Stochastic Subgradient Methods

Stephen Boyd and Almir Mutapcic, with additions by John Duchi
Notes for EE364b, Stanford University, Spring 2017–2018

April 12, 2018

1 Noisy unbiased subgradient

Suppose $f : \mathbf{R}^n \to \mathbf{R}$ is a convex function. We say that a random vector $\tilde{g} \in \mathbf{R}^n$ is a *noisy (unbiased) subgradient* of f at $x \in \mathbf{dom}\, f$ if $g = \mathbf{E}\,\tilde{g} \in \partial f(x)$, *i.e.*, we have

$$f(z) \geq f(x) + (\mathbf{E}\,\tilde{g})^T(z - x)$$

for all z. Thus, $\tilde{g}$ is a noisy unbiased subgradient of f at x if it can be written as $\tilde{g} = g + v$, where $g \in \partial f(x)$ and v has zero mean.

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

$$\forall z \qquad f(z) \geq f(x) + \mathbf{E}(\tilde{g}|x)^T(z - x)$$

holds almost surely. We can write this compactly as $\mathbf{E}(\tilde{g}|x) \in \partial f(x)$. ('Almost surely' is to be understood here.)

The noise can represent (presumably small) error in computing a true subgradient, error that arises in Monte Carlo evaluation of a function defined as an expected value, or measurement error.

Some references for stochastic subgradient methods are [Sho98, §2.4], [Pol87, Chap. 5]. Some books on stochastic programming in general are [BL97, Pre95, Mar05].

2 Stochastic subgradient method

The stochastic subgradient method is essentially the subgradient method, but using noisy subgradients and a more limited set of step size rules. In this context, the slow convergence of subgradient methods helps us, since the many steps help 'average out' the statistical errors in the subgradient evaluations.

We'll consider the simplest case, unconstrained minimization of a convex function $f : \mathbf{R}^n \to \mathbf{R}$. The stochastic subgradient method uses the standard update

$$x^{(k+1)} = x^{(k)} - \alpha_k \tilde{g}^{(k)},$$

where $x^{(k)}$ is the kth iterate, $\alpha_k > 0$ is the kth step size, and $\tilde{g}^{(k)}$ is a noisy subgradient of f at $x^{(k)}$,

$$\mathbf{E}(\tilde{g}^{(k)}|x^{(k)}) = g^{(k)} \in \partial f(x^{(k)}).$$

Even more so than with the ordinary subgradient method, we can have $f(x^{(k)})$ increase during the algorithm, so we keep track of the best point found so far, and the associated function value

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

The sequences $x^{(k)}$, $\tilde{g}^{(k)}$, and $f_{\rm best}^{(k)}$ are, of course, stochastic processes.

3 Convergence

We'll prove a very basic convergence result for the stochastic subgradient method, using step sizes that are square-summable but not summable,

$$\alpha_k \geq 0, \qquad \sum_{k=1}^{\infty} \alpha_k^2 = \|\alpha\|_2^2 < \infty, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty.$$

We assume there is an $x^\star$ that minimizes f, and a G for which $\mathbf{E}\,\|g^{(k)}\|_2^2 \leq G^2$ for all k. We also assume that R satisfies $\mathbf{E}\,\|x^{(1)} - x^\star\|_2^2 \leq R^2$.

We will show that

$$\mathbf{E}\, f_{\rm best}^{(k)} \to f^\star$$

as $k \to \infty$, *i.e.*, we have convergence in expectation. We also have convergence in probability: for any $\epsilon > 0$,

$$\lim_{k\to\infty} \mathbf{Prob}(f_{\rm best}^{(k)} \geq f^\star + \epsilon) = 0.$$

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

We have

$$\begin{aligned}
\mathbf{E}\left(\|x^{(k+1)} - x^\star\|_2^2 \;\middle|\; x^{(k)}\right) &= \mathbf{E}\left(\|x^{(k)} - \alpha_k\tilde{g}^{(k)} - x^\star\|_2^2 \;\middle|\; x^{(k)}\right) \\
&= \|x^{(k)} - x^\star\|_2^2 - 2\alpha_k\,\mathbf{E}\left(\tilde{g}^{(k)T}(x^{(k)} - x^\star) \;\middle|\; x^{(k)}\right) + \alpha_k^2\,\mathbf{E}\left(\|\tilde{g}^{(k)}\|_2^2 \;\middle|\; x^{(k)}\right) \\
&= \|x^{(k)} - x^\star\|_2^2 - 2\alpha_k\,\mathbf{E}(\tilde{g}^{(k)}|x^{(k)})^T(x^{(k)} - x^\star) + \alpha_k^2\,\mathbf{E}\left(\|\tilde{g}^{(k)}\|_2^2 \;\middle|\; x^{(k)}\right) \\
&\leq \|x^{(k)} - x^\star\|_2^2 - 2\alpha_k(f(x^{(k)}) - f^\star) + \alpha_k^2\,\mathbf{E}\left(\|\tilde{g}^{(k)}\|_2^2 \;\middle|\; x^{(k)}\right),
\end{aligned}$$

where the inequality holds almost surely, and follows because $\mathbf{E}(\tilde{g}^{(k)}|x^{(k)}) \in \partial f(x^{(k)})$.

