Chapter 5

Stochastic Gradient Methods

The stochastic gradient method (SGM) is one of the most popular algorithms in modern data analysis and machine learning. It has a long history, with variants having been invented and reinvented several times by different communities, under such names as "least mean squares," "back propagation," "online learning," and the "randomized Kaczmarz method." Most people attribute the stochastic gradient approach to the 1951 work of Robbins and Monro [27], who were interested in devising efficient algorithms for computing random means and roots of scalar functions for which only noisy values are available. In this chapter, we explore some of the properties and implementation details of the SGM.

As before, our goal is to minimize the multivariate convex function $f : \mathbb{R}^d \to \mathbb{R}$, which we assume to be smooth for purposes of this discussion. Extension to the nonsmooth convex case is straightforward, and left as an exercise in the chapter on nonsmooth methods. The SGM differs from methods of Chapters 3 and 4 in the kind of information that is available about f. In place of an exact value of $\nabla f(x)$, we assume that we can compute or acquire a vector $g(x, \xi) \in \mathbb{R}^d$, which is a function of a random variable ξ as well as x, such that

$$\nabla f(x) = \mathbb{E}_\xi [g(x, \xi)]. \tag{5.1}$$

We assume that ξ belongs to some space Ξ with distribution P, and $\mathbb{E}_\xi$ denotes the expectation taken over $\xi \in \Xi$ according to distribution P. The SGM proceeds by substituting $g(x, \xi)$ for the true gradient ∇f in the steepest-descent update formula, so each iteration is as follows:

$$x^{k+1} = x^k - \alpha_k g(x^k, \xi^k), \tag{5.2}$$

where the random variable ξ^k is chosen according to the distribution P (independently of the choices at other iterations) and $\alpha_k > 0$ is the steplength. The method steps in a direction that *in expectation* equals the steepest descent direction. Although $g(x^k, \xi^k)$ may differ substantially from $\nabla f(x^k)$ — it may contain a lot of "noise" — it also contains enough "signal" to make progress toward the optimal value of f over the long term. In typical applications, computation of the gradient estimate $g(x^k, \xi^k)$ is much cheaper than computation of the true gradient $\nabla f(x^k)$.

The choice of steplength α_k is critical to the theoretical and practical behavior of the SGM. We cannot expect to match the performance of steepest descent, in which we move along the true negative gradient direction $-\nabla f(x^k)$ rather than its noisy approximation $-g(x^k, \xi^k)$. In the steepest descent method, the constant steplength $\alpha_k \equiv 1/L$ (where L is the Lipschitz constant for ∇f) yields convergence; see Chapter 3. We can show that this constant-steplength choice will not yield the same convergence properties in the stochastic gradient context, by considering what happens if we initialize the method at the minimizer of f, that is, $x^0 = x^*$. Since $\nabla f(x^*) = 0$, there are no descent directions, and the methods of Chapter 3 will generate a zero step — as they should, since we are already at a solution. The stochastic gradient direction $g(x^0, \xi^0)$ may however be nonzero, causing the SGM to step away from the solution (and increase the objective). We can show however that for judicious choice of the steplength sequence $\{\alpha_k\}$, the sequence $\{x^k\}$ converges to x^*, or at least to a neighborhood of x^*, at rates that are typically slower than those achieved by (true-)gradient descent.

5.1 Examples and Motivation

There are many situations in which the SGM is a powerful tool. Here we discuss a few motivating examples that drive our subsequent implementation details and theoretical analyses.

