ASSET: APPROXIMATE STOCHASTIC SUBGRADIENT ESTIMATION TRAINING FOR SUPPORT VECTOR MACHINES

Sangkyun Lee[1] and Stephen J. Wright[2]
[1]*Computer Science Department, LS VIII, University of Technology, Dortmund, Germany*
[2]*Computer Sciences Department, University of Wisconsin, Madison, WI, U.S.A.*

Keywords: Stochastic approximation, Large-scale, Online learning, Support vector machines, Nonlinear kernels.

Abstract: Subgradient methods for training support vector machines have been quite successful for solving large-scale and online learning problems. However, they have been restricted to linear kernels and strongly convex formulations. This paper describes efficient subgradient approaches without such limitations, making use of randomized low-dimensional approximations to nonlinear kernels, and minimization of a reduced primal formulation using an algorithm based on robust stochastic approximation, which do not require strong convexity.

1 INTRODUCTION

The algorithms for training the support vector machines (SVMs) can be broadly categorized into (i) *decomposition methods* such as SVM-Light (Joachims, 1999) and LASVM (Bordes et al., 2005), (ii) *cutting-plane methods* for linear kernels (SVM-Perf (Joachims, 2006) and OCAS (Franc and Sonnenburg, 2008)) and for nonlinear kernels (CPNY (Joachims et al., 2009) and CPSP (Joachims and Yu, 2009)), and (iii) *subgradient methods* for linear kernels including Pegasos (Shalev-Shwartz et al., 2007) and SGD (Bottou, 2005). Subgradient methods are of particular interest, since they are well suited to large-scale and online learning problems.

This paper aims to provide practical subgradient algorithms for training SVMs with nonlinear kernels, overcoming the weakness of the updated Pegasos algorithm (Shalev-Shwartz et al., 2011) which uses exact kernel information and requires a dual variable for each training example in the worst case. Our approach uses a primal formulation with low-dimensional approximations to feature mappings. Such approximations are obtained either by approximating the Gram matrix or by constructing subspaces with random bases approximating the feature spaces induced by kernels. These approximations can be computed and applied to data points iteratively, and thus are suited to an online context. Further, we suggest an efficient way to make predictions for test points using the approximate feature mappings, without recovering the potentially large number of support vectors.

Unlike Pegasos, we use Vapnik's original SVM formulation without modifying the objective to be strongly convex. Our main algorithm takes steplengths of size $O(1/\sqrt{t})$ (associated with robust stochastic approximation methods (Nemirovski et al., 2009; Nemirovski and Yudin, 1983) and online convex programming (Zinkevich, 2003)), rather than the $O(1/t)$ steplength scheme in Pegasos. We see little practical difference between $O(1/\sqrt{t})$ steplengths and $O(1/t)$ steps.

2 NONLINEAR SVMS IN PRIMAL

We discuss the primal SVM formulation in a low-dimensional space induced by kernel approximation.

2.1 Structure of the Formulation

Let us consider the training point and label pairs $\{(\mathbf{t}_i, y_i)\}_{i=1}^m$ for $\mathbf{t}_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}$, and a feature mapping $\phi : \mathbb{R}^n \to \mathbb{R}^d$. Given a convex loss function $\ell(\cdot) : \mathbb{R} \to \mathbb{R} \cup \{\infty\}$ and $\lambda > 0$, the primal SVM problem (for classification) can be stated as follows :

$$\text{(P1)} \quad \min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \ \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{m}\sum_{i=1}^{m} \ell(y_i(\mathbf{w}^T\phi(\mathbf{t}_i) + b)).$$

By substituting the following into (P1):

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \phi(\mathbf{t}_i), \tag{1}$$

Lee S. and Wright S. (2012).
ASSET: APPROXIMATE STOCHASTIC SUBGRADIENT ESTIMATION TRAINING FOR SUPPORT VECTOR MACHINES.
In *Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods*, pages 223-228
DOI: 10.5220/0003786202230228

we obtain

$$\text{(P2)} \quad \min_{\alpha\in\mathbb{R}^m, b\in\mathbb{R}} \frac{\lambda}{2}\alpha^T\Psi\alpha + \frac{1}{m}\sum_{i=1}^{m}\ell\left(y_i(\Psi_{i\cdot}\alpha+b)\right),$$

