Z_t is independent with $\mathbf{w}_t$, taking sum in Eq. (5) when $t = (1-\alpha)T + 1, \ldots, T$, we gain

$$\sum_{t=(1-\alpha)T+1}^{T} \mathbb{E}\left[f\left(\mathbf{w}_t\right)\right] - \alpha T f\left(\mathbf{w}^*\right) \leq \frac{\lambda\left(1-\alpha\right)T}{2} W_T^\alpha + \frac{H}{2\lambda} \sum_{t=(1-\alpha)T+1}^{T} \frac{1}{t}$$

$$+ \sum_{t=1}^{T} \mathbb{P}\left(Z_t = 1\right)^{1/2} \mathbb{E}\left[\left\|\mathbf{w}_t - \mathbf{w}^*\right\|^2\right]^{1/2} \mathbb{E}\left[\left\|\Delta_t\right\|^2\right]^{1/2} \quad (12)$$

$$\leq \frac{\lambda\left(1-\alpha\right)T}{2} W_T^\alpha + \frac{H\log\left(1/\left(1-\alpha\right)\right)}{2\lambda}$$

$$+ \delta_\Phi M^{1/2} W^{1/2} \sum_{t=(1-\alpha)T+1}^{T} \mathbb{P}\left(Z_t = 1\right)^{1/2}$$

where we have used the inequality $\sum_{t=(1-\alpha)T+1}^{T} \frac{1}{t} \leq \log\left(1/\left(1-\alpha\right)\right)$.

$$\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T^\alpha\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{\lambda\left(1-\alpha\right)}{2\alpha} W_T^\alpha + \frac{\delta_\Phi M^{1/2} W^{1/2}}{\alpha T} \sum_{t=(1-\alpha)T+1}^{T} \mathbb{P}\left(Z_t = 1\right)^{1/2}$$

$$+ \frac{H\log\left(1/\left(1-\alpha\right)\right)}{2\lambda\alpha T}$$

ii) If Z_t is independent with $\mathbf{w}_t$ and (x_t, y_t), taking sum in Eq. (7) when $t = (1-\alpha)T + 1, \ldots, T$, we gain

$$\sum_{t=(1-\alpha)T+1}^{T} \mathbb{E}\left[f\left(\mathbf{w}_t\right)\right] - \alpha T f\left(\mathbf{w}^*\right) \leq \frac{\lambda\left(1-\alpha\right)T}{2} W_T^\alpha + \frac{H}{2\lambda} \sum_{t=(1-\alpha)T+1}^{T} \frac{1}{t}$$

$$+ \sum_{t=(1-\alpha)T+1}^{T} \mathbb{P}\left(Z_t = 1\right) \mathbb{E}\left[\left\|\mathbf{w}_t - \mathbf{w}^*\right\|^2\right]^{1/2} \mathbb{E}\left[\left\|\Delta_t\right\|^2\right]^{1/2}$$

$$\qquad (13)$$

$$\leq \frac{\lambda\left(1-\alpha\right)T}{2} W_T^\alpha + \frac{H\log\left(1/\left(1-\alpha\right)\right)}{2\lambda}$$

$$+ \delta_\Phi M^{1/2} W^{1/2} \sum_{t=(1-\alpha)T+1}^{T} \mathbb{P}\left(Z_t = 1\right)$$

$$\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T^\alpha\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{\lambda\left(1-\alpha\right)}{2\alpha} W_T^\alpha + \frac{\delta_\Phi M^{1/2} W^{1/2}}{\alpha T} \sum_{t=(1-\alpha)T+1}^{T} \mathbb{P}\left(Z_t = 1\right)$$

$$+ \frac{H\log\left(1/\left(1-\alpha\right)\right)}{2\lambda\alpha T}$$

iii) In general, taking sum in Eq. (9) when $t = (1 - \alpha) T + 1, \ldots, T$, we gain

