A Stochastic Conjugate Subgradient Algorithm for Kernelized Support Vector Machines: The Evidence

Di Zhang DZHANG22@USC.EDU
Suvrajeet Sen SUVRAJES@USC.EDU
University of Southern California

Abstract

Kernel Support Vector Machines (Kernel SVM) provide a powerful class of tools for classifying data whose classes are best identified via a nonlinear function. While a Kernel SVM is usually treated as a Quadratic Program (QP), its solution is usually obtained using stochastic gradient descent (SGD). In this paper we treat the Kernel SVM as a Stochastic Quadratic Linear Programming (SQLP) problem which motivates a decomposition-based algorithm that separates parameter choice from error estimation, with the latter being separable by data points. In order to take advantage of the quadratic structure due to the kernel matrix we introduce a conjugate subgradient approach. While convergence of the new method can be shown, the focus of this brief paper is on computational evidence which illustrates that our method maintains the scalability of SGD, while improving the accuracy of classification/optimization.

1. Introduction

In this paper, we focus on Kernel SVM by treating it as a two-stage SQLP represented as follows:

$$\min_{\alpha} \; f(\alpha) = \frac{1}{2}\langle \alpha, K\alpha \rangle + \frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - w_i \langle \alpha, K_i \rangle\}, \tag{1}$$

where K_i is the i-th row of the kernel matrix, and m represents the number of data points.

It is well-known that one can solve (1) by adopting specialized first-order methods such as Pegasos [7] where the subgradient calculations can be carried out very rapidly and in parallel (for a given α). However, for instances in which the condition number of the kernel matrix is large, first-order methods are provably inaccurate [5].

We propose a new algorithm that will accommodate both the curvature of quadratic functions, as well as the decomposition of subgradients by data points, as is common in stochastic programming. This combination includes the computational power of non-smooth conjugate gradient methods [8], sequential sampling and decomposition to provide both computational reliability as well as a well-defined convergence rate. While these theoretical results are provided in a companion paper (available from the authors), this paper only provides the computational evidence.

The rest of the paper is organized as follows: Stochastic Conjugate Subgradient(SCS) Algorithm appears in §2, the computational comparison with a first order method (Pegasos) appears in §3 and our conclusions are provided in §4.

Algorithm 1 Stochastic Conjugate Subgradient (SCS) Algorithm

$\varepsilon > 0, \tau > 0, \delta_0, \eta_1 > 1, \eta_2 > 0, \gamma > 1$ and $k \leftarrow 0$.
Randomly generate s_0 samples from the data set and build an initial RBF kernel Q^0 based on the given samples.
Set a feasible solution $\hat{\alpha}_0 \in \mathbb{R}^{s_0}$ and an initial direction $d_0 \in \mathbb{R}^{s_0}$
$f_0(\alpha) = \frac{1}{2}\langle \alpha, Q^0 \alpha \rangle + \frac{1}{s_0}\sum_{i=1}^{s_0} \max\{0, 1 - w_i \langle \alpha, Q_i^0 \rangle\}$.
while $||d_k|| > \varepsilon$ **do**
    $k = k + 1$
    Obtain $g_k \in \partial f_{k-1}(\hat{\alpha}_k)$, $G_k = \{g_k, -d_{k-1}\}$ and calculate $d_k = -Nr(G_k)$.
    Apply Algorithm 2 to find step size t_k.
    Set $\alpha_k = \hat{\alpha}_k + t_k d_k$.
    Next randomly generate a set of new samples s_k of cardinality $|s_k|$.
    Define $S_k \triangleq S_{k-1} \cup s_k$.
    Let $\beta_k = (\alpha_k, \overrightarrow{0}) \in \mathbb{R}^{|S_k|}$, $\hat{\beta}_{k-1} = (\hat{\alpha}_{k-1}, \overrightarrow{0}) \in \mathbb{R}^{|S_k|}$; build Q^k using S_k.
    Construct $f_k(\alpha) = \frac{1}{2}\alpha^T Q^k \alpha + \frac{1}{S_k}\sum_{i=1}^{S_k} \max\{0, 1 - w_i \langle \alpha, Q_i^k \rangle\}$
    Randomly generate a set of new samples T_k of cardinality $|S_k|$ independent of S_{k-1}.
    Construct $\hat{f}_k(\alpha) = \frac{1}{2}\alpha^\top \hat{Q}^k \alpha + \frac{1}{S_k}\sum_{i=1}^{S_k} \max\{0, 1 - \hat{w}_i \langle \alpha, \hat{Q}_i^k \rangle\}$ based on T_k.
    if $f_k(\beta^k) - f_k(\hat{\beta}^{k-1}) \leq \eta_1(\hat{f}_k(\alpha^k) - \hat{f}_k(\hat{\alpha}^{k-1}))$ *and* $||d_k|| > \eta_2 \delta_k$ **then**
        $\hat{\alpha}^k \leftarrow \beta^k, \quad \delta_k \leftarrow \min\{\gamma \delta_{k-1}, \frac{\varepsilon}{\xi}\}$
    else
        $\hat{\alpha}^k \leftarrow \hat{\beta}^{k-1}, \quad \delta_k \leftarrow \frac{\delta_{k-1}}{\gamma}$
    end
end

