Stochastic Subgradient Methods

Lingjie Weng **Yutian Chen**
Bren School of Information and Computer Science
University of California, Irvine
{wengl, yutianc}@ics.uci.edu

Abstract

Stochastic subgradient methods play an important role in machine learning. We introduced the concepts of subgradient methods and stochastic subgradient methods in this project, discussed their convergence conditions as well as the strong and weak points against their competitors. We demonstrated the application of (stochastic) subgradient methods to machine learning with a running example of training support vector machines (SVM) throughout this report.

1 Introduction

We have explored a variety of optimization methods in class (e.g., generalized descent methods, Newton's method, interior-point methods), all of which apply to differentiable functions. However, many functions that arise in practice maybe non-differentiable at certain places. A common example is the absolute value function $f(x) = |x|$, or its multivariate extension, the ℓ_1-norm $f(x) = \|x\|_1 = \sum_{i=1}^{n} |x_i|$. Therefore, we aim to discuss a generalization of the gradient for non-differentiable convex functions.

In a lot of applications, the exact subgradient could be difficult to calculate because of errors in measurements, uncertainty in the data or because the computation is intractable. However, it is usually possible to get a noisy estimate to the subgradient. In this case we can use the noisy estimate as the true value in the subgradient method, which is called the *stochastic subgradient method*. We'll show in this report that many convergence conditions still apply to the stochastic version, and the expectation of the solution could achieve the same order of convergence rate as the exact subgradient method with often much fewer computations. The example of training SVMs as a stochastic subgradient method illustrates its advantage over deterministic counterparts. It also shows the close connection between the statistic optimization and machine learning theories. This part of the report is mostly based on the lecture notes of Stephen Boyd [1] and a tutorial on stochastic optimization in ICML 2010 [8].

2 Subgradient Methods

2.1 Definition

A vector $g \in \mathbb{R}^n$ is a subgradient of a convex function at x if

$$f(y) \geq f(x) + g^T(y - x) \quad \forall y \tag{1}$$

If f is convex and differentiable, then its gradient at x is a subgradient. But a subgradient can exist even when f is not differentiable at x, as illustrated in figure 1. The same example shows that there can be more than one subgradient of a function f at a point x. The set of all subgradients at x is called the subdifferential, and is denoted by $\partial f(x)$.

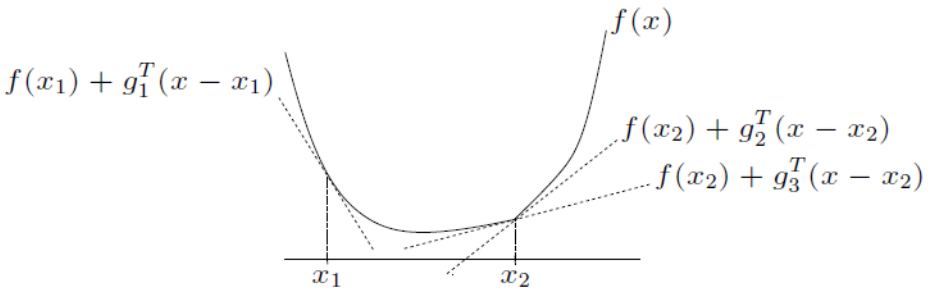


Figure 1: At x_1, the convex function f is differentiable, and g_1(which is the derivative of f at x_1) is the unique subgradient at x_1. At the point x_2, f is not differentiable. At this point, f has many subgradients, g_2 and g_3, are shown. Taken from [1]

2.2 Convergence for subgradient methods

Subgradient methods are iterative methods for solving convex minimization problems. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function with domain $\mathbb{R}^n$. A classical subgradient method iterates:

$$x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)} \tag{2}$$

where $x^{(k)}$ denotes the kth iterate, $g^{(k)}$ denotes any subgradient of f at $x^{(k)}$, and $\alpha_k > 0$ is the kth step size. Thus, at each iteration of the subgradient method,we take a step in the direction of a negative subgradient. Recall that a subgradient of f at x is any vector g that satisfies the inequality $f(y) \geq f(x) + g^T(y - x)$for all y. It may happen that $-g^{(k)}$ is not a descent direction for f at $x^{(k)}$. We therefore maintain a list f_{best} that keeps track of the lowest objective function value found so far, i.e.,the one with smallest function value.At each step, we set

$$f_{\text{best}}^{(k)} = \min\{f_{\text{best}}^{(k-1)}, f(x^{(k)})\} \tag{3}$$

and set $i_{\text{best}}^{(k)} = k$ if $f(x^{(k)}) = f_{\text{best}}^{(k)}$, i.e., if $x^{(k)}$ is the best point found so far(In a descent method, there is no need to do this, since the current point is always the best one so far.) Then we have the best objective value found in k iterations

$$f_{\text{best}}^{(k)} = \min\{f(x^{(1)}), ..., f(x^{(k)})\} \tag{4}$$

The subgradient methods are guaranteed to converge to within some range of the optimal value under certain assumption. We assume

(1) there is a minimizer of f, say x^*.