$$\sum_{t=(1-\alpha)T+1}^{T} \mathbb{E}\left[f\left(\mathbf{w}_t\right)\right] - \alpha T f\left(\mathbf{w}^*\right) \leq \frac{\lambda\left(1-\alpha\right)T}{2} W_T^{\alpha} + \frac{H}{2\lambda} \sum_{t=(1-\alpha)T+1}^{T} \frac{1}{t}$$

$$+ \sum_{t=(1-\alpha)T+1}^{T} \mathbb{E}\left[\left\|\mathbf{w}_t - \mathbf{w}^*\right\|^2\right]^{1/2} \mathbb{E}\left[\left\|\Delta_t\right\|^2\right]^{1/2}$$

$$\leq \frac{\lambda\left(1-\alpha\right)T}{2} W_T^{\alpha} + \frac{H \log\left(1/\left(1-\alpha\right)\right)}{2\lambda} + \delta_\Phi M^{1/2} W^{1/2} \alpha T$$

$$(14)$$

$$\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T^{\alpha}\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{\lambda\left(1-\alpha\right)}{2\alpha} W_T^{\alpha} + \delta_\Phi M^{1/2} W^{1/2} + \frac{H \log\left(1/\left(1-\alpha\right)\right)}{2\lambda\alpha T}$$

$\blacksquare$

Proof of Theorem 10

The proof of this theorem is similar to that of Theorem 8 which relies on Hoeffding inequality.

From Eqs. (12, 13, 14), we achieve

$$\frac{1}{\alpha T} \sum_{t=(1-\alpha)T+1}^{T} \mathbb{E}\left[f\left(\mathbf{w}_t\right)\right] - f\left(\mathbf{w}^*\right) \leq \frac{H \log\left(1/\left(1-\alpha\right)\right)}{2\lambda\alpha T} + d_T$$

Let us denote $X = f\left(\mathbf{w}_r\right) - f\left(\mathbf{w}^*\right)$, where r is uniformly sampled from $\{(1 - \alpha) T + 1, 2, \ldots, T\}$. We have

$$\mathbb{E}_r\left[X\right] = \frac{1}{\alpha T} \sum_{t=(1-\alpha)T+1}^{T} \mathbb{E}\left[f\left(\mathbf{w}_t\right)\right] - f\left(\mathbf{w}^*\right) \leq \frac{H \log\left(1/\left(1-\alpha\right)\right)}{2\lambda\alpha T} + d_T$$

It follows that

$$\mathbb{E}\left[X\right] = \mathbb{E}_{(x_t, y_t)_{t=1}^{T}}\left[\mathbb{E}_r\left[X\right]\right] \leq \frac{H \log\left(1/\left(1-\alpha\right)\right)}{2\lambda\alpha T} + d_T$$

Let us denote $\Delta_T^{\alpha} = \max_{(1-\alpha)T+1\leq t\leq T}\left(f\left(\mathbf{w}_t\right) - f\left(\mathbf{w}^*\right)\right)$ which implies that $0 < f\left(\mathbf{w}_r\right) - f\left(\mathbf{w}^*\right) < \Delta_T^{\alpha}$. Applying Hoeffding inequality for the random variable X, we gain

$$\mathbb{P}\left(X - \mathbb{E}\left[X\right] > \varepsilon\right) \leq \exp\left(-\frac{2\varepsilon^2}{\left(\Delta_T^{\alpha}\right)^2}\right)$$

$$\mathbb{P}\left(X - \frac{H \log\left(1/\left(1-\alpha\right)\right)}{2\lambda\alpha T} - d_T > \varepsilon\right) \leq \exp\left(-\frac{2\varepsilon^2}{\left(\Delta_T^{\alpha}\right)^2}\right)$$

$$\mathbb{P}\left(X \leq \frac{H \log\left(1/\left(1-\alpha\right)\right)}{2\lambda\alpha T} + d_T + \varepsilon\right) > 1 - \exp\left(-\frac{2\varepsilon^2}{\left(\Delta_T^{\alpha}\right)^2}\right)$$