Now we take expectation to get

$$\mathbf{E}\,\|x^{(k+1)} - x^\star\|_2^2 \leq \mathbf{E}\,\|x^{(k)} - x^\star\|_2^2 - 2\alpha_k(\mathbf{E}\,f(x^{(k)}) - f^\star) + \alpha_k^2 G^2,$$

using $\mathbf{E}\,\|\tilde{g}^{(k)}\|_2^2 \leq G^2$. Recursively applying this inequality yields

$$\mathbf{E}\,\|x^{(k+1)} - x^\star\|_2^2 \leq \mathbf{E}\,\|x^{(1)} - x^\star\|_2^2 - 2\sum_{i=1}^{k} \alpha_i(\mathbf{E}\,f(x^{(i)}) - f^\star) + G^2\sum_{i=1}^{k} \alpha_i^2.$$

Using $\mathbf{E}\,\|x^{(1)} - x^\star\|_2^2 \leq R^2$, $\mathbf{E}\,\|x^{(k+1)} - x^\star\|_2^2 \geq 0$, and $\sum_{i=1}^k \alpha_i^2 \leq \|\alpha\|_2^2$, we have

$$2\sum_{i=1}^k \alpha_i(\mathbf{E}\,f(x^{(i)}) - f^\star) \leq R^2 + G^2\|\alpha\|_2^2.$$

Therefore we have

$$\min_{i=1,\ldots,k}(\mathbf{E}\,f(x^{(i)}) - f^\star) \leq \frac{R^2 + G^2\|\alpha\|_2^2}{2\sum_{i=1}^k \alpha_i},$$

which shows that $\min_{i=1,\ldots,k} \mathbf{E}\,f(x^{(i)})$ converges to $f^\star$.

Finally, we note that by Jensen's inequality and concavity of the minimum function, we have

$$\mathbf{E}\,f_{\rm best}^{(k)} = \mathbf{E}\min_{i=1,\ldots,k} f(x^{(i)}) \leq \min_{i=1,\ldots,k} \mathbf{E}\,f(x^{(i)}),$$

so $\mathbf{E}\,f_{\rm best}^{(k)}$ also converges to $f^\star$.

To show convergence in probability, we use Markov's inequality to obtain, for $\epsilon > 0$,

$$\mathbf{Prob}(f_{\rm best}^{(k)} - f^\star \geq \epsilon) \leq \frac{\mathbf{E}(f_{\rm best}^{(k)} - f^\star)}{\epsilon}.$$

The righthand side goes to zero as $k \to \infty$, so the lefthand side does as well.

4 Example

We consider the problem of minimizing a piecewise-linear function,

$$\text{minimize} \quad f(x) = \max_{i=1,\ldots,m}(a_i^T x + b_i),$$

with variable $x \in \mathbf{R}^n$. At each step, we evaluate a noisy subgradient of the form $\tilde{g}^{(k)} = g^{(k)} + v^{(k)}$, where $g^{(k)} \in \partial f(x^{(k)})$ and $v^{(k)}$ are independent zero mean random variables.

We illustrate the stochastic subgradient method with the same problem instance used in the notes on subgradient methods, with $n = 20$ variables, $m = 100$ terms, and problem problem data a_i and b_i generated from a unit normal distribution. We start with $x^{(1)} = 0$, and use the square summable but not summable step rule $\alpha_k = 1/k$. To report $f(x^{(k)}) - f^\star$, we compute the optimal value $f^\star$ using linear programming.

The noises $v^{(k)}$ are IID $\mathcal{N}(0, 0.5I)$. Since the norm of the vectors a_i is on the order of 4 or 5 (the RMS value is $\sqrt{20}$), the subgradient noise is around 25% compared to the true subgradient.

Figure 1 shows the convergence of the stochastic subgradient method for two realizations of the noisy subgradient process, together with the noise-free case for comparison. This figure shows that convergence is only a bit slower with subgradient noise.

We carried out 100 realizations, and show the (sample) mean and standard deviation of $f_{\rm best}^{(k)} - f^\star$ for k in multiples of 250, in figure 2. (The error bars show the mean plus and minus one standard deviation.) Figure 3 shows the empirical distribution (over the 100 realizations) of $f_{\rm best}^{(k)} - f^\star$ at iterations $k = 250$, $k = 1000$, and $k = 5000$.

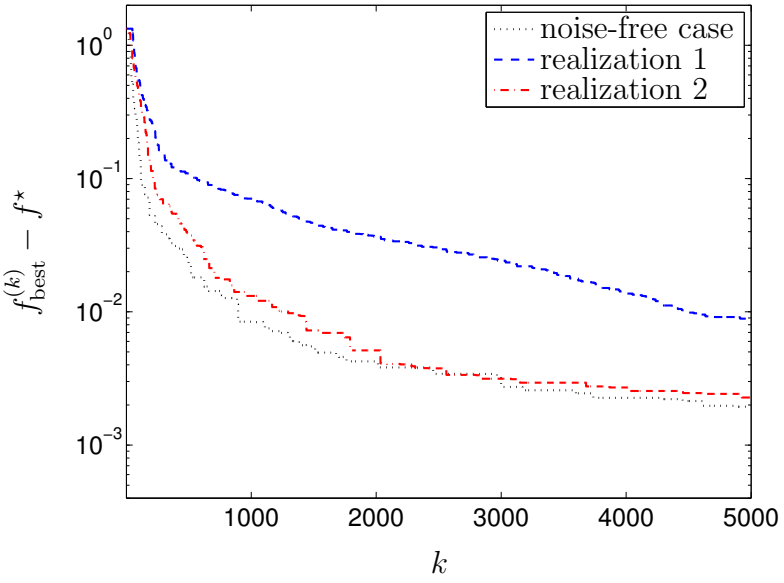

Figure 1: The value of $f_{\rm best}^{(k)} - f^\star$ versus iteration number k, for the subgradient method with step size $\alpha_k = 1/k$. The plot shows a noise-free realization, and two realizations with subgradient noise.

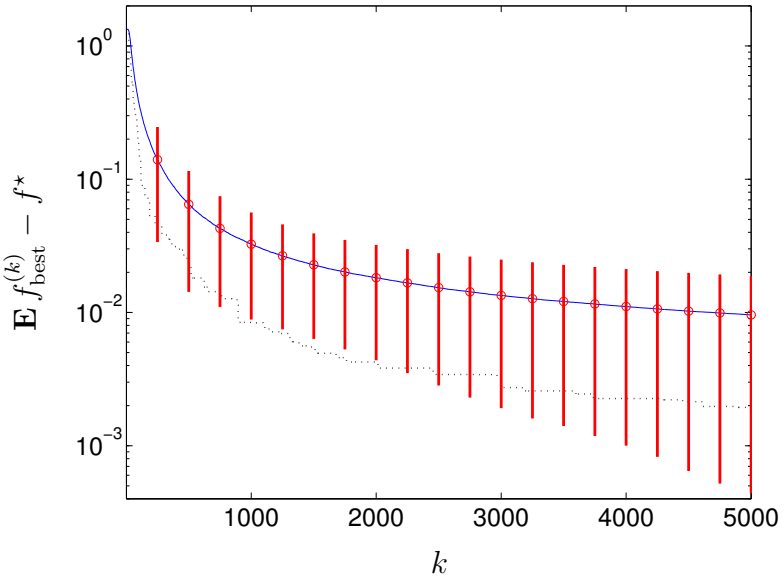

Figure 2: Average and one standard deviation error bars for $f_{\rm best}^{(k)} - f^\star$ versus iteration number k, computed using 100 realizations, every 250 iterations.