(2) the norm of the subgradients is bounded, i.e., there is a G such that $\|f^{(k)}\|_2 \leqslant G$ for all k. This will be the case if f satisfies the Lipschitz condition

$$|f(u) - f(v)| \leqslant G\|u - v\|_2, \tag{5}$$

for all u, v, because then $\|g\|_2 \leqslant G$ for any $g \in \partial f(x)$,and any x.

(3) a number R is known that satisfies $R \geqslant \|x^{(1)} - x^*\|_2$.

Under those three assumptions, the subgradient algorithm is guaranteed to converge to within some range of the optimal value, i.e., we have

$$\lim_{k\to\infty} f_{\text{best}}^{(k)} - f^* < \epsilon, \tag{6}$$

where f^* denotes the optimal value of the function f.The number ϵ is a function of the step size. So for different step size rules, the convergence results are different. The followings are five basic step size rules.

(a) Constant step size, $\alpha_k = \alpha$.

(b) Constant step length, $\alpha_k = \gamma / \|g^{(k)}\|_2$, which gives $\|x^{(k+1)} - x^{(k)}\|_2 = \gamma$.

(c) Square summable but not summable step size, i.e. any step sizes satisfying

$$\alpha_k \geq 0, \qquad \sum_{k=1}^{\infty} \alpha_k^2 < \infty, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty \tag{7}$$

(d) Nonsummable diminishing, i.e. any step sizes satisfying

$$\alpha_k \geq 0, \qquad \lim_{k \to \infty} \alpha_k = 0, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty \tag{8}$$

(e) Nonsummable diminishing step lengths, i.e.

$$\alpha_k = \gamma_k / \|g^{(k)}\|_2 \tag{9}$$

where$\gamma_k \geq 0, \qquad \lim_{k \to \infty} \gamma_k = 0, \qquad \sum_{k=1}^{\infty} \gamma_k = \infty.$

For all five rules, the step-sizes are determined "off-line", before the method is iterated, the step-sizes do not depend on preceding iterations. This "off-line" property of subgradient methods differs from the "on-line" step-size rules used for descent methods for differentiable functions: many methods for minimizing differentiable functions satisfy Wolfe's sufficient conditions for convergence, where step-sizes typically depend on the current point and the current search direction.

Termination of the subgradient search procedure occurs when $\frac{R^2 + G^2 \sum_{k=1}^{\infty} \alpha_i^2}{2 \sum_{k=1}^{\infty} \alpha_i} \leqslant \epsilon$ (which is an upper bound of $f_{\text{best}}^{(k)} - f^*$) is really, really slow. In fact, up to the present, the subgradient method has suffered from lack of definite termination criterion. So the subgradient methods are usually used for problems in which very high accuracy is not required.

2.3 Subgradient methods for unconstrained problem

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. We seek to solve the unconstrained problem

$$p^* = \min_x f(x) \Longleftrightarrow x^* \in \arg\min_{x \in \mathbb{R}^n} f(x) \tag{10}$$

For differentiable f the optimality condition reads

$$f(x^*) = \min_{x \in \mathbb{R}^n} f(x) \Longleftrightarrow 0 = \nabla f(x^*) \tag{11}$$

For non-differentiable f the optimality condition generalizes to

$$f(x^*) = \min_{x \in \mathbb{R}^n} f(x) \Longleftrightarrow 0 \in \partial f(x^*) \tag{12}$$

which follows from the definition of the subgradient $f(x) \geqslant f(x^*) + 0^T(x - x^*), x \in \mathbb{R}^n$. The pseudocode of the subgradient method for unconstrained convex function f is shown in Algorithm 1.

Algorithm 1 Subgradient method for minimizing unconstrained $f(x)$

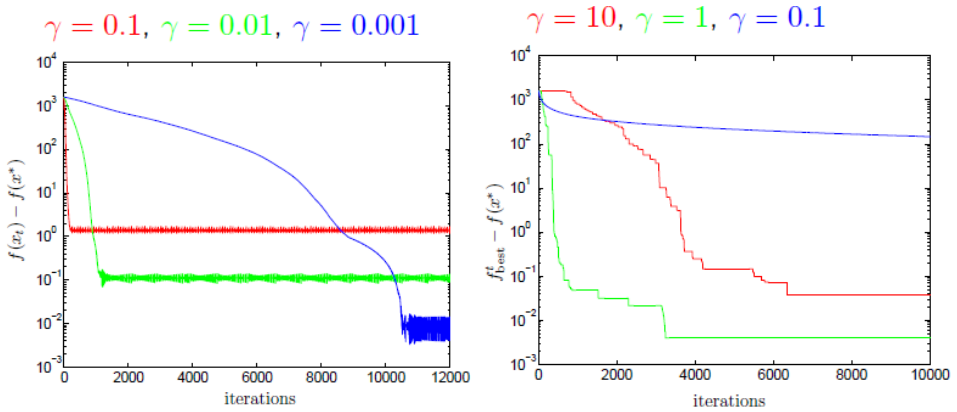
Initialize x_0, $k = 0$, compute a subgradient $g^k \in \partial f(x^k)$
repeat
  $x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}$
until Termination condition is satisfied
$x^{(k)}$: kth iterate, $\alpha_k > 0$: kth step size, $g^{(k)}$: any subgradient of f at $x^{(k)}$.