Choosing $\delta = \exp\left(-\frac{2\varepsilon^2}{(\Delta_T^\alpha)^2}\right)$ or $\varepsilon = \Delta_T^\alpha \sqrt{\frac{1}{2}\log\frac{1}{\delta}}$, then with the probability at least $1 - \delta$, we have

$$f\left(\mathbf{w}_r\right) - f\left(\mathbf{w}^*\right) \leq \frac{H\log\left(1/\left(1-\alpha\right)\right)}{2\lambda\alpha T} + d_T + \Delta_T^\alpha \sqrt{\frac{1}{2}\log\frac{1}{\delta}}$$

∎

Proof of Theorem 13

It is apparent that $f\left(\mathbf{w}\right)$ is L-strongly smooth w.r.t $\|.\|_2$. Therefore, we have

$$f\left(\mathbf{w}_r\right) - f\left(\mathbf{w}^*\right) \leq f'\left(\mathbf{w}^*\right)^\mathsf{T}\left(\mathbf{w}_r - \mathbf{w}^*\right) + \frac{L}{2}\|\mathbf{w}_r - \mathbf{w}^*\|^2 \leq \frac{L}{2}\|\mathbf{w}_r - \mathbf{w}^*\|^2$$

It follows that $\Delta_T^\alpha \leq \frac{1}{2}L M_T^\alpha$. Hence we gain the conclusion. ∎

12. Proofs of Bound for L2 Loss

We now consider the upper bound of $\|\mathbf{w}^*\|$ in the case that ℓ_2 loss is being used for regression problem. Concretely, we have the following theorem whose proof is similar to that of Theorem 1 in Shalev-Shwartz et al. (2007, 2011).

Theorem 23. *If* $\mathbf{w}^* = argmin_{\mathbf{w}} \left(\frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \mathbf{w}^\mathsf{T} \Phi \left(x_i \right) \right)^2 \right)$ *then* $\|\mathbf{w}^*\| \leq y_{max} \lambda^{-1/2}$.

Proof Let us consider the equivalent constrains optimization problem

$$
\min_{\mathbf{w}, \boldsymbol{\xi}} \left(\frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_{i=1}^{N} \xi_i^2 \right)
$$
$$
\text{s.t.: } \xi_i = y_i - \mathbf{w}^\mathsf{T} \Phi \left(x_i \right), \forall i
$$

The Lagrange function is of the following form

$$
\mathcal{L} \left(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha} \right) = \frac{\lambda}{2} \|\mathbf{w}^2\| + \frac{1}{N} \sum_{i=1}^{N} \xi_i^2 + \sum_{i=1}^{N} \alpha_i \left(y_i - \mathbf{w}^\mathsf{T} \Phi \left(x_i \right) - \xi_i \right)
$$

Setting the derivatives to 0, we gain

$$
\nabla_{\mathbf{w}} \mathcal{L} = \lambda \mathbf{w} - \sum_{i=1}^{N} \alpha_i \Phi \left(x_i \right) = 0 \rightarrow \mathbf{w} = \lambda^{-1} \sum_{i=1}^{N} \alpha_i \Phi \left(x_i \right)
$$

$$
\nabla_{\xi_i} \mathcal{L} = \frac{2}{N} \xi_i - \alpha_i = 0 \rightarrow \xi_i = \frac{N \alpha_i}{2}
$$

Substituting the above to the Lagrange function, we gain the dual form

$$
\mathcal{W} \left(\boldsymbol{\alpha} \right) = - \frac{\lambda}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^{N} y_i \alpha_i - \frac{N}{4} \sum_{i=1}^{N} \alpha_i^2
$$
$$
= - \frac{1}{2\lambda} \left\| \sum_{i=1}^{N} \alpha_i \Phi \left(x_i \right) \right\|^2 + \sum_{i=1}^{N} y_i \alpha_i - \frac{N}{4} \sum_{i=1}^{N} \alpha_i^2
$$