where $\Psi\in\mathbb{R}^{m\times m}$ is defined by $\Psi_{ij} := \phi(\mathbf{t}_i)^T\phi(\mathbf{t}_j)$ for $i,j=1,2,\ldots,m$, and $\Psi_{i\cdot}$ denotes the i-th row of Ψ. Optimality conditions for (P2) are as follows:

$$\lambda\Psi\alpha + \frac{1}{m}\sum_{i=1}^{m}\beta_i y_i\Psi_{i\cdot}^T = 0, \quad \frac{1}{m}\sum_{i=1}^{m}\beta_i y_i = 0, \qquad (2)$$

$$\text{for some } \beta_i\in\partial\ell\left(y_i(\Psi_{i\cdot}\alpha+b)\right),\ i=1,2,\ldots,m.$$

Then we can derive the following result via convex analysis (see Lee and Wright (2011) for details).

Proposition 1. *Let $(\alpha,b)\in\mathbb{R}^m\times\mathbb{R}$ be a solution of (P2). Then if we define* $\mathbf{w}$ *by* (1)*, $(\mathbf{w},b)\in\mathbb{R}^d\times\mathbb{R}$ is a solution of (P1).*

Without loss of generality, (2) suggests that we can constrain α to have the form

$$\alpha_i = -\frac{y_i}{\lambda m}\beta_i.$$

These results clarify the connection between the expansion coefficient α and the dual variable β, which is introduced in Chapelle (2007) but not fully explicated there.

2.2 Reformulation with Approximations

Consider the feature mapping $\phi^\circ:\mathbb{R}^n\to\mathcal{H}$ to a Hilbert space $\mathcal{H}$ induced by a kernel $k^\circ:\mathbb{R}^n\times\mathbb{R}^n\to\mathbb{R}$ satisfying Mercer's Theorem. Suppose that we have an approximation $\phi:\mathbb{R}^n\to\mathbb{R}^d$ of ϕ° for which

$$k^\circ(\mathbf{s},\mathbf{t})\approx\phi(\mathbf{s})^T\phi(\mathbf{t}), \qquad (3)$$

for all inputs $\mathbf{s}$ and $\mathbf{t}$ of interest. If we construct a matrix $V\in\mathbb{R}^{m\times d}$ by defining its i-th row as

$$V_{i\cdot} = \phi(\mathbf{t}_i)^T, \ \ i=1,2,\ldots,m, \ \text{ then we have} \qquad (4)$$

$$\Psi := VV^T \approx \Psi^\circ := [k^\circ(\mathbf{t}_i,\mathbf{t}_j)]_{i,j=1,2,\ldots,m}. \qquad (5)$$

Ψ is a positive semidefinite rank-d approximation to Ψ°. Substituting $\Psi=VV^T$ and $\gamma := V^T\alpha$ in (P2) leads to the equivalent formulation

$$\text{(PL)} \quad \min_{\gamma\in\mathbb{R}^d, b\in\mathbb{R}} \frac{\lambda}{2}\gamma^T\gamma + \frac{1}{m}\sum_{i=1}^{m}\ell(y_i(V_{i\cdot}\gamma+b)).$$

This problem can be regarded as a *linear* SVM with transformed feature vectors $V_{i\cdot}^T\in\mathbb{R}^d$, $i=1,2,\ldots,m$.

2.3 Approximating the Kernel

We discuss two techniques to obtain V satisfying (5).

2.3.1 Kernel Matrix Approximation

For some integer d and s such that $0<d\le s<m$, we choose s elements at random from the index set $\{1,2,\ldots,m\}$ to form a subset $\mathcal{S}$. We then find the best rank-d approximation $W_{\mathcal{S},d}$ to $(\Psi^\circ)_{\mathcal{S}\mathcal{S}}$, and its pseudo-inverse $W_{\mathcal{S},d}^+$. We choose V so that

$$VV^T = (\Psi^\circ)_{\cdot\mathcal{S}}W_{\mathcal{S},d}^+(\Psi^\circ)_{\cdot\mathcal{S}}^T, \qquad (6)$$

where $(\Psi^\circ)_{\cdot\mathcal{S}}$ denotes the column submatrix of Ψ° corresponding to $\mathcal{S}$. When s is sufficiently large, this approximation approaches the *best* rank-d approximation in expectation (Drineas and Mahoney, 2005).