5.1.1 Noisy Gradients

The simplest application of the SGM is to the case when the gradient estimate $g(x, \xi)$ is the true gradient with additive noise, that is,

$$g(x, \xi) = \nabla f(x) + \xi, \tag{5.3}$$

where ξ is some noise process. The unbiasedness property (5.1) will hold provided that $\mathbb{E}(\xi) = 0$. Our analysis below reveals a protocol for choosing step sizes α_k so that the SGM (5.4) converges. Formula (5.4) reduces in this case to

$$x^{k+1} = x^k - \alpha_k(\nabla f(x^k) + \xi^k). \tag{5.4}$$

5.1.2 Incremental Gradient Method

The incremental gradient method, also known as the perceptron or back-propagation, is one of the most common variants of the SGM. Here we assume that f has the form of a finite sum, that is,

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x). \tag{5.5}$$

where n is usually very large. Computing a full gradient ∇f generally requires computation of ∇f_i, $i = 1, 2, \ldots, n$ — a computation that scales proportionally to n in general. Iteration k of the incremental gradient procedure selects some index i_k from $\{1, \ldots, n\}$ and computes

$$x^{k+1} = x^k - \alpha_k \nabla f_{i_k}(x^k).$$

That is, we choose one of the functions f_i and follow its negative gradient. The standard incremental gradient method chooses i_k to cycle through the components $\{1, 2, \ldots, n\}$ in order, that is, we set $i_k = (k \mod n) + 1$ for $k = 0, 1, 2, \ldots$. An alternative is to choose i_k according to some random procedure at each iteration. The latter approach is a special case of stochastic gradient. We see this by defining the random variable space Ξ to be the set of indices $\{1, 2, \ldots, n\}$, and the choice of random variable ξ^k at iteration k is an index i_k, so that $g(x^k, \xi^k) = \nabla f_{i_k}(x^k)$. Here, the distribution P is such that $P(i) = 1/n$ for all $i = 1, 2, \ldots, n$. The unbiasedness property (5.1) holds, since

$$E_\xi(g(x, \xi)) = \frac{1}{n} \sum_{i=1} \nabla f_i(x) = \nabla f(x).$$

The convergence analysis of the randomized incremental gradient / stochastic gradient method is straightforward, as we will see. Surprisingly, analysis of the original variant of incremental gradient, in which indices i_k are chosen in a deterministic, cyclic order, is more challenging, and the convergence guarantees are weaker.

5.1.3 Classification and the Perceptron

Classification is a canonical problem in machine learning. We are provided data consisting of pairs (x_i, y_i), with $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$ for $i = 1, \ldots n$. The goal is to find a vector $w \in \mathbb{R}^d$ such that

$$w^T x_i > 0 \text{ for } y_i = 1, \quad w^T x_i < 0 \text{ for } y_i = -1.$$

Ideally, w defines a line with all positive examples on one side and all negative examples on the other. (Often, the division is not so clean — there is no line that perfectly separates the two classes — but we can still search for a w that most nearly achieves this goal.)

A popular algorithm for finding w called the *perceptron* was invented in the 1950s. It uses one example at a time to generate a sequence $\{w^k\}$, $k = 1, 2, \ldots$ from some starting point w^0. At iteration k, we choose one of our data pairs (x_{i_k}, y_{i_k}) and update according to the formula

$$w^{k+1} = (1 - \gamma)\, w^k + \begin{cases} \eta y_{i_k} x_{i_k} & \text{if } y_{i_k} (w^k)^T x_{i_k} < 1 \\ 0 & \text{otherwise,} \end{cases} \tag{5.6}$$

for some positive parameters γ and η. The idea behind this iteration is that if the current guess w^k classifies the pair (x_{i_k}, y_{i_k}) incorrectly, then we "nudge" w^k in a way that makes $(w^k)^T x_{i_k}$ closer to the correct sign. If w^k produces correct classification on this example, we do not change it.

This method is an instance of the SGM. A quick calculation shows that this procedure is obtained by applying the SGM to the cost function

$$\frac{1}{n} \sum_{i=1}^{n} \max\left(1 - y_i x_i^T w, 0\right) + \frac{\lambda}{2} \|w\|_2^2 \,, \tag{5.7}$$

where ξ_k is the index i_k of a single term from the summation. In the update equation (5.6), we have chosen $\gamma = (\eta\lambda)/n$ where $\alpha_k \equiv \eta$ is the stepsize parameter. (In machine learning, the stepsize is often referred to as the *learning rate*.) The cost function (5.7) is often called the *Support Vector Machine*. In the parlance of our times, the perceptron algorithm is equivalent to "training" a support vector machine using the SGM.