Let's take the L_2-norm SVM learning problem as an example. Consider the primal form of SVM:

$$\begin{aligned} \min \quad & \|\mathbf{w}\|^2 + \frac{C}{m} \sum_{i=1}^{m} \xi_i \\ s.t. \quad & (\langle \mathbf{w}, \mathbf{x}_i \rangle y_i) \geqslant 1 - \xi_i, i = 1, ..., m \\ & \xi_i \geqslant 0, i = 1, ..., m \end{aligned} \tag{13}$$

Table 1: Subgradients of function $F(\mathbf{w}, \boldsymbol{\xi})$

i-th function	$\partial_{\boldsymbol{w}} f_i(\boldsymbol{w}, b)$	$\partial_b f_i(\boldsymbol{w}, b)$
$I_+ = \{i \mid y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) > 1\}$	$\mathbf{0}$	0
$I_0 = \{i \mid y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) = 1\}$	$\mathbf{Co}\{\mathbf{0}, -y_i \boldsymbol{x}_i\}$	$\mathbf{Co}\{0, -y_i\}$
$I_- = \{i \mid y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) < 1\}$	$-y_i \boldsymbol{x}_i$	$-y_i$



(a) (b)

Figure 2: Constrained SVM: the value of $f_{\text{best}}^{(t)} - f^*$ versus iteration number t. Taken from [10]

Note that $\xi_i \geqslant \max\{0, 1 - (\langle \mathbf{w}, \mathbf{x}_i \rangle y_i)\}$ and the equality is attained in the optimum, i.e., we can optimize as

$$s.t. \quad \xi_i = \max\{0, 1 - (\langle \mathbf{w}, \mathbf{x}_i \rangle y_i)\} = \ell(\langle \mathbf{w}, \mathbf{x}_i \rangle y_i) \tag{14}$$

(which is usually called the hinge loss). We can therefore reformulate the problem as an unconstrained convex problem

$$\min \|\mathbf{w}\|^2 + \frac{C}{m} \sum_{i=1}^{m} \ell(\langle \mathbf{w}, \mathbf{x}_i \rangle y_i) \tag{15}$$

The first term is differentiable, the second term is a point-wise maximum, thus this formulation fit the subgradient framework. The corresponding subgradients are shown in table 2 where b is the coefficient for the constant term of x.

For this L_2-norm SVM learning problem, we considered the constant step length rule $\alpha_t = \frac{\gamma}{\|g^t\|_2}$ and the diminishing step size rule $\alpha_t = \frac{\gamma}{t}$. The function converges to a suboptimal solution $\lim_{t \to \infty} f_{\text{best}}^{(t)} - f^* \leqslant \frac{\gamma G}{2}$ for the first step size rule, while it converges to optimal solution $\lim_{t \to \infty} f_{\text{best}}^{(t)} - f^* = 0$ for the second step size rule. Figure 3(a) and 3(b) shows convergence of $f_{\text{best}}^{(t)} - f^*$ for different γ. The figures reveal a trade-off: larger γ gives faster convergence, but larger final suboptimality. Non-monotonicity and the slow convergence of $f(x^{(t)})$ are also clearly seen from these plots.

2.4 Subgradient methods for constrained problem

One extension of the subgradient method is the projected subgradient method, which solves the constrained optimization problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in C \end{array} \tag{16}$$

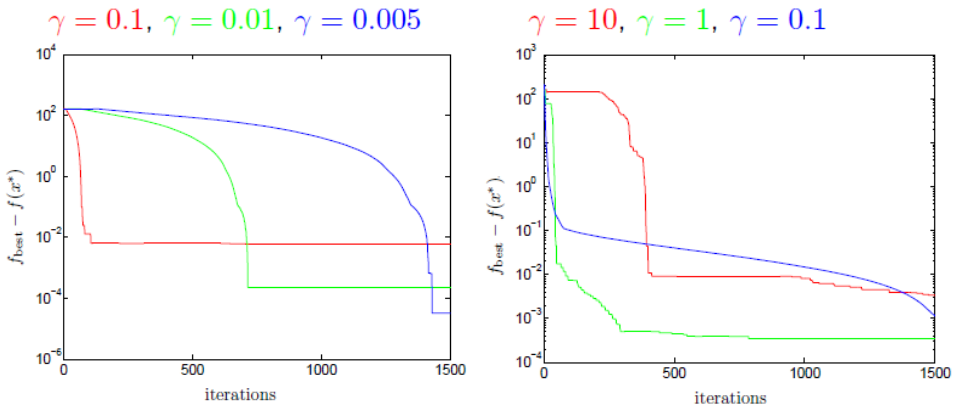


Figure 3: Constrained SVM: the value of $f_{\text{best}}^{(t)} - f^*$ versus iteration number t. Taken from [10]