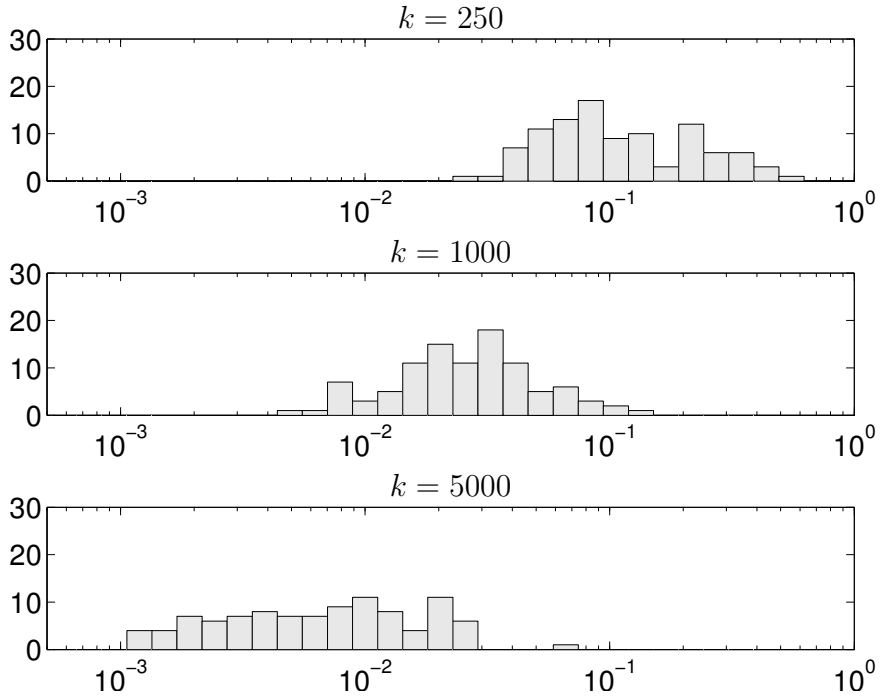

Figure 3: Empirical distributions of $f_{\text{best}}^{(k)} - f^\star$ at $k = 250$, $k = 1000$, and $k = 5000$ iterations, based on 100 realizations.

5 Stochastic programming

A stochastic programming problem has the form

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

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

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

where ϵ is a positive parameter, and $(\cdot)_+$ denotes positive part. Here $\mathbf{E}\, f_i(x,\omega)_+$ 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

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

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

Constraints of the form $\mathbf{Prob}(f_i(x,\omega) \leq 0) \geq \eta$, which require a constraint to hold with a probability (or reliability) exceeding η, are called *chance constraints*. These constraints cannot be directly handled using the simple trick above, but an expected violation constraint can often give a good approximation for a chance constraint. (Some chance constraints can be handled exactly, *e.g.*, when $f(x,\omega) = (a+\omega)^T x - b$, ω is Gaussian, and $\eta \geq 0.5$.)

Recall that Jensen's inequality tells us

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

Now consider the problem

$$\begin{array}{ll} \text{minimize} & f_0(x, \mathbf{E}\, \omega) \\ \text{subject to} & f_i(x, \mathbf{E}\, \omega) \leq 0, \quad i=1,\ldots,m, \end{array} \tag{2}$$

obtained by replacing the random variable in each function by its expected value. This problem is sometimes called the *certainty equivalent* of the original stochastic programming problem (1), even though they are equivalent only in very special cases. By Jensen's inequality, the constraint set for the uncertainty equivalent problem is larger than the original stochastic problem (1), and its objective is smaller. It follows that the optimal value of the uncertainty equivalent problem gives a lower bound on the optimal value of the stochastic problem (1). (It can be a poor bound, of course.)

5.1 Noisy subgradient of expected function value

Suppose $F : \mathbf{R}^n \times \mathbf{R}^p \to \mathbf{R}$, and $F(x, w)$ is convex in x for each w. We define

$$f(x) = \mathbf{E}\, F(x, w) = \int F(x, w)p(w)\, dw,$$

where p is the density of w. (The integral is in $\mathbf{R}^p$.) The function f is convex. We'll show how to compute a noisy unbiased subgradient of f at x.

The function f comes up in many applications. We can think of x as some kind of design variable to be chosen, and w as some kind of parameter that is random, *i.e.*, subject to statistical fluctuation. The function F tells us the cost of choosing x when w takes a particular value; the function f, which is deterministic, gives the average cost of choosing x, taking the statistical variation of w into account. Note that the dimension of w, how it enters F, and the distribution are not restricted; we only require that for each value of w, F is convex in x.

Except in some very special cases, we cannot easily compute $f(x)$ exactly. However, we can approximately compute f using Monte Carlo methods, if we can cheaply generate samples of w from its distribution. (This depends on the distribution.) We generate M independent samples $w_1, \ldots, w_M$, and then take

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

as our estimate of $f(x)$. We hope that if M is large enough, we get a good estimate. In fact, $\hat{f}(x)$ is a random variable with $\mathbf{E}\, \hat{f}(x) = f(x)$, and a variance equal to c/M, where c is the variance of $F(x, \omega)$. If we know or bound c, then we can at least bound the probability of a given level of error, *i.e.*, $\mathbf{Prob}(|\hat{f}(x) - f(x)| \geq \epsilon)$. In many cases it's possible to carry out a much more sophisticated analysis of the error in Monte Carlo methods, but we won't pursue that here. A summary for our purposes is: we cannot evaluate $f(x)$ exactly, but we can get a good approximation, with (possibly) much effort.