5.1.4 Empirical Risk Minimization

In machine learning, the Support Vector Machine is one of many instances of the class of optimization problems called *Empirical Risk Minimization*. Many classification, regression, and decision tasks can be evaluated as expected values of error over the data-generating distributions. The most common example is known as *statistical risk*. Given a data generating distribution $p(x, y)$, and a *loss function* $\ell(u, v)$ we define the risk as

$$R[f] := \mathbb{E}[\ell(f(x), y)], \tag{5.8}$$

where the expectation is over the data space (x, y) according to probability distribution $p(x, y)$. The function ℓ measures the cost of assigning the value $f(x)$ when the quantity to be estimated is y. (Typically ℓ becomes larger when $f(x)$ deviates further from y.) The quantity R is the expected loss of the decision rule $f(x)$ with respect to the probability distribution $p(x, y)$ of the data. The goal of many learning tasks is to choose the function f that minimizes the risk. For example, the Support Vector Machine uses a *hinge loss* for the function ℓ, that measures the distance between the prediction $w^T x$ and the correct half-space. In regression problems, y is a target variate, and the loss measures the difference between $f(x)$ and y according to the square function $\ell(f(x), y) = (1/2)(f(x) - y)^2$.

Often, minimization of the risk is computationally intractable and depends strongly on knowing the likelihood and prior models for the data pairs (x, y). A popular alternative uses *samples* to provided an estimate for the true risk. Suppose we have a process that generates independent, identically distributed (i.i.d.) samples $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ from the joint distribution $p(x, y)$. For these data points and a fixed decision rule $\hat{x}(y)$, we can expect that *the empirical risk*

$$R_{\text{emp}}[f] := \frac{1}{n} \sum_{i=1}^{n} \ell(f(y_i), x_i) \tag{5.9}$$

is "close" to the true risk. Indeed, $R_{\text{emp}}[f]$ is a random variable equal to the sample mean of the loss function. If we take the expectation with respect to our samples, we have

$$\mathbb{E}[R_{\text{emp}}[f]] = R[f]\,.$$

Given these samples, the empirical risk is no longer a function of the likelihood and prior models. We have obtained a simpler optimization problem, in which the objective is a finite sum — a form like that of (5.5). Minimizing this empirical risk corresponds to finding the best function f that minimizes the average loss over our data sample.

The SGM and empirical risk minimization (ERM) are intimately related. One variant of ERM formulates the problem finitely as (5.9) and then applies the randomized incremental gradient approach of Section 5.1.2 to this function. Another variant does not explicitly take a finite data sample, instead applying the SGM directly to (5.8). At each step, a pair (x, y) is sampled according to the distribution $p(x, y)$, and a step is taken along the negative gradient of loss function ℓ with respect to f, evaluated at the point $(f(x), y)$. [1]

The perceptron algorithm is a particular instance of ERM, in which we define $f(x) = w^T x$ (so that f is parametrized by the vector w) and $\ell(f(x), y) = \max(1 - yx^T w, 0)$.

5.2 Insights into Randomness and Steplength

Before turning to a rigorous analysis of the SGM, it will be useful to get some background and insight into how to choose the stepsize parameters. We consider some simple but informative examples.

[1] **SJW:** Is this a fair summary? I am not sure I have the terminology right.

5.2.1 Example: Computing a Mean

Let us consider applying an incremental gradient method to the scalar function

$$f(x) := \frac{1}{2n} \sum_{i=1}^{n} (x - \omega_i)^2 \tag{5.10}$$

where ω_i are n fixed scalars. This function has the form of the finite sum (5.5) when we define $f_i(x) = (1/2)(x - \omega_i)^2$, so that

$$\nabla f_i(x) = x - \omega_i \, .$$

Consider first the deterministic method in which we work through the component functions *in order*, starting with $x^0 = 0$ and using the stepsize $\alpha_k = 1/(k+1)$. The first few iterations are:

$$\begin{aligned} x^1 &= x^0 - (x^0 - \omega_1) = \omega_1, \\ x^2 &= x^1 - \frac{1}{2}\left(x^1 - \omega_2\right) = \frac{1}{2}\omega_1 + \frac{1}{2}\omega_2, \\ x^3 &= x^2 - \frac{1}{3}\left(x^2 - \omega_3\right) = \frac{1}{3}\omega_1 + \frac{1}{3}\omega_2 + \frac{1}{3}\omega_3, \end{aligned}$$

so that

$$x^k = \left(\frac{k-1}{k}\right) x^{k-1} + \frac{1}{k}\omega_k = \frac{1}{k}\sum_{j=1}^{k} \omega_j \, , \quad k = 1, 2, \ldots \, . \tag{5.11}$$