where C is a convex set, the projected subgradient method uses the iteration

$$x^{(k+1)} = P\left(x^{(k)} - \alpha_k g^{(k)}\right) \tag{17}$$

where P is (Euclidean) projection on C. The pseudocode of the projected subgradient method for constrained convex function f is shown in Algorithm 2.

Algorithm 2 Subgradient method for minimizing constrained $f(x)$

Initialize x_0, $k = 0$, compute a subgradient $g^k \in \partial f(x^k)$
repeat
  $\tilde{x}^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}$
  $x^{(k+1)} = P(\tilde{x}^{(k+1)})$
until Termination condition is satisfied
$x^{(k)}$: kth iterate, $\alpha_k > 0$: kth step size, $g^{(k)}$: any subgradient of f at $x^{(k)}$.

For the same SVM learning example

$$\min \|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m} \ell(\langle \mathbf{w}, \mathbf{x}_i\rangle y_i) \tag{18}$$

An alternative formulation of the same problem is

$$\begin{aligned} \min \quad & \frac{1}{m}\sum_{i=1}^{m} \ell(\langle \mathbf{w}, \mathbf{x}_i\rangle y_i) \\ s.t. \quad & \|\mathbf{w}\| \leqslant \mathbf{B} \end{aligned} \tag{19}$$

note that the regularization constant $C > 0$ is replaced here by a less abstract constant $\mathbf{B} > 0$. To implement subgradient method, we need only a projection operator

$$P(\mathbf{w}) = \{\mathbf{w} \to \frac{\mathbf{w}}{\|\mathbf{w}\|}\mathbf{B}\} \tag{20}$$

so after update $\mathbf{w} := \mathbf{w} - \alpha_k \mathbf{w}'$, we project $\mathbf{w} := \frac{\mathbf{w}}{\|\mathbf{w}\|}\mathbf{B}$.

For the reformulated L_2-norm SVM learning problem, we also plotted the convergence results with constant step length rule and diminishing step size. Figure 4(a) and 4(b) show convergence of $f_{\text{best}}^{(t)} - f^*$ for different γ. Comparing these two plots with figure 3(a) and 3(b) respectively, we found this approach converges much faster.

The subgradient method can also be extended to solve the general inequality constrained problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leqslant 0, \quad i = 1, \ldots, m \end{array} \tag{21}$$

where f_i are convex. The algorithm takes the same form:

$$x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)} \tag{22}$$

where $\alpha_k > 0$ is a step size, and $g^{(k)}$ is a subgradient of the objective or one of the constraint functions at$x$. More specifically, we take

$$g^{(k)} = \begin{cases} \partial f_0(x) & \text{if } f_i(x) \leq 0 \; \forall i = 1 \ldots m \\ \partial f_j(x) & \text{for some } j \text{ such that } f_j(x) > 0 \end{cases} \tag{23}$$

where ∂f denotes the subdifferential of f. In other words: If the current point is feasible, the algorithm uses an objective subgradient, as if the problem is unconstrained; if the current point is infeasible, the algorithm chooses a subgradient of any violated constraint.

In this generalized version of the subgradient algorithm, the iterates can be (and often are) infeasible. In contrast, the iterates of the projected subgradient method (and the basic subgradient algorithm) are always feasible. So when the current point is infeasible, we can use an over-projection step length, as we would when solving convex inequalities. If we know (or estimate) f^*, we can use the Polyak's step length when the current point is feasible. Thus our step lengths are chose as

$$\alpha_k = \begin{cases} (f_0(x^{(k)}) - f^*)/\|g^{(k)}\|_2^2, \; x^{(k)} & \text{feasible} \\ (f_i(x^{(k)}) + \epsilon)/\|g^{(k)}\|_2^2, \; x^{(k)} & \text{infeasible} \end{cases} \tag{24}$$

where ϵ is a small positive margin, and i is the index of the most violated inequality in the case where $x^{(k)}$ is infeasible.

3 Stochastic Subgradient Methods

When the subgradient of the objective function is difficult to compute exactly due to various reasons such as errors in measurements or intractability in the computation, we can use a noisy estimate of the subgradient for optimization. With a proper choice of the step size, we can guarantee the convergence with probability 1 or even stronger conclusions. Stochastic methods also apply when the objective function itself is difficult to compute exactly. In the following subsections, we'll introduce the stochastic subgradient descent (SD) method, its convergence condition and stochastic programming.