2. Stochastic Conjugate Subgradient Algorithm

The main ingredients of the algorithm 1 include three important components:

- Sequential function approximation. In many cases we have a fixed number of data points. However, our focus is on situations where the data can be queried from an oracle sequentially. As a result, we will not fix the value of m in (1). Instead, we use an online version of sample average approximation (OSAA) [2, 3] to approximate function f using f_k, where

$$f_k(\alpha) = \frac{1}{2}\alpha^T Q^k \alpha + \frac{1}{S_k}\sum_{i=1}^{S_k} \max\{0, 1 - w_i \langle \alpha, Q_i^k \rangle\}, \tag{2}$$

  Using Hoeffding's Inequality, [4], we can use (2) to approximate (1) to an arbitrarily high level of accuracy as the number of samples increases.

- Direction finding. This idea is inspired by Wolfe's non-smooth conjugate subgradient method which uses the smallest norm of the convex combination of the previous search direction (d_{k-1}) and the current subgradient (g_k).

- Choice of step size. This is achieved by combining concepts of both trust region and line search methods [6, 8].

Most of the logic of Algorithm 2 is motivated by Wolfe's Line Search Algorithm [8]. The output of the step size will satisfy two metrics: (i) Identify a set L which includes points that improve the objective function value, (ii) Identify set R for which the directional derivative estimate is improved. The algorithm seeks points which belong to $L \cap R$.

Algorithm 2 Line Search Algorithm

Set $m_2 < m_1 < 0.5$ and $b = \frac{\delta_k}{n}$.
Let $g(t) \in \partial \hat{f}_{k-1}(\hat{\alpha}_k + t \cdot d_k)$ and define the intervals L and R.

$$L = \{t > 0 \mid \hat{f}_{k-1}(\hat{\alpha}_k + t \cdot d_k) - \hat{f}_{k-1}(\hat{\alpha}_k) \leq -m_1 ||d_k||^2 t\},$$
$$R = \{t > 0 \mid \langle g(t), d_k \rangle \geq -m_2 ||d_k||^2\}.$$

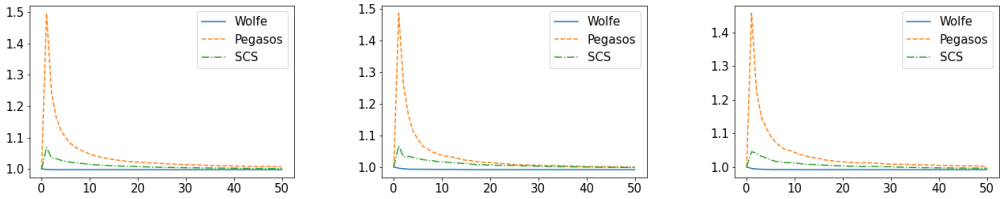
Choose $t||d_k|| \in [b, \delta_k]$.
if $t \in L \backslash R$ **then**
    $t = 2t$ until $t \in R$ or $t||d_k|| > \delta_k$. Set $I = [t/2, t]$.
    if $t||d_k|| > \delta_k$ **then**
        Return $t/2$.
    else
        if $t \in L \cap R$ **then**
            Return t
        else
            while $t \notin L \cap R$ **do**
                Set t be the middle point of I
                if $t \in R \backslash L$ **then**
                    Replace I by its left half
                else
                    Replace I by its right half
                end
            end
        end
    end
end
if $t \in R \backslash L$ **then**
    $t = t/2$ until $t \in L$ or $t||d_k|| < b$. Set I be the interval of $[t, 2t]$.
    if $t \in L \cap R$ **then**
        Return t
    else
        if $t||d_k|| < b$ **then**
            Return $t = 0$
        else
            Repeat line (7)-(9) while $t \notin L \cap R$.
        end
    end
end
Return t