Let $G : \mathbf{R}^n \times \mathbf{R}^p \to \mathbf{R}^n$ be a function that satisfies

$$G(x, w) \in \partial_x F(x, w)$$

for each x and w. In other words, $G(x, w)$ selects a subgradient for each value of x and w. If $F(x, w)$ is differentiable in x then we must have $G(x, w) = \nabla_x F(x, w)$.

We claim that

$$g = \mathbf{E}\, G(x, w) = \int G(x, w)p(w)\, dw \in \partial f(x).$$

To see this, note that for each w and any z we have

$$F(z, w) \geq F(x, w) + G(x, w)^T(z - x),$$

since $G(x,w) \in \partial_x F(x,w)$. Multiplying this by $p(w)$, which is nonnegative, and integrating gives

$$\begin{aligned}\int F(z,w)p(w)\,dw &\geq \int \left(F(x,w) + G(x,w)^T(z-x)\right)p(w)\,dw \\ &= f(x) + g^T(z-x).\end{aligned}$$

Since the lefthand side is $f(z)$, we've shown $g \in \partial f(x)$.

Now we can explain how to compute a noisy unbiased subgradient of f at x. Generate independent samples $w_1, \ldots, w_M$. We then take

$$\tilde{g} = \frac{1}{M}\sum_{i=1}^{M} G(x, w_i).$$

In other words, we evaluate a subgradient of f, at x, for M random samples of w, and take $\tilde{g}$ to be the average. At the same time we can also compute $\hat{f}(x)$, the Monte Carlo estimate of $f(x)$. We have $\mathbf{E}\,\tilde{g} = \mathbf{E}\,G(x,w) = g$, which we showed above is a subgradient of f at x. Thus, $\tilde{g}$ is a noisy unbiased sugradient of f at x.

This result is independent of M. We can even take $M = 1$. In this case, $\tilde{g} = G(x, w_1)$. In other words, we simply generate *one* sample w_1 and use the subgradient of F for that value of w. 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.

5.2 Example

We consider the problem of minimizing the expected value of a piecewise-linear convex function with random coefficients,

$$\text{minimize} \quad f(x) = \mathbf{E}\max_{i=1,\ldots,m}(a_i^T x + b_i),$$

with variable $x \in \mathbf{R}^n$. The data vectors $a_i \in \mathbf{R}^n$ and $b_i \in \mathbf{R}$ are random with some given distribution. We can compute an (unbiased) approximation of $f(x)$, and a noisy unbiased subgradient $g \in \partial f(x)$, using Monte Carlo methods.

We consider a problem instance with $n = 20$ variables and $m = 100$ terms. We assume that $a_i \sim \mathcal{N}(\bar{a}_i, 5I)$ and $b \sim \mathcal{N}(\bar{b}, 5I)$. The mean values $\bar{a}_i$ and $\bar{b}$ are generated from unit normal distributions (and are the same as the constant values used called a_i and b_i in previous examples). We take $x^{(1)} = 0$ as the starting point and use the step size rule $\alpha_k = 1/k$.

We first compare the solution of the stochastic problem x_{stoch} (obtained from the stochastic subgradient method) with x_{ce}, the solution of the certainty equivalent problem

$$\text{minimize} \quad f_{\text{ce}}(x) = \max_{i=1,\ldots,m}(\mathbf{E}\,a_i^T x + \mathbf{E}\,b_i)$$

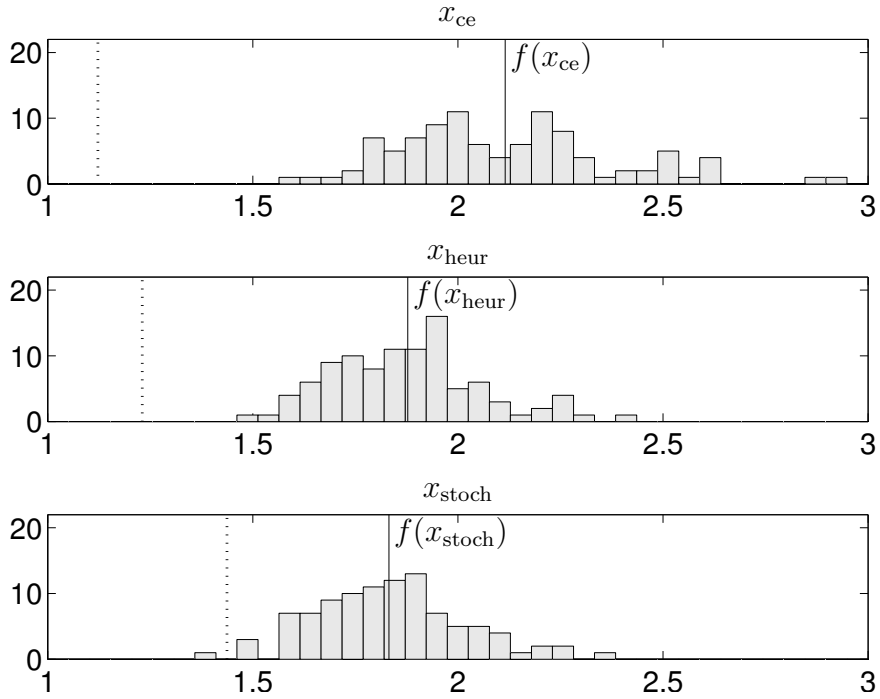

Figure 4: Empirical distributions of $\max_i(a_i^T x + b_i)$ for the certainty equivalent solution x_{ce}, heuristic-based solution x_{heur}, and the stochastic optimal solution x_{stoch}. The dark lines show the value of $f(x)$, and the dotted lines show the value of $f_{\mathrm{ce}}(x)$.