3.1 Stochastic Subgradient Methods

For a convex function $f : \mathbf{R}^n \to \mathbf{R}$, $\tilde{g}$ is called the *noisy (unbiased) subgradient* of f at $x \in \mathbf{dom} f$ if $g = \mathbf{E}\tilde{g} \in \partial f(x)$, i.e.,

$$f(z) \geq f(x) + (\mathbf{E}\tilde{g})^T (z - x), \quad \forall z \in \mathbf{dom} f \tag{25}$$

If x is also a random variable, then we say that $\tilde{g}$ is a noisy subgradient of f at x (which is random) if $\mathbf{E}(\tilde{g}|x) \in \partial f(x)$. We can consider the noisy subgradient as the sum of g and a random variable with zero mean. The noise can come from error in computing the true subgradient, error in Monte Carlo evaluation of a function defined as an expected value, or measurement error.

The algorithm of stochastic subgradient method is almost the same as the regular subgradient method except for using a noisy estimate instead of the true subgradient and a more limited set of step size rules. The pseudocode of the stochastic subgradient method for minimization a convex function f without constraints is shown in Algorithm 3.

As the regular subgradient method, $f(x^{(k)})$ can increase during the algorithm, so we keep track of the best point found so far, and the associated function value

$$f_{best}^{(k)} = \min\{f(x^{(1)}), \ldots, f(x^{(k)})\} \tag{26}$$

Algorithm 3 Stochastic subgradient method for minimizing $f(x)$

Initialize x_0, $k = 0$
repeat
$x^{(k+1)} = x^{(k)} - \alpha_k \tilde{g}^{(k)}$
until Termination condition is satisfied
$x^{(k)}$: kth iterate, $\alpha_k > 0$: kth step size, $\tilde{g}^{(k)}$: noisy subgradient of f at $x^{(k)}$.

We'll show a very basic convergence result for the stochastic subgradient method. Assume there is an x^* that minimizes f, and a G for which $\mathbf{E}\|\tilde{g}(k)\|_2^2 \leq G^2$ for all k. We also assume that B satisfies $\mathbf{E}\|x^{(1)} - x^*\|_2^2 \leq B^2$. It can be proved that

$$\mathbf{E}f_{best}^{(k)} - f^* \leq \frac{B^2 + G^2 \sum_{i=1}^k \alpha_i^2}{2\sum_{i=1}^k \alpha_i} \tag{27}$$

we can get the convergence in expectation

$$\lim_{k\to\infty} \mathbf{E}f_{best}^{(k)} \to f^* \tag{28}$$

for various step size rules such as square-summable but not summable sequence (e.g. $\alpha_k = 1/k$), and not summable diminishing sequence (e.g. $\alpha_k = 1/\sqrt{k}$). Using Markov's inequality, we obtain the convergence in probability, that is, for any $\epsilon > 0$,

$$\lim_{k\to\infty} \mathbf{Prob}(f_{best}^{(k)} \geq f^* + \epsilon) = 0 \tag{29}$$

More sophisticated methods can be used to show almost sure convergence.

3.2 Applying SD to SVM

Let's still take the SVM learning problem as an example. Consider the norm constraint formulation of SVM:

$$\min_{\|\mathbf{w}\|_2 \leq B} F(\mathbf{w}) = \min_{\|\mathbf{w}\|_2 \leq B} \frac{1}{m} \sum_{i=1}^m \ell(\langle \mathbf{w}, \mathbf{x}_i \rangle y_i) \tag{30}$$

The objective function $F(\mathbf{w})$ is the average hinge loss over the training set. The regular gradient descent algorithm, referred to as *batch subgradient method*, computes the subgradient as the average of the subgradients of the loss function for every data point, i.e.,

$$g^{(k)} = \frac{1}{m} \sum_{i=1}^m \partial_{\mathbf{w}} \ell(\langle \mathbf{w}_k, \mathbf{x}_i \rangle y_i) \tag{31}$$

and then project the new parameter vector back to the ball of radius B at each step as $\mathbf{w}^{(k+1)} \leftarrow B\frac{\mathbf{w}^{(k+1)}}{\|\mathbf{w}^{(k+1)}\|}$ if the new vector outside the ball.

In contrast, we consider the stochastic version, Algorithm 3, with the extra projection step above. At each iteration, the stochastic subgradient is computed by randomly drawing a sample from m data points, $x_{i(k)}$, and computing the noisy subgradient as

$$\tilde{g}^{(k)} = \partial_{\mathbf{w}} \ell(\langle \mathbf{w}_k, \mathbf{x}_{i(k)} \rangle y_{i(k)}) \tag{32}$$

It's easy to see that $\mathbf{E}\tilde{g}^{(k)} = g^{(k)}$. In this example, we choose the step size rule as $\alpha_k = \frac{B/G}{\sqrt{k}}$ and the final solution is computed as $\bar{\mathbf{w}} = \frac{1}{k}\sum_{i=1}^k \mathbf{w}_k$.