3. Computational Results

Our computational experiments are based on data sets available at the UCI Machine Learning Repository [1]. Our study considers three different methods: Kernelized Pegasos algorithm[1], SCS algorithm and Wolfe's algorithm [8]: Wolfe uses objective function (1) while Pegasos and SCS use (2). However, in order to compare the progress of the objective function values for different algorithms, we will track the values of (1) for all three algorithms. The values of $\{f(\hat{\alpha}_k)\}_{k=1}^{k=50}$ and

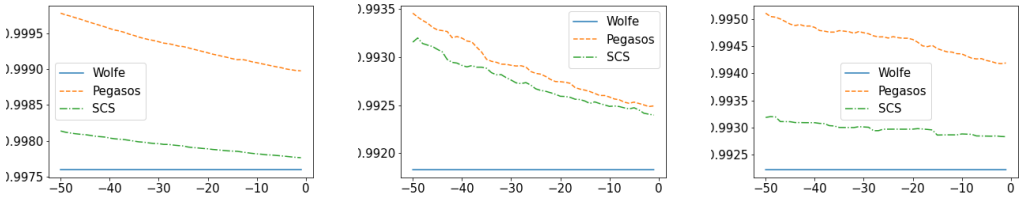
1. The computations below are based on our implementation of the Pegasos Algorithm.

$\{f(\hat{\alpha}_k)\}_{k=-50}^{k=-1}$ are shown in Figures 1 and 2 (In the interest of brevity, we only illustrate the results for small data sets).



Breast Cancer Heart Failure Wine Quality

Figure 1: $\{f(\hat{\alpha}_k)\}_{k=1}^{k=50}$ for different combinations (data,algorithm).



Breast Cancer Heart Failure Wine Quality

Figure 2: $\{f(\hat{\alpha}_k)\}_{k=-50}^{k=-1}$ for different combinations (data,algorithm).

Remark: (a) For objective function value, SCS is lower than the Pegasos algorithm. (b) SCS also converges faster than the Pegasos Algorithm. (c) The beauty of the Pegasos algorithm is that it provides a decision rule without providing a solution to (1). This decision rule provides a classifier without any reference to optimality. On the other hand, the SCS algorithm is truly an optimization algorithm with a stopping rule in which the steps (and stopping) are guided by d_k ($||d_k|| < \varepsilon$).

The performance of all three algorithms for different data sets are given in Table 1. Since Wolfe's algorithm requires the entire data set, the instances with very large data sets are beyond its reach, and are reported by N/A in Table 1. The accuracy and time reported in the Table 1 are based on the average of 20 independent runs for each pair (data, algorithm) using different random seeds.

4. Conclusion

Our approach leads to a class of online algorithms that go beyond first-order approximations. It includes several features of Wolfe's method (e.g., stability and good convergence rate) as well as the online aspects of Pegasos which promotes good computational times (see figure(1)). In contrast to the Pegasos algorithm, we find that the optimization performance of SCS algorithm is more reliable and provides consistently lower objective values (see Figures 1 and 2). It appears that such reliability may be difficult to achieve using first-order methods without additional efforts in fine-tuning.

Table 1: Classification performance for different data sets

		heart attack	breast cancer	wine quality	avila Bible	magic telescope	room occupancy	Skin-Nonskin
	samples	273	500	680	2000	5000	7500	200000
Pegasos	accuracy	0.833	0.95	0.855	0.714	0.735	0.974	0.93
	time(s)	0.750	1.147	3.933	5.602	25.992	56.549	176.454
SCS	accuracy	0.833	0.97	0.86	0.718	0.740	0.976	0.97
	time(s)	6.121	4.921	12.497	38.266	33.256	39.887	19.684
Wolfe	accuracy	0.833	0.97	0.87	0.726	N/A	N/A	N/A
	time(s)	1.481	2.289	4.195	32.978	N/A	N/A	N/A

Acknowledgment. This paper is supported, in part, by a grant AFOSR FA9550-20-1-0006.

References

[1] Andrew Frank. Uci machine learning repository. *http://archive. ics. uci. edu/ml*, 2010.

[2] Julia L Higle and Suvrajeet Sen. Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Mathematics of operations research*, 16(3):650–669, 1991.

[3] Julia L Higle and Suvrajeet Sen. Finite master programs in regularized stochastic decomposition. *Mathematical Programming*, 67(1):143–168, 1994.

[4] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. In *The collected works of Wassily Hoeffding*, pages 409–426. Springer, 1994.

[5] Yinyu Luenberger, David G. Ye. *Linear and Nonlinear Programming*, volume 2. Springer, 1984.

[6] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer Science & Business Media, 2006.

[7] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[8] Philip Wolfe. A method of conjugate subgradients for minimizing nondifferentiable functions. In *Nondifferentiable optimization*, pages 145–173. Springer, 1975.