(which is the same piecewise-linear minimization problem considered in earlier examples). We also compare it with x_{heur}, the solution of the problem

$$\text{minimize} \quad f_{\mathrm{heur}}(x) = \max_{i=1,\ldots,m}(\mathbf{E}\, a_i^T x + \mathbf{E}\, b_i + \lambda\|x\|_2),$$

where λ is a positive parameter. The extra terms are meant to account for the variation in $a_i^T x + b$ caused by variation in a_i; the problem above can be cast as an SOCP and easily solved. We chose $\lambda = 1$, after some experimentation.

The certainty equivalent values for the three points are $f_{\mathrm{ce}}(x_{\mathrm{ce}}) = 1.12$, $f_{\mathrm{ce}}(x_{\mathrm{heur}}) = 1.23$, and $f_{\mathrm{ce}}(x_{\mathrm{stoch}}) = 1.44$. We (approximately) evaluate $f(x)$ for these three points, based on 1000 Monte Carlo samples, obtaining $f(x_{\mathrm{ce}}) \approx 2.12$, $f(x_{\mathrm{heur}}) \approx 1.88$, and $f(x_{\mathrm{stoch}}) \approx 1.83$. The empirical distributions of $\max_i(a_i^T x + b_i)$, for $x = x_{\mathrm{stoch}}$, $x = x_{\mathrm{heur}}$, and for $x = x_{\mathrm{ce}}$, are shown in figure 4.

In summary, the heuristic finds a point that is good, but not quite as good as the stochastic optimal. Both of these points are much better than the certainty equivalent point.

Now we show the convergence of the stochastic subgradient method, evaluating noisy subgradients with $M = 1$, $M = 10$, $M = 100$, and $M = 1000$ samples at each step. For

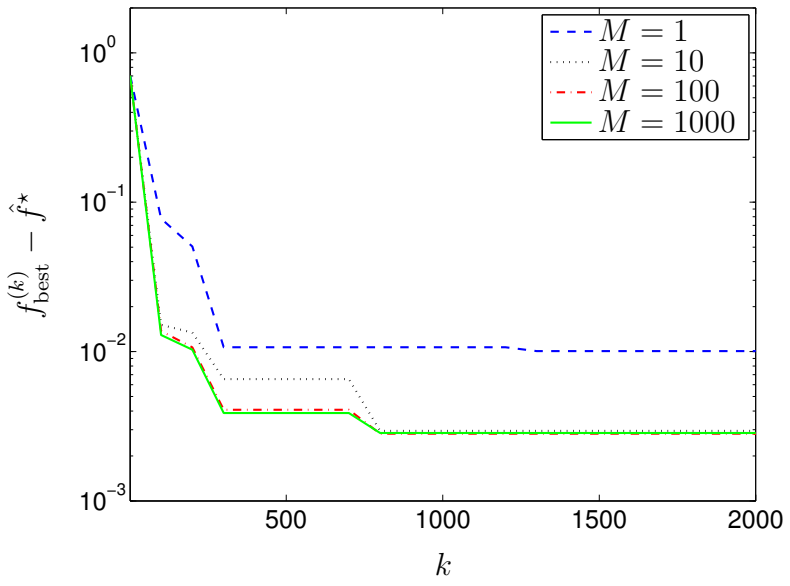

Figure 5: The value of $f_{\text{best}}^{(k)} - f^\star$ versus iteration number k, for the stochastic subgradient method with step size rule $\alpha_k = 1/k$. The plot shows one realization for noisy subgradients evaluated using $M = 1$, $M = 10$, $M = 100$, and $M = 1000$.

$M = 1000$, we are computing a fairly accurate subgradient; for $M = 1$, the variance in our computed subgradient is large.

We (approximately) evaluate the function $f(x)$ using $M = 1000$ samples at each iteration, and keep track of the best value $f_{\text{best}}^{(k)}$. We estimate $f^\star$ by running the stochastic subgradient algorithm for a large number of iterations. Figure 5 shows the convergence for one realization for each of the four values of M. As expected, convergence is faster with M larger, which yields a more accurate subgradient. Assuming the cost of an iteration is proportional to M, $M = 1$ seems to be the best choice. In any case, there seems little advantage (at least in this example) to using a value of M larger than 10.

6 More examples

6.1 Minimizing expected maximum violation

The vector $x \in \mathbf{R}^n$ is to be chosen subject to some (deterministic) linear inequalities $Fx \preceq g$. These can represent manufacturing limits, cost limits, etc. Another set of *random* inequalities is given as $Ax \preceq b$, where A and b come from some distribution. We would like these inequalities to hold, but because A and b vary, there may be no choice of x that results in $Ax \preceq b$ almost surely. We still would like to choose x so that $Ax \preceq b$ holds often, and when it is violated, the violation is small.

Perhaps the most natural problem formulation is to maximize the yield, defined as $\mathbf{Prob}(Ax \preceq b)$. In some very special cases (*e.g.*, A is deterministic and b has log-concave density), this can be converted to a convex problem; but in general it is not convex. Also, yield is not sensitive to how much the inequalities $Ax \preceq b$ are violated; a small violation is the same as a large variation, as far as the yield is concerned.

Instead, we will work with the *expected maximum violation* of the inequalities. We define the *maximum violation* as

$$\max((Ax-b)_+) = \max_i (a_i^T x - b)_+,$$

where a_i are the rows of A. The maximum violation is zero if and only if $Ax \preceq b$; it is positive otherwise. The expected violation is

$$\mathbf{E} \max((Ax-b)_+) = \mathbf{E}\left(\max_i (a_i^T x - b_i)_+\right).$$

It is a complicated but convex function of x, and gives a measure of how often, and by how much, the constraints $Ax \preceq b$ are violated. It is convex no matter what the distribution of A and b is.

As an aside, we note that many other interesting measures of violation could also be used, such as the expected total (or average) violation, $\mathbf{E}\,\mathbf{1}^T(a_i^T x - b_i)_+$, or the expected sum of the squares of the individual violations, $\mathbf{E}\,\|(Ax-b)_+\|_2^2$. We can even measure violation by the expected distance from the desired polyhedron, $\mathbf{E}\,\mathbf{dist}(x, \{z \mid Az \preceq b\})$, which we call the expected violation distance.