It can be proved that for the batch subgradient descent with such a choice the step size, we can achieve the sub-optimality:

$$F(\mathbf{w}^{(k)}) - F(\mathbf{w}^*) \leq O(\frac{GB}{\sqrt{k}}) \tag{33}$$

where $\mathbf{w}^*$ is the optimal solution. This is the best possible convergence rate using only $F(\mathbf{w})$ and $\partial F(\mathbf{w})$. For SD we can get the same guarantee in the expectation of the objective function:

$$\mathbf{E}[F(\bar{\mathbf{w}}^{(k)}] - F(\mathbf{w}^*) \leq O(\frac{GB}{\sqrt{k}}) \tag{34}$$

Taking into consideration that the batch method takes m times as many operations as the stochastic method with the same guarantee on errors, we would come to an intuitive conclusion that if only taking simple gradient steps, it's better to be stochastic. Formally speaking, to guarantee an error bound of ϵ in $F(\mathbf{w})$, the runtime of SD is $O(\frac{G^2B^2}{\epsilon^2}d)$ while the runtime of batch subgradient method is $O(\frac{G^2B^2}{\epsilon^2}md)$, where d is the dimensionality of the feature vector. The stochastic method achieves the optimal runtime if only using gradients and only assuming Lipschitz condition. The difference in speed is significant for high dimensional problems.

We should notice that the error bound of the stochastic method is defined in expectation. The convergence rate as a function of update steps for individual realizations of the method is usually slower than the batch method due to the noise. In fact, the slow convergence acts as a smoothing factor that reduces the variance in the stochastic subgradients. We could consider a mixed method that draws a subset of data points from $\{x_i\}$ and computes the average as the stochastic subgradient. This method, usually called *mini-batch method*, provides a tradeoff between the computational burden at each step and the error variance (relevant to convergence rate). Also, the runtime of SD is optimal only in the first order methods. Second order methods such as Newton's method can use much fewer steps than SD for the same error bound although it may take much more time in each step.

3.3 Stochastic Programming

We have discussed in the previous subsections the situation when the subgradient is difficult or expensive to compute and we take a noisy estimate to optimize the objective function. Now we'll talk about the case when the objective function itself is difficult to compute. A *stochastic programming* problem has the form

$$\begin{array}{ll} \text{minimize} & \mathbf{E}f_0(x,\omega) \\ \text{subject to} & \mathbf{E}f_i(x,\omega) \leq 0, \quad i = 1,\ldots,m \end{array} \tag{35}$$

where $x \in \mathbf{R}^n$ is the optimization variable, and ω is a random variable. If $f_i(x,\omega)$ is convex in x for each ω, the problem is a convex stochastic programming problem. In this case the objective and constraint functions are convex.

Stochastic programming can be used to model a variety of robust design or decision problems with uncertain data. Although the basic form above involves only expectation or average values, some tricks can be used to capture other measures of the probability distributions of $f_i(x,\omega)$. We can replace an objective or constraint term $\mathbf{E}f(x,\omega)$ with $\mathbf{E}\Phi(f(x,\omega))$, where Φ is a convex increasing function. For example, with $\Phi(u) = \max(u,0)$, we can form a constraint of the form

$$\mathbf{E}f_i(x,\omega)_+ \leq \epsilon \tag{36}$$

where ϵ is a positive parameter, and $(.)_+$ denotes positive part. Here $Ef_i(x,\omega)_+$ has a simple interpretation as the expected violation of the ith constraint. It's also possible to combine the constraints, using a single constraint of the form

$$\mathbf{E}\max(f_1(x,\omega)_+,\ldots,f_m(x,\omega_+) \leq \epsilon \tag{37}$$

The lefthand side here can be interpreted as the expected worst violation (over all constraints).

Problem (35) is associated with another optimization problem that replaces the random variable by its expected value, which is sometimes called the certainty equivalent of the original stochastic programming,

$$\begin{array}{ll} \text{minimize} & f_0(x,\mathbf{E}\omega) \\ \text{subject to} & f_i(x,\mathbf{E}\omega) \leq 0, \quad i = 1,\ldots,m \end{array} \tag{38}$$

By Jensen's inequality, we get

$$\mathbf{E}f_i(x,\omega) \geq f_i(x,\mathbf{E}\omega) \tag{39}$$

It tells us that the constraint set for the certainty equivalent problem is larger than the original stochastic problem (35), and its objective is smaller. It follows that the optimal value of the certainty equivalent problem gives a lower bound on the optimal value of the stochastic problem (35). (It can be a poor bound, of course.)