To obtain $W_{\mathcal{S},d}$, we form the eigen-decomposition $(\Psi^\circ)_{\mathcal{S}\mathcal{S}} = QDQ^T$ ($Q\in\mathbb{R}^{s\times s}$ orthogonal, D diagonal). Taking $\bar{d}\le d$ to be the number of positive elements in D, we have $W_{\mathcal{S},d} = Q_{\cdot,1..\bar{d}}D_{1..\bar{d},1..\bar{d}}Q^T_{\cdot,1..\bar{d}}$ ($Q_{\cdot,1..\bar{d}}$ denotes the first $\bar{d}$ columns of Q, and so on). The pseudo-inverse is thus $W_{\mathcal{S},d}^+ = Q_{\cdot,1..\bar{d}}D^{-1}_{1..\bar{d},1..\bar{d}}Q^T_{\cdot,1..\bar{d}}$, and V satisfying (6) is therefore

$$V = (\Psi^\circ)_{\cdot\mathcal{S}}Q_{\cdot,1..\bar{d}}D^{-1/2}_{1..\bar{d},1..\bar{d}}. \qquad (7)$$

In practice, rather than defining d a priori, we can choose a threshold $0<\varepsilon_d\ll 1$, then choose the largest integer $\bar{d}\le s$ such that $D_{\bar{d}\bar{d}}\ge\varepsilon_d$. (Therefore $\bar{d}=d$.)

For each sample set $\mathcal{S}$, this approach requires $O(ns^2+s^3)$ operations for the creation and factorization of $(\Psi^\circ)_{\mathcal{S}\mathcal{S}}$, assuming that the evaluation of each kernel entry takes $O(n)$ time. Since our algorithm only requires a single row of V in each iteration, the computation cost of (7) can be amortized over iterations: the cost is $O(sd)$ per iteration if the corresponding row of Ψ° is available; $O(ns+sd)$ otherwise.

2.3.2 Feature Mapping Approximation

The second approach finds a mapping $\phi:\mathbb{R}^n\to\mathbb{R}^d$ that satisfies $\langle\phi^\circ(\mathbf{s}),\phi^\circ(\mathbf{t})\rangle = \mathbb{E}\left[\langle\phi(\mathbf{s}),\phi(\mathbf{t})\rangle\right]$, where the expectation is over the random variables that determine ϕ. Such mapping can be constructed explicitly by random projections (Rahimi and Recht, 2008),

$$\phi(\mathbf{t}) = \sqrt{\frac{2}{d}}\left[\cos(\nu_1^T\mathbf{t}+\omega_1),\cdots,\cos(\nu_d^T\mathbf{t}+\omega_d)\right]^T \qquad (8)$$

where $\nu_1,\ldots,\nu_d\in\mathbb{R}^n$ are i.i.d. samples from a distribution with density $p(\nu)$, and $\omega_1,\ldots,\omega_d\in\mathbb{R}$ are from the uniform distribution on $[0,2\pi]$. The density function $p(\nu)$ is determined by the types of kernels. For the Gaussian kernel $k^\circ(\mathbf{s},\mathbf{t})=\exp(-\sigma\|\mathbf{s}-\mathbf{t}\|_2^2)$, we have $p(\nu)=\frac{1}{(4\pi\sigma)^{d/2}}\exp\left(-\frac{\|\nu\|_2^2}{4\sigma}\right)$, from the Fourier transformation of k°.

This approximation method is less expensive than the previous approach, requiring only $O(nd)$ operations for each data point (assuming that sampling of each vector $v_i \in \mathbb{R}^n$ takes $O(n)$ time).

2.4 Efficient Prediction

Given the solution (γ, b) of (PL), the prediction of a new data point $\mathbf{t} \in \mathbb{R}^n$ can be made efficiently without recovering the support vector coefficient α in (P2), with cost as low as $d/$(no. support vectors) of the cost of an exact-kernel approach.

For the feature mapping approximation, we can simply use the decision function $f(\mathbf{t}) = \mathbf{w}^T\phi(\mathbf{t}) + b$. Using the definitions (1), (4), and $\gamma := V^T\alpha$, we obtain

$$f(\mathbf{t}) = \phi(\mathbf{t})^T \sum_{i=1}^{m} \alpha_i \phi(\mathbf{t}_i) + b = \phi(\mathbf{t})^T \gamma + b.$$

The time complexity in this case is $O(nd)$.