Back to our main story, and using expected (maximum) violation, we have the problem

$$\begin{array}{ll} \text{minimize} & \mathbf{E} \max((Ax-b)_+) = \mathbf{E}\left(\max_i (a_i^T x - b_i)_+\right) \\ \text{subject to} & Fx \preceq g. \end{array}$$

The data are F, g, and the distribution of A and b.

We'll use a stochastic projected subgradient method to solve this problem. Given a point x that satisfies $Fx \preceq g$, we need to generate a noisy unbiased subgradient $\tilde{g}$ of the objective. To do this we first generate a sample of A and b. If $Ax \preceq b$, we can take $\tilde{g} = 0$. (Note: if a subgradient is zero, it means we're done; but that's not the case here.) In the stochastic subgradient algorithm, $\tilde{g} = 0$ means we don't update the current value of x. But we don't stop the algorithm, as we would in a deterministic projected subgradient method.

If $Ax \preceq b$ is violated, choose j for which $a_j^T x - b_j = \max_i (a_i^T x - b_i)_+$. Then we can take $\tilde{g} = a_j$. We then set $x_{\text{temp}} = x - \alpha_k a_j$, where α_k is the step size. Finally, we project x_{temp} back to the feasible set to get the updated value of x. (This can be done by solving a QP.)

We can generate a number of samples of A and b, and use the method above to find an unbiased noisy subgradient for each case. We can use the average of these as $\tilde{g}$. If we take enough samples, we can simultaneously estimate the expected maximum violation for the current value of x.

A reasonable starting point is a point well inside the certainty equivalent inequalities $\mathbf{E}\,Ax \preceq \mathbf{E}\,b$, that also satisfies $Fx \preceq g$. Simple choices include the analytic, Chebyshev, or maximum volume ellipsoid centers, or the point that maximizes the margin $\max(b - Ax)$, subject to $Fx \preceq g$.

6.2 On-line learning and adaptive signal processing

We suppose that $(x, y) \in \mathbf{R}^n \times \mathbf{R}$ have some joint distribution. Our goal is to find a weight vector $w \in \mathbf{R}^n$ for which $w^T x$ is a good estimator of y. (This is linear regression; we can add an extra component to x that is always one to get an affine estimator of the form $w^T x + v$.) We'd like to choose a weight vector that minimizes

$$J(w) = \mathbf{E}\, l(w^T x - y),$$

where $l : \mathbf{R} \to \mathbf{R}$ is a convex *loss function*. For example, if $l(u) = u^2$, J is the mean-square error; if $l(u) = |u|$, J is the mean-absolute error. Other interesting loss functions include the Huber function, or a function with dead zone, as in $l(u) = \max\{|u| - 1, 0\}$. For the special case $l(u) = u^2$, there is an analytic formula for the optimal w, in terms of the mean and covariance matrix of (x, y). But we consider here the more general case. Of course J is convex, so minimizing J over w is a convex optimization problem.

We consider an on-line setting, where we do not know the distribution of (x, y). At each step (which might correspond to time, for example), we are given a sample $(x^{(i)}, y^{(i)})$ from the distribution. After k steps, we could use the k samples to estimate the distribution; for example, we could choose w to minimize the average loss under the empirical distribution. This approach requires that we store all past samples, and we need to solve a large problem each time a new sample comes in.

Instead we'll use the stochastic subgradient method, which is really simple. In particular, it requires essentially no storage (beyond the current value of w), and very little computation. The disadvantage is that it is slow.

We will carry out a stochastic subgradient step each time a new sample arrives. Suppose we are given a new sample $(x^{(k+1)}, y^{(k+1)})$, and the current weight value is $w^{(k)}$. Form

$$l'(w^{(k)T} x^{(k+1)} - y^{(k+1)}) x^{(k+1)},$$

where l' is the derivative of l. (If l is nondifferentiable, substitute any subgradient of l in place of l'.) This is a noisy unbiased subgradient of J.

Our on-line algorithm is very simple: when $(x^{(k+1)}, y^{(k+1)})$ becomes available, we update $w^{(k)}$ as follows:

$$w^{(k+1)} = w^{(k)} - \alpha_k l'(w^{(k)T} x^{(k+1)} - y^{(k+1)}) x^{(k+1)}. \tag{3}$$

Note that $w^{(k)T} x^{(k+1)} - y^{(k+1)}$ is the prediction error for the $k+1$ sample, using the weight from step k. We can choose (for example) $\alpha_k = 1/k$.

In signal processing, updates of the form (3) are widely used in *adaptive signal processing*, typically with $l(u) = u^2$. In this case the update (3) is called the *LMS* (least mean-square) algorithm.

We illustrate the method with a simple example with $n = 10$, with $(x, y) \sim \mathcal{N}(0, \Sigma)$, where Σ is chosen randomly, and $l(u) = |u|$. (For the problem instance, we have $\mathbf{E}(y^2) \approx 12$.) The update has the simple form

$$w^{(k+1)} = w^{(k)} - \alpha_k\, \mathbf{sign}(w^{(k)T} x^{(k+1)} - y^{(k+1)}) x^{(k+1)}.$$

We take $\alpha_k = 1/k$. (In adaptive signal processing, this is called a *sign algorithm*.)

We compute $w^\star$ using the stochastic subgradient algorithm, which we run for 5000 iterations. Figure 6 shows the behaviour of the prediction error $w^{(k)T}x^{(k+1)} - y^{(k+1)}$ for the first 300 iterations of the run. Figure 7 shows the empirical distibution (over the 1000 realizations) of the prediction errors for $w^\star$.

6.3 Large datasets and Monte Carlo methods

As an extension of the preceding section, we may revisit the stochastic programming problem (1) and consider performing the stochastic (sub)gradient method in this situation. It is in these problems that stochastic subgradient methods, and more generally subgradient methods, really become effective approaches to the solution of large-scale optimization problems. It is perhaps intuitive that this might be expected: while the subgradient method by itself is quite slow, our convergence arguments suggest that its performance barely degrades even when there is substantial noise in the computation of approximate subgradients $\tilde{g}$.

