Pegasos: Primal Estimated sub-GrAdient Solver for SVM

Abstract

$\tilde{8}\tilde{0}\tilde{0},\tilde{0}\tilde{0}\tilde{0}(\lambda\epsilon))$

We describe and analyze a simple and effective iterative algorithm for solving the optimization problem cast by Support Vector Machines (SVM). Our method alternates between stochastic gradient descent steps and projection steps. We prove that the number of iterations required to obtain a solution of accuracy ϵ is $\tilde{O}(1/\epsilon)$. In contrast, previous analyses of stochastic gradient descent methods require $\Omega(1/\epsilon^2)$ iterations. As in previous devised SVM solvers, the number of iterations also scales linearly with $1/\lambda$, where λ is the regularization parameter of SVM. For a linear kernel, the total run-time of our method is $\tilde{O}(d/(\lambda\epsilon))$, where d is a bound on the number of non-zero features in each example. Since the run-time does *not* depend directly on the size of the training set, the resulting algorithm is especially suited for learning from large datasets. Our approach can seamlessly be adapted to employ non-linear kernels while working solely on the primal objective function. We demonstrate the efficiency and applicability of our approach by conducting experiments on large text classification problems, comparing our solver to existing state-of-the art SVM solvers. For example, it takes less than 5 seconds for our solver to converge when solving a text classification problem from Reuters Corpus Volume 1 (RCV 1) with $800,000$ training examples.

1. Introduction

Support Vector Machines (SVMs) are effective and popular classification learning tool. The task of learning a support vector machine is cast as a constrained quadratic programming. However, in its native form, it is in fact an unconstrained empirical loss minimization with a penalty term for the norm of the classifier that is being learned. Formally, given a training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{+1, -1\}$, we would like to find the minimizer of the problem

$$\min_{\mathbf{w}} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{m}\sum_{(\mathbf{x},y)\in S} \ell(\mathbf{w};(\mathbf{x},y)) \tag{1}$$

where

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

We denote the objective function of Eq. (1) by $f(\mathbf{w})$. An optimization method finds an ϵ-

accurate solution $\hat{\mathbf{w}}$ if $f(\hat{\mathbf{w}}) \leq min_{\mathbf{w}} f(\mathbf{w}) + \epsilon$. The original SVM problem also includes a bias term, b. We omit the bias throughout the first sections and defer the description of an extension which employs a bias term to Sec. 4.

We describe and analyze in this paper a simple iterative algorithm, called Pegasos, for solving Eq. (1). The algorithm performs T iterations and also requires an additional parameter k, whose role is explained in the sequel. Pegasos alternates between stochastic subgradient descent steps and projection steps. The parameter k determines the number of examples from S the algorithm uses on each iteration for estimating the subgradient. When $k = m$, Pegasos reduces to a variant of the subgradient projection method. We show that in this case the number of iterations that is required in order to achieve an ϵ-accurate solution is $\tilde{O}(1/(\lambda\epsilon))$. At the other extreme, when $k = 1$, we recover a variant of the stochastic (sub) gradient method. In the stochastic case, we analyze the probability of obtaining a good approximate solution. Specifically, we show that with probability of at least $1 - \delta$ our algorithm finds an ϵ-accurate solution using only $\tilde{O}(1/(\delta\lambda\epsilon))$ iterations, while each iteration involves a single inner product between $\mathbf{w}$ and $\mathbf{x}$. This rate of convergence does *not* depend on the size of the training set and thus our algorithm is especially suited for large datasets.

2. The Pegasos Algorithm

In this section we describe the Pegasos algorithm for solving the optimization problem given in Eq. (1). The algorithm receives as input two parameters: T - the number of iterations to perform; k - the number of examples to use for calculating sub-gradients. Initially, we set $\mathbf{w}_1$ to any vector whose norm is at most $1/\sqrt{\lambda}$. On iteration t of the algorithm, we first choose a set $A_t \subseteq S$ of size k. Then, we replace the objective in Eq. (1) with an approximate objective function,

$$f(\mathbf{w}; A_t) = \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{k} \sum_{(\mathbf{x},y)\in A_t} \ell(\mathbf{w}; (\mathbf{w}, y)).$$

Note that we overloaded our original definition of f as the original objective can be denoted either as $f(\mathbf{w})$ or as $f(\mathbf{w}; S)$. We interchangeably use both notations depending on the context. Next, we set the learning rate $\eta_t = 1/(\lambda t)$ and define A_t^+ to be the set of examples for which $\mathbf{w}$ suffers a non-zero loss. We now perform a two-step update as follows. We scale $\mathbf{w}_t$ by $(1 - \eta_t\lambda)$ and for all examples $(\mathbf{x}, y) \in A_t^+$ we add to $\mathbf{w}$ the vector

$\frac{y\eta_t}{k}\mathbf{x}$. We denote the resulting vector by $\mathbf{w}_{t+\frac{1}{2}}$. This step can be also written as $\mathbf{w}_{t+\frac{1}{2}} = \mathbf{w}_t - \eta_t \nabla_t$, where

$$\nabla_t = \lambda \mathbf{w}_t - \frac{1}{|A_t|} \sum_{(\mathbf{x},y)\in A_t^+} y\mathbf{x} \tag{3}$$

The definition of the hinge-loss implies that ∇_t is a sub-gradient of $f(\mathbf{w}; A_t)$ at $\mathbf{w}_t$. Last, we set $\mathbf{w}_{t+1}$ to be the projection of $\mathbf{w}_{t+\frac{1}{2}}$ onto the set

$$B = \{\mathbf{w} : ||\mathbf{w}|| \leq 1/\sqrt{\lambda}\} \tag{4}$$

That is, $\mathbf{w}_{t+1}$ is obtained by scaling $\mathbf{w}_{t+\frac{1}{2}}$ by $min\{1, 1/(\sqrt{\lambda}||\mathbf{w}_{t+\frac{1}{2}}||)\}$. As we show in our analysis below, the optimal solution of SVM is in the set B. Informally speaking, we can always project back onto the set B as we only get closer to the optimum. The output of Pegasos is the last vector $\mathbf{w}_{T+1}$.

Note that if we choose $A_t = S$ on each round t then we obtain the sub-gradient projection method. On the other extreme, if we choose A_t to contain a single randomly selected example, then we recover a variant of the stochastic gradient method. In general, we allow A_t to be a set of k examples sampled i.i.d. from S.

We conclude this section with a short discussion of implementation details when the instances are sparse, namely, when each instance has very few non-zero elements. In this case, we can represent $\mathbf{w}$ as a triplet $(\mathbf{v}, a, \nu)$ where $\mathbf{v}$ is a dense vector and a, ν are scalars. The vector $\mathbf{w}$ is defined through the triplet as follows: $\mathbf{w} = a\mathbf{v}$ and ν stores the squared norm of $\mathbf{w}$, $\nu = ||\mathbf{w}||^2$. Using this representation, it is easily verified that the total number of operations required for performing one iteration of Pegasos with $k = 1$ is $O(d)$, where d is the number of non-zero elements in $\mathbf{x}$.