The stepsize $\alpha_k = 1/(k+1)$ was the one originally proposed by Robbins and Monro [27], and it makes perfect sense for this simple example, as it produces iterates that are the running average of all the samples ω_j encountered so far. The $1/k$ step has two other important features.

- Even when the gradients $g(x; i) = \nabla f_i(x)$ are bounded in norm, the iterates can traverse an arbitrary distances across the search space, because $\sum_{k=0}^{\infty} 1/(k+1) = \infty$. Thus, convergence can be obtained even when the starting point x^0 is arbitrarily far from the solution x^*.
- The steplengths shrink to zero, so that when the iterates reach a neighborhood of the solution x^*, they tend to stay there, even though the search directions $g(x; \xi)$ contain noise.

For this simple example, the global minimum of f is found after n steps of the cyclic, incremental method — there is no need for randomness. In fact, the SGM applied to (5.10) is unlikely to converge in a finite number of iterations. There are however some cases in which randomness produces much better performance than cyclic schemes, or other scheme in which deterministic choices are made of the samples ω_i, as we see in the next section.

Let us consider now a "continuous" version of (5.10):

$$f(x) = \tfrac{1}{2}\mathbb{E}_\omega[(x - \omega)^2], \tag{5.12}$$

where ω is some random variable with mean μ and variance σ^2. At step j of the SGM, we select some value ω_{j+1} from the distribution of ω, independently of the choices of ω that were made at previous iterations. We take a step of length $1/(j+1)$ in direction $x^j - \omega_{j+1}$. After k steps, starting from $x^0 = 0$, we have as above that

$$x^k = \frac{1}{k}\sum_{j=1}^{k} \omega_j \, .$$

By plugging this value into (5.12), and taking the expectation over ω and all the random variables $\omega_1, \omega_2, \ldots, \omega_j$, we obtain

$$f(x^k) = \tfrac{1}{2}\mathbb{E}_{\omega_1,\omega_2,\ldots,\omega_k,\omega}\left[\left(\frac{1}{k}\sum_{j=1}^{k}\omega_j - \omega\right)^2\right] = \frac{1}{2k}\sigma^2 + \frac{1}{2}\sigma^2. \tag{5.13}$$

In this simple case too, we can compute the minimizer of (5.12) exactly. We have

$$f(x) = \tfrac{1}{2}\mathbb{E}[x^2 - 2\omega x + \omega^2] = \tfrac{1}{2}x^2 - \mu x + \tfrac{1}{2}\sigma^2 + \tfrac{1}{2}\mu^2\,.$$

Thus the minimizer of f is $x^* = \mu$, with $f(x^*) = \frac{1}{2}\sigma^2$. By comparing this value with (5.13), we have

$$f(x^k) - f(x^*) = \frac{1}{2k}\sigma^2.$$

Statistically speaking, it can be shown that x^k is the highest-quality estimate that can be attained for x^* given the sequence $\{\omega_1, \omega_2, \ldots, \omega_k\}$. Interestingly, the SGM, which considered the samples ω_{j+1} one at a time and made a step after each iteration, is able to achieve the same quality as an estimator that made use of the complete set of data $\{\omega_1, \omega_2, \ldots, \omega_k\}$ at once. Even so, the convergence rate for this best-possible performance is sublinear: The sequence of differences between function values and their optimum $\{f(x^k) - f^*\}$ shrinks like $1/k$, rather than decreasing exponentially to zero. This demonstrates a fundamental limitation of the SGM: Linear convergence cannot be expected in general. *Statistics*, not computation or algorithm design, stands in the way of linear convergence rates.