Let us denote $(\mathbf{w}^*, \boldsymbol{\xi}^*)$ and $\boldsymbol{\alpha}^*$ be the primal and dual solutions, respectively. Since the strong duality holds, we have

$$
\frac{\lambda}{2} \|\mathbf{w}^*\|^2 + \frac{1}{N} \sum_{i=1}^{N} \xi_i^{*2} = - \frac{\lambda}{2} \|\mathbf{w}^*\|^2 + \sum_{i=1}^{N} y_i \alpha_i^* - \frac{N}{4} \sum_{i=1}^{N} \alpha_i^{*2}
$$

$$
\lambda \|\mathbf{w}^*\|^2 = \sum_{i=1}^{N} y_i \alpha_i^* - \frac{N}{4} \sum_{i=1}^{N} \alpha_i^{*2} - \frac{1}{N} \sum_{i=1}^{N} \xi_i^{*2}
$$
$$
\leq \sum_{i=1}^{N} \left(y_i \alpha_i^* - \frac{N}{4} \alpha_i^{*2} \right) \leq \sum_{i=1}^{N} \frac{y_i^2}{N} \leq y_{\max}^2
$$

We note that we have used $g\left(\alpha_i^*\right) = y_i\alpha_i^* - \frac{N}{4}\alpha_i^{*2} \leq g\left(\frac{2y_i}{N}\right) = \frac{y_i^2}{N}$. Hence, we gain the conclusion. ∎

Lemma 24. *Assume that ℓ_2 loss is using, the following statement holds*

$$\|\mathbf{w}_{T+1}\| \leq \lambda^{-1}\left(y_{max} + \frac{1}{T}\sum_{t=1}^{T}\|\mathbf{w}_t\|\right)$$

where $y_{max} = \max_{y\in\mathcal{Y}}|y|$.

Proof We have the following

$$\mathbf{w}_{t+1} = \begin{cases} \prod_S\left(\frac{t-1}{t}\mathbf{w}_t - \eta_t\alpha_t\Phi\left(x_t\right)\right) & \text{if } Z_t = 0 \\ \prod_S\left(\frac{t-1}{t}\mathbf{w}_t - \eta_t\alpha_t\Phi\left(c_{i_t}\right)\right) & \text{otherwise} \end{cases}$$

It follows that

$$\|\mathbf{w}_{t+1}\| \leq \frac{t-1}{t}\|\mathbf{w}_t\| + \frac{1}{\lambda t}|\alpha_t| \quad \text{since } \|\Phi\left(x_t\right)\| = \|\Phi\left(c_{i_t}\right)\| = 1$$

It happens that $l'\left(\mathbf{w}_t; x_t, y_t\right) = \alpha_t\Phi\left(x_t\right)$. Hence, we gain

$$|\alpha_t| = \left|y_t - \mathbf{w}_t^{\mathsf{T}}\Phi\left(x_t\right)\right| \leq y_{max} + \|\mathbf{w}_t\|\,\|\Phi\left(x_t\right)\| \leq y_{max} + \|\mathbf{w}_t\|$$

It implies that

$$t\|\mathbf{w}_{t+1}\| \leq (t-1)\|\mathbf{w}_t\| + \lambda^{-1}\left(y_{max} + \|\mathbf{w}_t\|\right)$$

Taking sum when $t = 1, 2, \ldots, T$, we achieve

$$T\|\mathbf{w}_{T+1}\| \leq \lambda^{-1}\left(Ty_{max} + \sum_{t=1}^{T}\|\mathbf{w}_t\|\right)$$

$$\|\mathbf{w}_{T+1}\| \leq \lambda^{-1}\left(y_{max} + \frac{1}{T}\sum_{t=1}^{T}\|\mathbf{w}_t\|\right) \tag{15}$$

∎

Theorem 25. *If $\lambda > 1$ then $\|\mathbf{w}_{T+1}\| \leq \frac{y_{max}}{\lambda-1}\left(1 - \frac{1}{\lambda^T}\right) < \frac{y_{max}}{\lambda-1}$ for all T.*