The objective function $f(x) = \int F(x,\omega)p(\omega)d\omega$ is usually not easy to compute exactly especially for a function with a high dimensional variable. However, in many applications, we can approximately compute f using Monte Carlo methods. Assume we generate M independent identically distributed (*iid*) samples $\omega_1, \ldots, \omega_M$. We can compute an estimate of f as

$$\hat{f}(x) = \frac{1}{M}\sum_{i=1}^{M} F(x,\omega_i) \tag{40}$$

Further more, if we calculate the subgradient of $F(x,\omega)$ for each value of ω, $\partial_x F(x,\omega_i)$, it can be proved that their average, $\tilde{g}(x) = \frac{1}{M}\sum_{i=1}^{M}\partial_x F(x,\omega_i)$, gives an unbiased noisy estimate of the subgradient of $f(x)$, i.e.,

$$g(x) \triangleq \mathbf{E}\tilde{g}(x) = \mathbf{E}\partial F(x,\omega) \in \partial_x \mathbf{E}f(x,\omega) \tag{41}$$

Therefore, we can use the stochastic subgradient descent method described in subsection 3.1 to solve the optimization problem (35). This result is independent of M. We can even take $M = 1$. In this case, $\tilde{g} = \partial F(x,\omega_1)$. In other words, we simply generate one sample ω_1 and use the subgradient of F for that value of ω. In this case $\tilde{g}$ could hardly be called a good approximation of a subgradient of f, but its mean is a subgradient, so it is a valid noisy unbiased subgradient.

On the other hand, we can take M to be large. In this case, $\tilde{g}$ is a random vector with mean g and very small variance, i.e., it is a good estimate of g, a subgradient of f at x.

3.4 Applying Stochastic Programming to SVM

The stochastic programming is closely related to another training setting in machine learning: minimizing generalization error. Since the purpose of machine learning is to learn a model that generalizes to unobserved data, it makes more sense to minimize the generalization error $\mathbf{E}_p[\ell(\mathbf{w},z)]$ rather than the error on the empirical distribution $\frac{1}{M}\sum_{i=1}^{M}\ell(\mathbf{w},z_i)$ (sometimes called empirical risk), where p is the true distribution of data z, and z_i is the ith of the M data items in the training set. Actually, optimization on the empirical risk often leads to over-fitting in the training set.

From this aspect, we would like to formulate the machine learning problem in the generic learning setting [9] (taking SVM training as an example again) as

$$\min_{\|\mathbf{w}\|_2\le B} F(\mathbf{w}) = \min_{\|\mathbf{w}\|_2\le B} \mathbf{E}_z F(\mathbf{w},z) \tag{42}$$

$$= \min_{\|\mathbf{w}\|_2\le B} \mathbf{E}_{(x,y)}\ell(\langle\mathbf{w},\mathbf{x}\rangle y) \tag{43}$$

In general, training a model by minimizing the generalization error (42) is a stochastic programming problem. Since the loss function ℓ of SVM is a convex function, problem (43) is also a convex stochastic programming problem.

We compare two approaches to problem (43). One approach is called sample average approximation (SAA) [4, 7, 5]. When an mth sample, (x_m, y_m), is collected, SAA optimizes the empirical loss

$$\min_{\mathbf{w}} \hat{F}(\mathbf{w}) = \min_{\mathbf{w}} \frac{1}{m}\sum_{i=1}^{m}\ell(\langle\mathbf{w},\mathbf{x}_i\rangle y_i) \tag{44}$$

by any optimization method such as SD, batch subgradient descent or Newton's method. This corresponds to the empirical risk minimization method we mentioned above if we consider the set of m samples as the training set. Analysis on the error bound with SAA is typically based on the uniform concentration theorem.

The other approach is called sample approximation (SA) [6]. SA uses the simple stochastic subgradient descent method to directly minimize $F(\mathbf{w})$. Every time a new sample is collected, a noisy

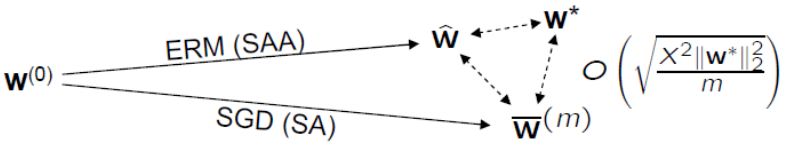


Figure 4: Illustration of the relationships between the initialization ($\mathbf{w}^{(0)}$), outputs of SAA/SA, $\hat{\mathbf{w}}/\bar{\mathbf{w}}^{(m)}$ and the optimal solution $\mathbf{w}^*$. Taken from [8]

.

subgradient is computed as $\tilde{g}^{(k)} = \partial_{\mathbf{w}}\ell(\langle \mathbf{w}^{(k)}, \mathbf{x}_k\rangle y_k)$. Then $\mathbf{w}^{(k)}$ is updated based on that weak estimate to $\partial F(\mathbf{w}^{(k)})$. Since this is a typical stochastic subgradient descent method on a convex optimization problem, conclusions about convergence rate on the general stochastic methods also apply here.