5.2.2 The (Randomized) Kaczmarz Method

The potential benefits of randomness can be seen when we consider a special case of the following linear least squares problem:

$$\min\ f(x) := \frac{1}{2n}\sum_{i=1}^{n}\left(a_i^T x - b_i\right)^2, \tag{5.14}$$

where $\|a_i\| = 1$, $i = 1, 2, \ldots, n$. Assume that there exists an x^* such that $a_i^T x^* = b_i$ for $i = 1, 2, \ldots, n$. This point will be a minimizer of f, with $f(x^*) = 0$. The SGM with stepsize $\alpha_k \equiv 1$ — known as the *randomized Kaczmarz* method — yields the recursion

$$x^{k+1} = x^k - a_{i_k}\left(a_{i_k}^T x^k - b_{i_k}\right) = x^k - a_{i_k}a_{i_k}^T(x^k - x^*).$$

Aggregating the effects of the first k iterations, we obtain

$$x^{k+1} - x^* = \left(I - a_{i_k}a_{i_k}^T\right)\left(x^k - x^*\right) = \prod_{j=0}^{k}\left(I - a_{i_j}a_{i_j}^T\right)\left(x^0 - x^*\right).$$

Iteration k is a projection of the current iterate x^k onto the plane defined by $a_{i_k}^T x = b_{i_k}$. If two successive subspaces are close to one another, x^{k+1} and x^k are close together, and we do not make

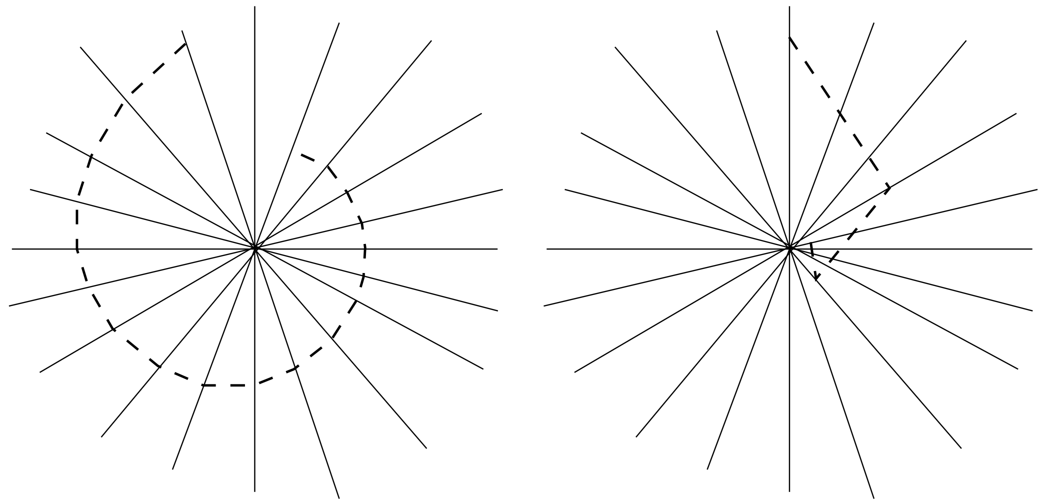

Figure 5.1: Kaczmarz method. Deterministic, ordered choice (left) leads to slow convergence; randomized Kaczmarz (right) converges faster.

much progress toward x^*. The following example describes a set of vectors $\{a_1, a_2, \dots, a_n\}$ such that the sequential ordering of a_{i_k}, $k = 0, 1, 2, \dots$ (that is, $i_0 = 1, i_1 = 2, i_2 = 3, \dots$) yields slow progress, while much faster convergence is attained by making random choices of i_k for each k.