As a particular special case, consider the situation in which we have a large sample, which we represent as pairs (a_i, b_i), and have objective

$$f(x) = \frac{1}{m} \sum_{i=1}^{m} F(x; (a_i, b_i)),$$

where for each pair (a, b), the objective $F(x; (a, b))$ is convex in $x \in \mathbf{R}^n$. In this situation, if we choose any $m_0 \leq m$ as a batch size, we may construct a stochastic subgradient by taking a subsample of indices $i_1, \ldots, i_{m_0}$ uniformly at random, either with or without replacement, from $\{1, \ldots, m\}$, then setting

$$\tilde{g} \in \frac{1}{m_0} \sum_{j=1}^{m_0} \partial_x F(x; (a_{i_j}, b_{i_j})).$$

It is clear that $\mathbf{E}\, \tilde{g} \in \partial f(x)$ with this choice. The interesting, though obvious, aspect of this is that if $m_0 \ll m$, then it is m/m_0-times more efficient to compute the noisy unbiased subgradient $\tilde{g}$ as opposed to the full subgradient of the objective. The power of stochastic subgradient methods comes from the fact that, as often we only care about getting a moderately accurate solution, we can *often* take $m_0 \ll m$ to compute stochastic subgradients, use these in the iteration of the methods, and achieve substantial computational benefits.

Here we work in reasonable detail through one such instance of this. We generate data pairs as follows. We fix a vector $w^\star \in \mathbf{R}^n$, chosen uniformly from the unit sphere (so that $\|w^\star\| = 1$), and then generate vectors $a_i \stackrel{\text{iid}}{\sim} \mathcal{N}(0, I)$, the normal distribution on $\mathbf{R}^n$. We then set $b_i = a_i^T w^\star + \xi_i$ for ξ_i drawn according to a Cauchy distribution. For our objective, we take the absolute error

$$F(x; (a, b)) = |a^T x - b|.$$

With this setup, our objective becomes $f(x) = \frac{1}{m} \|Ax - b\|_1$, the mean ℓ_1-error in our predictions, which is a natural choice for heavy-tailed data.

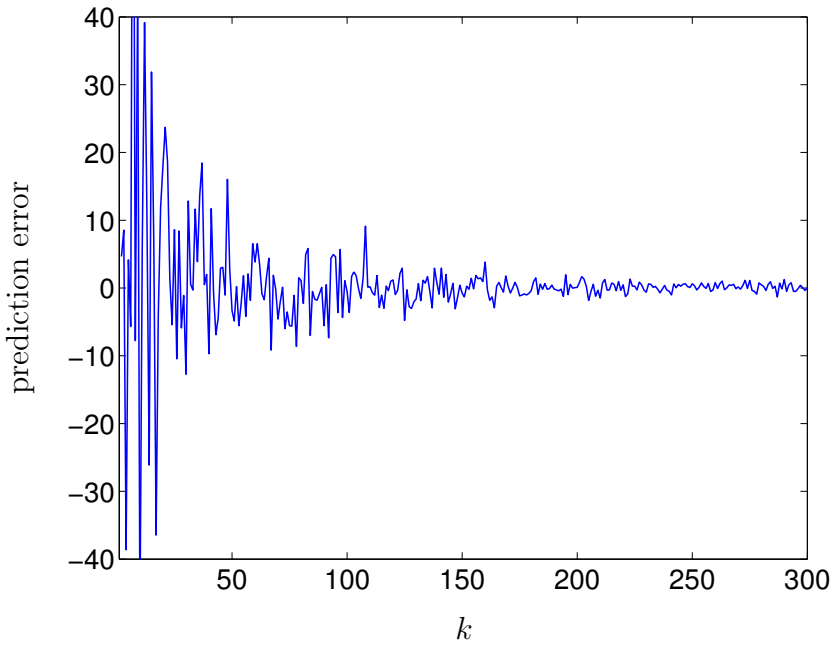

Figure 6: Prediction error $w^{(k)T}x^{(k+1)} - y^{(k+1)}$ versus iteration number k.

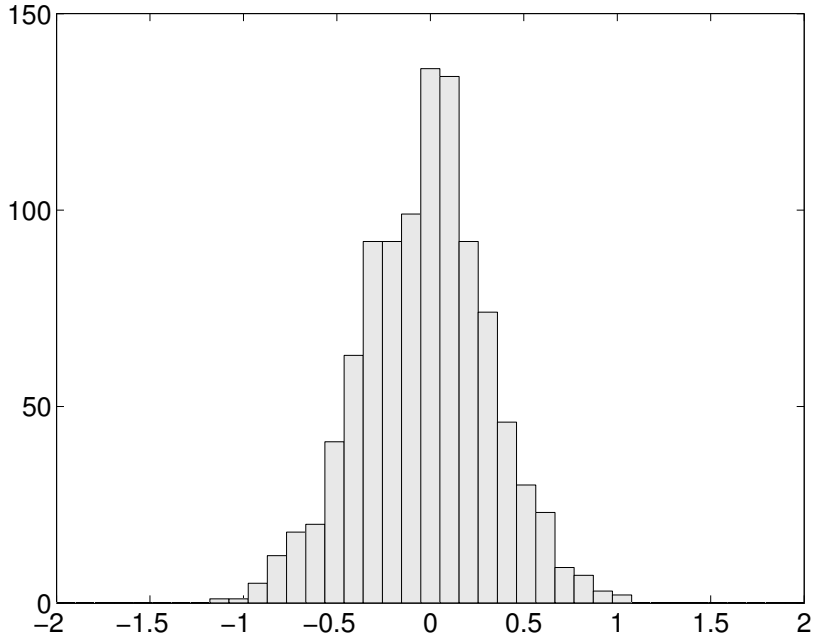

Figure 7: Empirical distribution of prediction errors for $w^\star$, based on 1000 samples.