For the kernel matrix approximation approach, we do not know $\phi(\mathbf{t})$, but from (3) we have

$$\phi(\mathbf{t})^T \mathbf{w} + b = \sum_{i=1}^{m} \alpha_i \phi(\mathbf{t})^T \phi(\mathbf{t}_i) + b \approx \sum_{i=1}^{m} \alpha_i k^\circ(\mathbf{t}_i, \mathbf{t}) + b.$$

To evaluate this, we set $\alpha_i = 0$ for all components $i \notin \mathcal{S}$ and $s = d = \bar{d}$. Denoting the nonzero subvector of α by $\alpha_{\mathcal{S}}$, we have $V^T\alpha = V_{\mathcal{S}\cdot}^T \alpha_{\mathcal{S}} = \gamma$. So from (7) and $(\Psi^\circ)_{\mathcal{S}\mathcal{S}} = QDQ^T$ we obtain $\gamma = \left[(\Psi^\circ)_{\mathcal{S}\mathcal{S}} Q_{\cdot,1..\bar{d}} D^{-1/2}_{1..\bar{d},1..\bar{d}}\right]^T \alpha_{\mathcal{S}} = D^{1/2}_{1..\bar{d},1..\bar{d}} Q^T_{\cdot,1..\bar{d}} \alpha_{\mathcal{S}}$. That is, $\alpha_{\mathcal{S}} = Q_{\cdot,1..\bar{d}} D^{-1/2}_{1..\bar{d},1..\bar{d}} \gamma$, which can be computed in $O(d^2)$ time. Therefore, prediction of a test point in this approach will take $O(d^2 + nd)$, including kernel evaluation time.

3 THE ASSET ALGORITHM

Consider the general convex optimization problem

$$\min_{x \in X} f(x), \quad D_X := \max_{x \in X} ||x||_2$$

where f is a convex function and $X \subset \mathbb{R}^d$ is a compact convex set with the radius D_X. We assume that at any $x \in X$, we have available $G(x;\xi)$, a stochastic subgradient estimate depending on random variable $\xi \in \Xi \subset \mathbb{R}^p$ that satisfies $\mathbb{E}[G(x;\xi)] \in \partial f(x)$. The norm deviation of the stochastic subgradients is measured by D_G defined as follows:

$$\mathbb{E}[\|G(x;\xi)\|_2^2] \le D_G^2, \quad \forall x \in X, \xi \in \Xi.$$

Iterate Update. We update iterates as follows:

$$x^j = \Pi_X(x^{j-1} - \eta_j G(x^{j-1}; \xi^j)), \quad j = 1, 2, \ldots,$$

Algorithm 1: ASSET Algorithm.

1: Set $(\gamma^0, b^0) = (\mathbf{0}, 0)$, $(\tilde{\gamma}, \tilde{b}) = (\mathbf{0}, 0)$, $\tilde{\eta} = 0$;
2: **for** $j = 1, 2, \ldots, N$ **do**
3: $\quad \eta_j = \frac{D_X}{D_G\sqrt{j}}$.
4: $\quad$ Choose $\xi^j \in \{1, \ldots, m\}$ at random.
5: $\quad V_{\xi^j\cdot} = \begin{cases} V_{\xi^j\cdot} & \text{for } V \text{ as in (7), or} \\ \phi(\mathbf{t}_{\xi^j}) & \text{for } \phi(\cdot) \text{ as in (8)}. \end{cases}$
6: $\quad$ Compute $G\left(\begin{bmatrix}\gamma^{j-1} \\ b^{j-1}\end{bmatrix}; \xi^j\right)$ following Table 1.
7: $\quad \begin{bmatrix}\gamma^j \\ b^j\end{bmatrix} = \Pi_X\left(\begin{bmatrix}\gamma^{j-1} \\ b^{j-1}\end{bmatrix} - \eta_j G\left(\begin{bmatrix}\gamma^{j-1} \\ b^{j-1}\end{bmatrix}; \xi^j\right)\right)$.
8: $\quad$ **if** $j \ge \bar{N}$ **then**
9: $\quad\quad \begin{bmatrix}\tilde{\gamma} \\ \tilde{b}\end{bmatrix} = \frac{\tilde{\eta}}{\tilde{\eta}+\eta_j}\begin{bmatrix}\tilde{\gamma} \\ \tilde{b}\end{bmatrix} + \frac{\eta_j}{\tilde{\eta}+\eta_j}\begin{bmatrix}\gamma^j \\ b^j\end{bmatrix}$. $\tilde{\eta} = \tilde{\eta} + \eta_j$.
10: $\quad$ **end if**
11: **end for**
12: Define $\tilde{\gamma}^{\bar{N},N} := \tilde{\gamma}$ and $\tilde{b}^{\bar{N},N} := \tilde{b}$.