We show the conclusions of SAA and SA below without proofs. Let $\hat{\mathbf{w}}$ be the optimal solution of problem (44). We can find an upper bound on the error $L(\hat{\mathbf{w}}) - L(\mathbf{w}^*) \leq O\sqrt{\frac{G^2B^2}{m}}$ by the uniform concentration theorem. Thus to guarantee $L(\bar{\mathbf{w}}_{SAA}^{(m)}) \leq L(\mathbf{w}^*) + \epsilon$, we need a sample of size $m = \Omega(\frac{G^2B^2}{\epsilon^2})$ which leads to the conclusion that the runtime to guarantee an upper bound on the error is $\Omega(md) = \Omega(\frac{G^2B^2}{\epsilon^2}d)$. Notice that this result applies to any optimization algorithm.

In contrast, for SA after one pass over the m data items, the generalization error with $\bar{\mathbf{w}}_{SA}^{(m)}$ is upper bounded by $L(\bar{\mathbf{w}}_{SA}^{(m)}) \leq L(\bar{\mathbf{w}}^*) + O(\sqrt{\frac{G^2B^2}{m}})$. So to guarantee that $L(\bar{\mathbf{w}}_{SA}^{(m)}) - L(\bar{\mathbf{w}}^*) \leq \epsilon$ the sample size would be $m = O(\frac{G^2B^2}{\epsilon^2})$ and consequently the runtime for an error gap of ϵ is $O(md) = O(\frac{G^2B^2}{\epsilon^2}d)$.

SA guarantees the best-possible generalization error with optimal runtime in theory. The relationship between the outputs of SAA/SA and the optimal solution is shown in figure 4. However, we should notice that SA achieves the optimal generalization error up to a constant factor, and in practice, SA still seems to be better than SA in the generalization error. Nevertheless, stochastic gradient descent is usually considered the fastest algorithm for solving many machine learning problems.

4 Discussion

The subgradient method is very simple and general, it applies directly to nondifferentiable f,any subgradient from $\partial f(x^{(k)})$ is sufficient. It is guaranteed to converge for properly selected step size. However, comparing with ordinary gradient method, the convergence can be much slower, as it usually requires a lot of iterations. There is no monotonic convergence since it is not a descent method, so the line search cannot be used to final optimal α_k. Besides, there is no good stopping criteria.

Subgradient method is part of a large class of methods known as first-order methods, which are based on subgradients or closely related objects. In contrast, second-order methods(like interior-point methods or Newton's method) typically use Hessian information; each iteration is expensive (both in flops and memory) but convergence is obtained very quickly. There are several general approaches that can be used to speed up subgradient methods. Localization methods such as cutting-plane and ellipsoid methods are typically much faster than basic subgradient methods. Another general approach is to base the update on some conic combination of previously evaluated subgradients. For example, in bundle methods, the update direction is found as the least-norm convex combination of some previous subgradients. Further more, the convergence can be improved by keeping memory of past steps,

$$x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)} + \beta_k(x^{(k)} - x^{(k-1)}) \tag{45}$$

where α_k and β_k are positive.We can interpret the second term as a memory term. Such algorithms have state, whereas the basic subgradient methods is stateless. Conjugate gradients methods have a similar form.

The stochastic variant of the subgradient method relaxes the condition for the application subgradient method where exact computation of the subgradient isn't necessary anymore. This usually slows down the convergence as a function of update steps due to the noise in the descent direction, but it could save a lot of time in each step because a noisy estimate can be easy to obtain. Moreover, the same order of convergence rate as the deterministic descent methods either in minimizing the empirical error or the generalization error makes SD a very competitive tool in machine learning.

Stochastic learning is in fact closed related to online learning except for few differences such as the objective function (online learning minimizes empirical regret), and the assumption for observation (adversary vs *iid*) distributions [8]. There are also some other learning algorithms that enjoy similar or even better advantages than SD including stochastic coordinate descent [3] and stochastic second order methods [2].

References

[1] S. Boyd. Stochastic subgradient methods. lecture slides and notes for ee364b. Stanford University, Spring quarter 2007-2008.

[2] R. Byrd, G. Chin, W. Neveitt, and J. Nocedal. On the use of stochastic hessian information in unconstrained optimization. Technical report, Technical Report, Northwestern University, 2010.

[3] C. Hsieh, K. Chang, C. Lin, S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415. ACM, 2008.

[4] A. J. Kleywegt, A. Shapiro, and T. H. de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM J. on Optimization*, 12:479–502, February 2002.

[5] E. Plambeck, B. Fu, S. Robinson, and R. Suri. Sample-path optimization of convex stochastic performance functions. *Mathematical Programming*, 75(2):137–176, 1996.

[6] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[7] R. Y. Rubinstein and A. Shapiro. Optimization of static simulation models by the score function method. *Math. Comput. Simul.*, 32:373–392, October 1990.

[8] N. Srebro and A. Tewari. Stochastic optimization: Icml 2010 tutorial, July 2010.

[9] V. Vapnik. The nature of statistical learning theory. 1995.

[10] F. Vojtech. A short tutorial on subgradient methods.