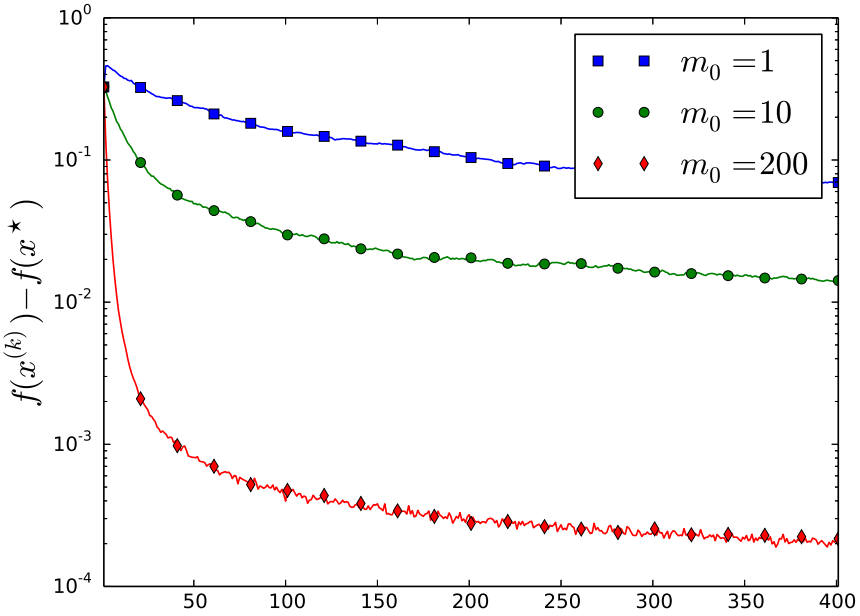

Figure 8: Empirical performance of stochastic subgradient methods using either a single sample ($m_0 = 1$), a sample of ten pairs (a_i, b_i) ($m_0 = 10$), and the full subgradient ($m_0 = m = 200$) for the absolute regression problem. The horizontal axis indexes the number of *iterations* of each method.

With this in mind, we can now compare a projected subgradient method for this problem with $m = 200$ and m_0 chosen from $1, 10$, and 200. Each iteration updates

$$x^{(k+1)} = \Pi(x^{(k)} - \alpha_k \tilde{g}^{(k)})$$

where Π denotes projection onto the ℓ_2-ball of radius 1. We give two figures. In the first (Fig. 8), we show the optimality gap $f(x^{(k)}) - f(x^\star)$ versus iteration, averaged over 50 different experiments. For each method, we used stepsize $\alpha_k = \frac{\alpha}{\sqrt{k}}$, where the scalar α was chosen to yield the best performance in terms of optimality gap. On a per-iteration basis, it is clear that the subgradient method using the full sample size $m_0 = 200$ has the best convergence as a function of the iterations.

On the other hand, when we normalize by the amount of actual *computation* performed, the story is quite different. In Figure 9 we plot the optimality gap $f(x^{(k)}) - f^\star$ against the total amount of computation each method requires, where computation is given by the total number of computations of individual subgradients $\partial_x F(x; (a_i, b_i)) = a_i \, \mathbf{sign}(a_i^T x - b_i)$, each of which requires $O(n)$ time. From this figure, we see that in the time the subgradient method requires to make even a small amount of progress, both subsampled methods have essentially made all of the progress they will make.

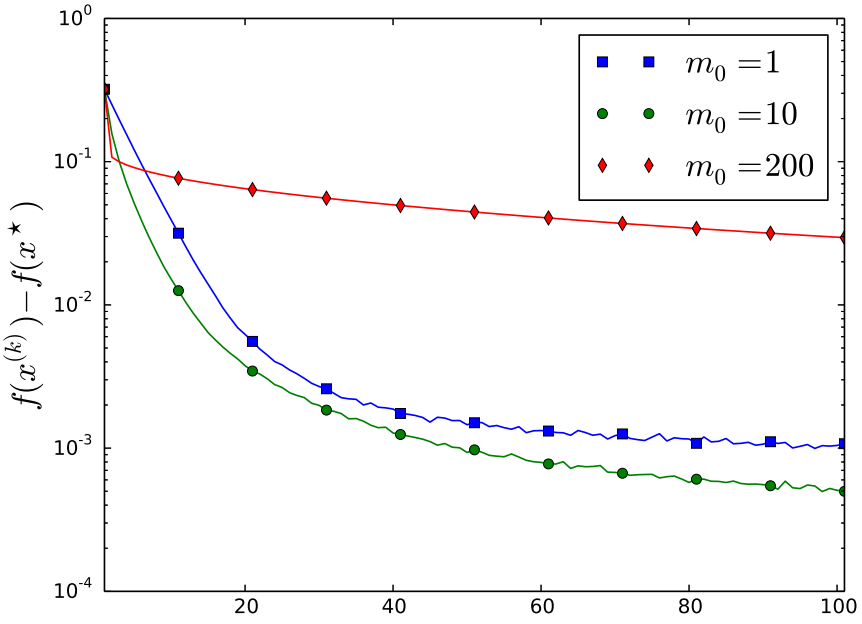

Figure 9: Empirical performance of stochastic subgradient methods using either a single sample ($m_0 = 1$), a sample of ten pairs (a_i, b_i) ($m_0 = 10$), and the full subgradient ($m_0 = m = 200$) for the absolute regression problem. The horizontal axis index the number of *computations* of each method. Thus, for the single sample $m_0 = 1$ case, each tick mark indicates a total of m subgradient steps; for $m_0 = 10$, each tick indicates $m/m_0 = 20$ subgradient steps.

Acknowledgments

May Zhou helped with a preliminary version of this document. We thank Abbas El Gamal for helping us simplify the convergence proof. Trevor Hastie suggested the on-line learning example.

References

[BL97] J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer, New York, 1997.

[Mar05] K. Marti. *Stochastic Optimization Methods*. Springer, 2005.

[Pol87] B. Polyak. *Introduction to Optimization*. Optimization Software, Inc., 1987.

[Pre95] A. Prekopa. *Stochastic Programming*. Kluwer Academic Publishers, 1995.

[Sho98] N. Shor. *Nondifferentiable Optimization and Polynomial Problems*. Nonconvex Optimization and its Applications. Kluwer, 1998.