For $n \ge 3$, set $\omega_n := \pi/n$ and define the vectors a_i as follows:

$$a_i = \begin{bmatrix} \cos(i\omega_n) \\ \sin(i\omega_n) \end{bmatrix}, \quad i = 1, 2, \dots, n. \tag{5.15}$$

Define $b_i = 0$, $i = 1, 2, \dots, n$, so that the solution of (5.14) is $x^* = 0$. We have that $\|a_i\| = 1$ for all i, and in addition that $\langle a_i, a_{i+1} \rangle = \cos(\omega_n)$ for $1 \le i \le n-1$. The matrices $A_i := I - a_i a_i^T$ are positive semidefinite for all i, and we have the identity

$$\mathbb{E}_j(A_j) = \frac{1}{n} \sum_{i=1}^n A_i = \tfrac{1}{2} I\,. \tag{5.16}$$

Any set of unit vectors satisfying (5.16) is called a *normalized tight frame*, and the vectors (5.18) form a *harmonic frame*, due to their trigonometric origin.

Consider a *randomized* version of the Kaczmarz method, in which we select the vector a_{i_k} with equal likelihood from among $\{a_1, a_2, \dots, a_n\}$, with the choice made independently at each iteration. The expected decrease in error over iteration k, conditional on the value of x^k, is

$$\mathbb{E}_{i_k}(x^{k+1} - x^* \,|\, x^k) = \left(\mathbb{E}_{i_k}(I - a_{i_k} a_{i_k}^T)\right)(x^k - x^*) = \tfrac{1}{2}(x^k - x^*), \tag{5.17}$$

where we used (5.16) to obtain the fraction of 1/2. The following argument shows exponential decrease of the expected error with rate (1/2) per iteration:

$$\begin{aligned} &\mathbb{E}(x^k - x^0) \\ &= \mathbb{E}_{i_0, i_1, \dots, i_{k-1}} \prod_{j=0}^{k-1} A_{i_j}(x^0 - x^*) = \left[\prod_{j=0}^{k-1} \mathbb{E}_{i_j}(A_{i_j})\right](x^0 - x^*) = \left[\mathbb{E}_{i_j}(A_{i_j})\right]^k (x^0 - x^*) = 2^{-k}(x^0 - x^*)\,. \end{aligned}$$

The critical step of taking the expectation inside the product is possible because of independence of the i_j, $j = 0, 1, \ldots, k-1$.)

The behavior of randomized Kaczmarz is shown in the right diagram in Figure 5.1, with the path traced by the iterations shown as a dotted line.

Why do we attain linear convergence for the randomized method, when the example of the previous subsection attained only a sublinear rate? The answer is that the solution x^* is a fixed point of both a gradient map *and* a stochastic gradient step. That is, both $\nabla f(x)$ and $\nabla f_i(x)$ approach zero as $x \to x^*$, for all $i = 1, 2, \ldots, n$. (For the same reason, we were able to use a large constant stepsize $\alpha_k \equiv 1$ rather than the usual decreasing stepsize.)

The fact that the vectors a_{i_k} are selected randomly, for $k = 0, 1, 2, \ldots$ is also critical to the fast convergence. If we use a deterministic order $i_k = k + 1$, $k = 0, 1, 2, \ldots, n-1$, the convergence analysis is quite different. Define the vectors

$$\hat{a}_i = \begin{bmatrix} \sin(-i\omega_n) \\ \cos(-i\omega_n) \end{bmatrix}, \tag{5.18}$$

and note that $A_i = I - a_i a_i^T = \hat{a}_i \hat{a}_i^T$. We have

$$\prod_{i=1}^{k} A_i = \hat{a}_k \hat{a}_1^T \prod_{j=1}^{k-1} \langle \hat{a}_j, \hat{a}_{j+1} \rangle = \hat{a}_k \hat{a}_1^T \cos^{k-1}(\omega_n) .$$

We therefore have

$$\|x^k - x^*\| = \left\| \prod_{i=1}^{k} \left(I - a_i a_i^T\right) \left(x^0 - x^*\right) \right\| = \cos^{k-1}(\omega_n) \left| \hat{a}_1^T \left(x^0 - x^*\right) \right| .$$

For $x^0 = (0, 1)^T$, we have $\hat{a}_1^T(x^0 - x^*) = \|x^0 - x^*\|$, so

$$\|x^k - x^*\| = \cos(-\pi/n)^{k-1} \|x^0 - x^*\|, \quad k = 0, 1, 2, \ldots, n.$$

This indicates geometric convergence, at a rate of $\cos(-\pi/n) \approx 1 - (1/2)(\pi/n)^2$ per iteration — a much slower rate than the rate of 1/2 achieved in the randomized case. (This analysis is deterministic, unlike that of the randomized case, which yields a bound only on the decrease in the *excepted* error.)