where $\{\xi^j\}_{j\ge1}$ is an i.i.d. random sequence, Π_X is the Euclidean projection onto X, and $\eta_j > 0$ is a step length. For our problem (PL), we have $x^j = (\gamma^j, b^j)$, and ξ^j is selected to be one of the indices $\{1, 2, \ldots, m\}$ with equal probability, and the subgradient estimate is constructed as shown in Table 1.

Feasible Sets. For classification, we set $X = \{[\gamma, b]^T \in \mathbb{R}^{d+1} : ||\gamma||_2 \le 1/\sqrt{\lambda},\ |b| \le B\}$ for sufficiently large B. (The bound on $\|\gamma\|$ is from Shalev-Shwartz et al. (2011, Theorem 1).) For regression with ε-insensitive loss where $0 \le \varepsilon < \|\mathbf{y}\|_\infty$ and $\mathbf{y} := (y_1, y_2, \ldots, y_m)^T$, we have a similar bound: $\|\gamma\|_2 \le \sqrt{\frac{2(\|\mathbf{y}\|_\infty - \varepsilon)}{\lambda}}$ (Lee and Wright, 2011, Theorem 1).

Estimation of D_G. Using M samples indexed by $\xi^{(l)}$, $l = 1, 2, \ldots, M$, at the first iterate (γ^0, b^0), we estimate D_G^2 as $\frac{1}{M}\sum_{l=1}^{M} d_l^2(||V_{\xi^{(l)}\cdot}||_2^2 + 1)$.

Our algorithm ASSET is summarized in Algorithm 1. Convergence requires the later iterates to be averaged; this begins at iterate $\bar{N} > 0$.

3.1 Convergence

The analysis of robust stochastic approximation (Nemirovski et al., 2009) provides theoretical support.

Theorem 1. *Given the output $\tilde{x}^{\bar{N},N} = (\tilde{\gamma}^{\bar{N},N}, \tilde{b}^{\bar{N},N})^T$ of Algorithm 1 and the optimal objective $f(x^*)$, we have*

$$E[f(\tilde{x}^{\bar{N},N}) - f(x^*)] \le C(\rho)\frac{D_X D_G}{\sqrt{N}}$$

where $C(\rho)$ depends on $\rho \in (0,1)$ where $\bar{N} = \lceil \rho N \rceil$.

Table 1: Loss functions and their corresponding subgradients for classification and regression tasks.

Task	Loss Function, ℓ	Subgradient Estimate, $G\left(\begin{bmatrix}\gamma^{j-1}\\ b^{j-1}\end{bmatrix};\xi^j\right)$
Classification	$\max\{1-y(\mathbf{w}^T\phi(\mathbf{t})+b),0\}$	$\begin{bmatrix}\lambda\gamma^{j-1}+d_jV_{\xi j.}^T\\ d_j\end{bmatrix}$, $d_j=\begin{cases}-y_{\xi j} & \text{if } y_{\xi j}(V_{\xi j.}\gamma^{j-1}+b^{j-1})<1\\ 0 & \text{otherwise}\end{cases}$
Regression	$\max\{\lvert y-(\mathbf{w}^T\phi(\mathbf{t})+b)\rvert-\varepsilon,0\}$	$\begin{bmatrix}\lambda\gamma^{j-1}+d_jV_{\xi j.}^T\\ d_j\end{bmatrix}$, $d_j=\begin{cases}-1 & \text{if } y_{\xi j}>V_{\xi j.}\gamma^{j-1}+b^{j-1}+\varepsilon,\\ 1 & \text{if } y_{\xi j}<V_{\xi j.}\gamma^{j-1}+b^{j-1}-\varepsilon,\\ 0 & \text{otherwise.}\end{cases}$

When we omit the intercept b in (PL), $f(x)$ becomes strongly convex. We can then use steplength $\eta_j=1/(\lambda j)$, and omit the averaging, to achieve faster convergence in theory. We refer to the resulting algorithm after modification as ASSET*. When $\lambda\approx 0$, convergence of ASSET* can be quite slow unless we have $D_G\approx 0$ as well.