Proof First we consider the sequence $\{s_T\}_T$ which is identified as $s_{T+1} = \lambda^{-1}\left(y_{max} + s_T\right)$ and $s_1 = 0$. It is easy to find the formula of this sequence as

$$s_{T+1} - \frac{y_{max}}{\lambda-1} = \lambda^{-1}\left(s_T - \frac{y_{max}}{\lambda-1}\right) = \ldots = \lambda^{-T}\left(s_1 - \frac{y_{max}}{\lambda-1}\right) = \frac{-\lambda^{-T}y_{max}}{\lambda-1}$$

$$s_{T+1} = \frac{y_{\max}}{\lambda - 1} \left(1 - \frac{1}{\lambda^T} \right)$$

We prove by induction by T that $\|\mathbf{w}_T\| \leq s_T$ for all T. It is obvious that $\|\mathbf{w}_1\| = s_1 = 0$. Assume that $\|\mathbf{w}_t\| \leq s_t$ for $t \leq T$, we verify it for $T + 1$. Indeed, we have

$$\|\mathbf{w}_{T+1}\| \leq \lambda^{-1} \left(y_{\max} + \frac{1}{T} \sum_{t=1}^{T} \|\mathbf{w}_t\| \right) \leq \lambda^{-1} \left(y_{\max} + \frac{1}{T} \sum_{t=1}^{T} s_t \right)$$

$$\leq \lambda^{-1} \left(y_{\max} + s_T \right) = s_{T+1}$$

∎

In addition, from Eq. (15) in case that $\lambda \leq 1$ we cannot bound $\|\mathbf{w}_{T+1}\|$. Concretely, we have the following theorem.

Theorem 26. *If $\{z_T\}_T$ is a sequence such that $z_{T+1} = \lambda^{-1} \left(y_{max} + \frac{1}{T} \sum_{t=1}^{T} z_t \right)$ with $z_1 = 0$ then in case that $\lambda \leq 1$ this sequence is not upper-bounded.*

Proof Let us denote $s_t = y_{\max}^{-1} z_t$. It is obvious that $s_1 = 0$ and $s_{T+1} = \lambda^{-1} \left(1 + \frac{1}{T} \sum_{t=1}^{T} s_t \right)$. We now prove by induction by T that

$$s_T \geq \sum_{t=1}^{T-1} \frac{1}{t} \text{ for all } T \geq 2$$

With $T = 2$, we have $s_2 = \lambda^{-1} \geq 1$. Assume that this holds for all $2 \leq t \leq T$, we verify it for $T + 1$.

$$s_{T+1} = \lambda^{-1} \left(1 + \frac{1}{T} \sum_{t=1}^{T} s_t \right) \geq 1 + \frac{1}{T} \sum_{t=1}^{T} s_t$$

$$\geq 1 + \frac{1}{T} \sum_{t=1}^{T} \sum_{n=1}^{t-1} \frac{1}{n} \geq 1 + \sum_{t=1}^{T-1} \frac{T-t}{Tt}$$

$$\geq 1 + \sum_{t=1}^{T} \left(\frac{1}{t} - \frac{1}{T} \right) = \sum_{t=1}^{T} \frac{1}{t}$$

The final conclusion is obvious from the fact $\sum_{t=1}^{T} \frac{1}{t} > \log(T+1)$ and hence can exceed any positive number. ∎

References

M. Badoiu and K. L. Clarkson. Optimal core-sets for balls. In *DIMACS Workshop on Computational Geometry*, 2002.

G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69(2-3):143–167, 2007.

C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Transaction Intelligence System Technology*, 2(3):27:1–27:27, May 2011. ISSN 2157-6904.

C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal Machine Learning Research*, 2:265–292, March 2002. ISSN 1532-4435.

K. Crammer, J. Kandola, and Y. Singer. Online classification on a budget. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2004.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal Machine Learning Research*, 7:551–585, 2006.

F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39:1–49, 2002.