The deterministic variant is plotted in the left diagram of Figure 5.1, which shows a slow spiral toward the solution.

The randomized Kaczmarz method was analyzed some years ago by signal processing researchers, independently of the long standing work on the SGM.

5.3 Convergence Analysis: Key Assumptions

We now turn to convergence analysis of the SGM, applied to the convex function $f : \mathbb{R}^d \to \mathbb{R}$, with steps of the form (5.4) and search directions $g(x, \xi)$ satisfying condition (5.1). To prove convergence, we need to assume some bounds on the sizes of the gradient estimates $g(x, \xi)$, so that the information they contain is not swamped by arbitrarily large amounts of noise. We assume that there are nonnegative constants L_g and B such that

$$\mathbb{E}_\xi \left[\|g(x; \xi)\|_2^2 \right] \le L_g^2 \|x - x^*\|^2 + B^2 \quad \text{for all } x. \tag{5.19}$$

Note that this assumption may be satisfied even when $g(x;\xi)$ is arbitrarily large for some combination of x and ξ; formula (5.19) requires only boundedness *in expectation over* ξ for each x. (Subsection 5.3.3 below contains an example in which ξ is unbounded but (5.19) still holds for suitable choices of L_g and B.)

Note that when $L_g = 0$ in (5.19), f cannot be strongly convex over an unbounded domain. If f were strongly convex function with modulus of convexity m, we would have

$$\|\nabla f(x)\| \geq \frac{m}{2}\|x - x^*\|$$

for all x. On the other hand, we have by Jensen's inequality that

$$\|\nabla f(x)\|^2 \leq \mathbb{E}[\|g(x;\xi)\|^2].$$

These two bounds together imply that it is not possible to find a B for which (5.19) holds with $L_g = 0$, if the domain of f is unbounded.

When f has the finite-sum form (5.5) and we have $\nabla f_{i_k}(x^k)$ as the gradient estimate at iterate x^k, where i_k chosen uniformly at random from $\{1, 2, \ldots, n\}$, as in Subsection 5.1.2, the bound (5.19) specializes to

$$\frac{1}{n}\sum_{i=1}^{n} \|\nabla f_i(x)\|^2 \leq L_g^2\|x - x^*\|^2 + B^2 \quad \text{for all } x. \tag{5.20}$$

The steplengths α_k in the stochastic gradient iteration formula (5.4) typically depend on the constants L_g and B in (5.19). Throughout, we will assume that the sequence $\{\xi_k\}_{k=0,1,2,\ldots}$ needed to generate the gradient approximations $g(x^k, \xi^k)$ is selected i.i.d. from a fixed distribution. (It is possible to weaken the i.i.d. assumptions, but we do not consider such extensions here.)

We now examine how the constants L_g and B appear in different problem settings, including those described in earlier sections.

5.3.1 Case 1: Bounded Gradients: $L_g = 0$

Suppose that the stochastic gradient function $g(\cdot;\cdot)$ is bounded almost surely for all x — that is, $L_g = 0$ in (5.19). This is true for the *logistic regression* objective

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} -y_i w^T x_i + \log(1 + \exp(w^T x_i)) \tag{5.21}$$

where $y_i \in \{0, 1\}$. Following the finite-sum setting (5.5), the random variable ξ is drawn uniformly from the set $\{1, 2, \ldots, n\}$, and

$$g(w;i) = \left(-y_i + \tfrac{\exp(w^T x_i)}{1+\exp(w^T x_i)}\right) x_i$$

Thus (5.19) holds with $L_g = 0$ and $B = \sup_{i=1,2,\ldots,n} \|x_i\|_2$.