Theorem 2. *Given the output x^N and $f(x^*)$, ASSET* with $\eta_j=1/(\lambda j)$ satisfies*

$$E[f(x^N)-f(x^*)]\le \max\left\{\left(\frac{D_G}{\lambda}\right)^2, D_X^2\right\}/N.$$

4 COMPUTATIONAL RESULTS

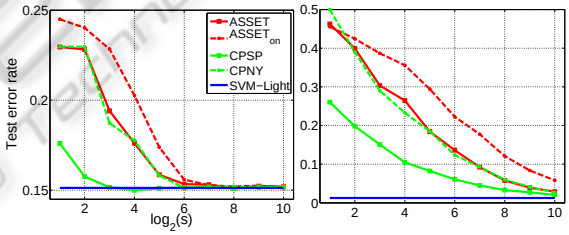
We implemented Algorithm 1 based on the open-source Pegasos code (ours is available at `http://pages.cs.wisc.edu/~sklee/asset/`.) The versions of our algorithms that use kernel matrix approximation are referred to as ASSET$_M$ and ASSET$^*_M$, while those with feature mapping approximation are called ASSET$_F$ and ASSET$^*_F$. For direct comparisons with other codes, we do not include intercept terms, since some of the other codes do not allow such terms to be used without penalization. All experiments with randomness are repeated 50 times.

Table 2 summarizes the six binary classification tasks we use, indicating the values of parameters λ and σ selected using SVM-Light to maximize the classification accuracy on each validation set. (For `MNIST-E`, we use the same parameters as in `MNIST`.) For the first five moderate-size tasks, we compare all of our algorithms against four publicly available codes: two cutting-plane methods CPNY and CPSP, and the other two are SVM-Light and LASVM. The original SVM-Perf (Joachims, 2006) and OCAS (Franc and Sonnenburg, 2008) are not included because they cannot handle nonlinear kernels. For `MNIST-E`, we compare our algorithms using feature mapping approximation to LASVM.

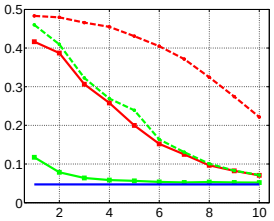
For our algorithms, the averaging parameter is set to $\bar{N}=m-100$ for all cases (averaging is performed for the final 100 iterates). The test error values are computed using the efficient schemes of Section 2.4.

4.1 Effect of Approximation Dimension

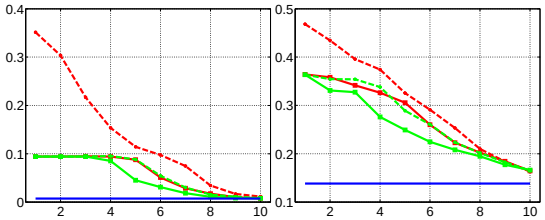
To investigate the effect of kernel approximation dimension on prediction accuracy, we vary the dimension parameter s in Section 2.3 in the range $[2,1024]$, with the eigenvalue threshold $\varepsilon_d=10^{-16}$. Note that s is an upper bound on the actual approximation dimension d for ASSET$_M$, but is equal to d for ASSET$_F$. The codes CPSP and CPNY have a parameter similar to s (as an upper bound of d). For purposes of comparison, we set that parameter to s. For the first five



(a) ADULT. (b) MNIST.



(c) CCAT.



(d) IJCNN. (e) COVTYPE.

Figure 1: The effect of the approximation dimension to the test error. The x-axis shows the values of s in log scale.

Table 2: Data sets and training parameters. [a]http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/, [b]http://leon.bottou.org/papers/loosli-canu-bottou-2006/.

Name	m (train)	valid/test	n	(density)	λ	σ	Note
ADULT	32561	8140/8141	123	(11.2%)	3.07e-08	0.001	UCI Repository.
MNIST	58100	5950/5950	784	(19.1%)	1.72e-07	0.01	Digits 0-4 vs. 5-9.
CCAT	78127	11575/11574	47237	(1.6%)	1.28e-06	1.0	RCV1-v2 collection.
IJCNN	113352	14170/14169	22	(56.5%)	8.82e-08	1.0	IJCNN 2001 Challenge[a].
COVTYPE	464809	58102/58101	54	(21.7%)	7.17e-07	1.0	Forest cover type 1 vs. rest.
MNIST-E	1000000	20000/20000	784	(25.6%)	1.00e-08	0.01	An extended MNIST set[b].