O. Dekel, S. Shalev-Shwartz, and Y. Singer. The forgetron: A kernel-based perceptron on a fixed budget. In *Advances in Neural Information Processing Systems (NIPS)*, pages 259–266, 2005.

M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *International Conference on Machine Learning (ICML)*, ICML '08, pages 264–271, New York, NY, USA, 2008. ACM.

Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, December 1999.

J. Hensman, N. Fusi, and N. D Lawrence. Gaussian processes for big data. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, page 282. Citeseer, 2013.

T. Joachims. Advances in kernel methods. chapter Making Large-scale Support Vector Machine Learning Practical, pages 169–184. MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-19416-3.

J. Kivinen, A. J. Smola, and R. C. Williamson. Online Learning with Kernels. *IEEE Transactions on Signal Processing*, 52:2165–2176, August 2004.

T. Le, P. Duong, M. Dinh, D. T Nguyen, V. Nguyen, and D. Phung. Budgeted semi-supervised support vector machine. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, June 2016a.

T. Le, T. D. Nguyen, V. Nguyen, and D. Phung. Dual space gradient descent for online learning. In *Advances in Neural Information Processing (NIPS)*, pages 4583–4591, December 2016b.

T. Le, V. Nguyen, T. D. Nguyen, and Dinh Phung. Nonparametric budgeted stochastic gradient descent. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 654–572, May 2016c.

T. Le, K. Nguyen, V. Nguyen, D. T. Nguyen, and D. Phung. Gogp: Fast online regression with gaussian processes. In *International Conference on Data Mining (ICDM)*, 2017.

J. Lu, S. C.H. Hoi, J. Wang, P. Zhao, and Z.-Y. Liu. Large scale online kernel learning. *Journal Machine Learning Research*, 2015.

L. Ming, W. Shifeng, and Z. Changshui. On the sample complexity of random fourier features for online learning: How many random fourier features do we need? *ACM Transaction Knowledge Discovery Data*, 8(3):13:1–13:19, June 2014. ISSN 1556-4681.

F. Orabona, J. Keshet, and B. Caputo. Bounded kernel-based online learning. *Journal Machine Learning Research*, 10:2643–2666, December 2009. ISSN 1532-4435.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances In Neural Infomration Processing Systems (NIPS)*, 2007.

A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *International Conference on Machine Learning (ICML)*, pages 449–456, 2012.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

S. Shalev-Shwartz and N. Srebro. Svm optimization: inverse dependence on training set size. In *International Conference on Machine Learning (ICML)*, pages 928–935. ACM, 2008.

S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss. *Journal of Machine Learning Research*, 14(1):567–599, 2013.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *International Conference on Machine Learning (ICML)*, pages 807–814, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3.

S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

I. Steinwart. Sparseness of support vector machines. *Journal Machine Learning Research*, 4:1071–1105, December 2003. ISSN 1532-4435.

I. W. Tsang, J. T. Kwok, P. Cheung, and N. Cristianini. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

I. W. Tsang, A. Kocsor, and J. T. Kwok. Simpler core vector machines with enclosing balls. In *International Conference on Machine Learning (ICML)*, ICML '07, pages 911–918, 2007.

Z. Wang and S. Vucetic. Twin vector machines for online learning on a budget. In *SIAM International Conference on Data Mining (SDM)*, pages 906–917, 2009.

Z. Wang and S. Vucetic. Online passive-aggressive algorithms on a budget. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9, pages 908–915, 2010.

Z. Wang, K. Crammer, and S. Vucetic. Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training. *Journal of Machine Learning Research*, 13(1):3103–3131, 2012.

K. Zhang, L. Lan, Z. Wang, and F. Moerchen. Scaling up kernel svm on limited resources: A low-rank linearization approach. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1425–1434, 2012.

P. Zhao, J. Wang, P. Wu, R. Jin, and S. C. H. Hoi. Fast bounded online gradient descent algorithms for scalable kernel-based online learning. *CoRR*, 2012.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning (ICML)*, pages 928–936, 2003.