Table 3: Training CPU time (in seconds, h:hours) and test error rate (%) in parentheses. Kernel approximation dimension is varied by setting $s = 512$ and $s = 1024$ for ASSET$_M$, ASSET$^*_M$, CPSP and CPNY. Decomposition methods do not depend on s, so their results are the same in both tables.

	Subgradient Methods		Cutting-plane		Decomposition	
$s = 512$	ASSET$_M$	ASSET$^*_M$	CPSP	CPNY	LASVM	SVM-Light
ADULT	23(15.1±0.06)	24(15.1±0.06)	3020(15.2)	8.2h(15.1)	1011(18.0)	857(15.1)
MNIST	97 (4.0±0.05)	101 (4.0±0.04)	550 (2.7)	348 (4.1)	588 (1.4)	1323 (1.2)
CCAT	95 (8.2±0.08)	99 (8.3±0.06)	800 (5.2)	62 (8.3)	2616 (4.7)	3423 (4.7)
IJCNN	87 (1.1±0.02)	89 (1.1±0.02)	727 (0.8)	320 (1.1)	288 (0.8)	1331 (0.7)
COVTYPE	697(18.2±0.06)	586(18.2±0.07)	1.8h(17.7)	1842(18.2)	38.3h(13.5)	52.7h(13.8)
$s = 1024$	ASSET$_M$	ASSET$^*_M$	CPSP	CPNY	LASVM	SVM-Light
ADULT	78(15.1±0.05)	83(15.1±0.04)	3399(15.2)	7.5h(15.2)	1011(18.0)	857(15.1)
MNIST	275 (2.7±0.03)	275 (2.7±0.02)	1273 (2.0)	515 (2.7)	588 (1.4)	1323 (1.2)
CCAT	265 (7.1±0.05)	278 (7.1±0.04)	2950 (5.2)	123 (7.2)	2616 (4.7)	3423 (4.7)
IJCNN	307 (0.8±0.02)	297 (0.8±0.01)	1649 (0.8)	598 (0.8)	288 (0.8)	1331 (0.7)
COVTYPE	2259(16.5±0.04)	2064(16.5±0.06)	4.1h(16.6)	3598(16.5)	38.3h(13.5)	52.7h(13.8)

moderate-size tasks, we ran our algorithms for 1000 epochs (1000m iterations). The baseline performance values were obtained by SVM-Light.

Figure 1 shows the results. Since ASSET$_M$ and ASSET$^*_M$ yield very similar results in all experiments, we do not plot ASSET$^*_M$. (For the same reason we show ASSET$_F$ but not ASSET$^*_F$.) For small σ values, as in Figure 1(a), all codes achieve good classification performance with small dimension. In other data sets, the chosen values of σ are larger and the intrinsic rank of the kernel matrix is higher, so performance continues to improve as s increases.

CPSP generally requires lower dimension than the others to achieve the same prediction performance. CPSP spends extra time to construct optimal basis functions, whereas the other methods depend on random sampling. However, all approximate-kernel methods including CPSP suffer considerably from the restriction in dimension for COVTYPE.

4.2 Speed Comparison

Here we ran all algorithms other than ours with their default stopping criteria. For ASSET$_M$ and ASSET$^*_M$, we checked the classification error on the test sets ten times per epoch, terminating when the error matched the performance of CPNY. (Since this code uses a similar Nyström approximation of kernel, it is the one most directly comparable with ours in terms of classification accuracy.) The test error was measured using the iterate averaged over the 100 iterations immediately preceding each checkpoint.

Results are shown in Table 3 for $s = 512$ and $s = 1024$. (LASVM and SVM-Light do not depend on s and so their results are the same in both tables.) Our methods are the fastest in most cases. Although the best classification errors among the approximate codes are obtained by CPSP, the runtimes of CPSP are considerably longer than for our methods. In fact, if we compare the performance of ASSET$_M$ with $s = 1024$ and CPSP with $s = 512$, ASSET$_M$ achieves similar test accuracy to CPSP (except for CCAT) but is faster by a factor between two and forty.

It is noteworthy that ASSET$_M$ shows similar performance to ASSET$^*_M$ despite the less impressive theoretical convergence rate of the former. This is because the values of optimal parameter λ were near zero, and thus the objective function lost the strong convexity condition required for ASSET$^*_M$ to work. We observed similar slowdown of Pegasos and SGD

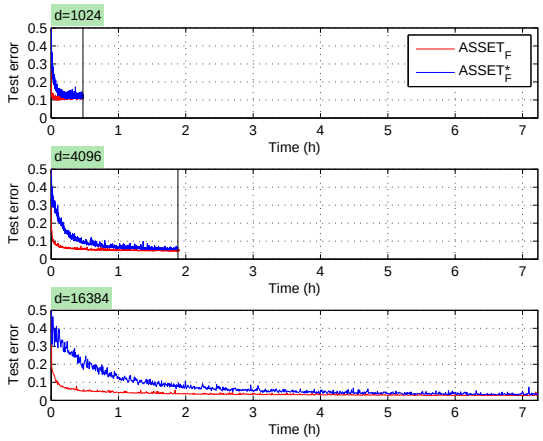


Figure 2: Progress of ASSET_F and ASSET_F^* to their completion (`MNIST-E`), in terms of test error rate.

when λ approaches zero for linear SVMs.

4.3 Large-scale Performance

We take the final data set `MNIST-E` and compare the performance of ASSET_F and ASSET_F^* to the online SVM code LASVM. For a fair comparison, we fed the training samples to the algorithms in the same order.

Figure 2 shows the progress on a single run of our algorithms, with various approximation dimensions d (which is equal to s in this case) in the range $[1024, 16384]$. Vertical bars in the graphs indicate the completion of training. ASSET_F tends to converge faster and shows smaller test error values than ASSET_F^*, despite the theoretical slower convergence rate of the former. With $d = 16384$, ASSET_F and ASSET_F^* required 7.2 hours to finish with a solution of 2.7% and 3.5% test error rate, respectively. LASVM produced a better solution with only 0.2% test error rate, but it required 4.3 days of computation to complete a single pass through the same data.

5 CONCLUSIONS

We have proposed a stochastic gradient framework for training large-scale and online SVMs using efficient approximations to nonlinear kernels, which can be extended easily to other kernel-based learning problems.

ACKNOWLEDGEMENTS

The authors acknowledge the support of NSF Grants DMS-0914524 and DMS-0906818. Part of this work has been supported by the German Research Foundation (DFS) grant for the Collaborative Research Center SFB 876: “Providing Information by Resource-Constrained Data Analysis”.

REFERENCES

Bordes, A., Ertekin, S., Weston, J., and Bottou, L. (2005). Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619.

Bottou, L. (2005). SGD: Stochastic gradient descent. http://leon.bottou.org/projects/sgd.

Chapelle, O. (2007). Training a support vector machine in the primal. *Neural Computation*, 19:1155–1178.

Drineas, P. and Mahoney, M. W. (2005). On the nystrom method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175.

Franc, V. and Sonnenburg, S. (2008). Optimized cutting plane algorithm for support vector machines. In *Proceedings of the 25th International Conference on Machine Learning*, pages 320–327.

Joachims, T. (1999). Making large-scale support vector machine learning practical. In *Advances in Kernel Methods - Support Vector Learning*, pages 169–184. MIT Press.

Joachims, T. (2006). Training linear SVMs in linear time. In *International Conference On Knowledge Discovery and Data Mining*, pages 217–226.

Joachims, T., Finley, T., and Yu, C.-N. (2009). Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59.

Joachims, T. and Yu, C.-N. J. (2009). Sparse kernel svms via cutting-plane training. *Machine Learning*, 76(2-3):179–193.

Lee, S. and Wright, S. J. (2011). Approximate stochastic subgradient estimation training for support vector machines. http://arxiv.org/abs/1111.0432.

Nemirovski, A., Juditsky, A., Lan, G., and Shapiro, A. (2009). Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609.

Nemirovski, A. and Yudin, D. B. (1983). *Problem complexity and method efficiency in optimization*. John Wiley.

Rahimi, A. and Recht, B. (2008). Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*, pages 1177–1184. MIT Press.

Shalev-Shwartz, S., Singer, Y., and Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814.

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

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, pages 928–936.