5.3.2 Case 2: Randomized Kaczmarz: $B = 0$, $L_g > 0$

Consider the least-squares objective (5.14), where we assume that $a_i \neq 0$ but not necessarily $\|a_i\| = 1$ for each i. Assume that there is x^* for which $f(x^*) = 0$, that is, $a_i^T x^* = b_i$ for all $i = 1, 2, \dots, n$. By substituting into (5.14), we obtain

$$f(x) = \frac{1}{2n} \sum_{i=1}^{n} (x - x^*)^T a_i a_i^T (x - x^*)$$

and, with the random variable ξ being drawn uniformly from $\{1, 2, \dots, n\}$, we have

$$g(x; i) = a_i a_i^T (x - x^*).$$

For the expected norm, we have

$$\mathbb{E}[\|g(x; i)\|^2] = \mathbb{E}[\|a_i\|^2 |a_i^T (x - x^*)|^2] \leq \mathbb{E}[\|a_i\|^4] \|x - x^*\|^2,$$

so that (5.19) can be satisfied by setting $L_g = \mathbb{E}[\|a_i\|^4]^{1/2}$ and $B = 0$.

5.3.3 Case 3: Additive Gaussian Noise

Consider the additive noise model (5.3) where ξ is from the Gaussian distribution with mean zero and covariance $\sigma^2 I$, that is, $\xi \in N(0, \sigma^2 I)$. We have $\mathbb{E}[g(x; \xi)] = \nabla f(x)$ and

$$\mathbb{E}[\|g(x; \xi)\|^2] = \|\nabla f(x)\|^2 + 2 \nabla f(x)^T \mathbb{E}(\xi) + \mathbb{E}(\|\xi\|^2) = \|\nabla f(x)\|^2 + d\sigma^2. \tag{5.22}$$

We can satisfy (5.19) by setting $B = \sigma\sqrt{d}$, and defining L_g to the Lipschitz constant of the gradient of f (because $\|\nabla f(x)\|^2 \leq \|\nabla f(x) - \nabla f(x^*)\|^2 \leq L^2 \|x - x^*\|^2$).

5.3.4 Case 4: Incremental Gradient

Consider the finite-sum formulation (5.5) in which the gradient ∇f_i of each term in the sum has Lipschitz constant L_i. As in Subsection 5.1.2, the distribution for the random variable ξ is discrete with n equally likely choices corresponding to the indices $i = 1, 2, \dots, n$ of the terms in the sum. For the ith term $f_i(x)$, we define x^{*i} to be any point for which $\nabla f_i(x^{*i}) = 0$. We then have

$$\begin{aligned}
\mathbb{E}_\xi[\|g(x; \xi)\|^2] &= \mathbb{E}_i[\|\nabla f_i(x)\|^2] \\
&\leq \mathbb{E}[L_i^2 \|x - x^{*i}\|^2] \\
&\leq \mathbb{E}\left[2L_i^2 \|x - x^*\|^2 + 2L_i^2 \|x^{*i} - x^*\|^2\right] \\
&= \frac{2}{n} \sum_{i=1}^{n} L_i^2 \|x - x^*\|^2 + \frac{2}{n} \sum_{i=1}^{n} L_i^2 \|x^{*i} - x^*\|^2 \,,
\end{aligned}$$

where we used the bound $\|a + b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$. Thus (5.19) holds if we define

$$L_g^2 = \frac{2}{n} \sum_{i=1}^{n} L_i^2, \quad B^2 = \frac{2}{n} \sum_{i=1}^{n} L_i^2 \|x^{*i} - x^*\|^2 \,.$$

There is nice intuition for this choice of B. If $x^{*i} = x^*$ for all i, then $B = 0$, as in the case of Randomized Kaczmarz (Subsection 5.3.2). (The term L_g^2 in the derivation above is also smaller by a factor of 2.)

5.4 Convergence Analysis

Our convergence results track the decrease in certain measures of error as a function of iterations. These measures are of two types. The first is an expected squared error in the point x, that is $\mathbb{E}\left[\|x - x^*\|^2\right]$, where x^* is the solution and the expectation is taken over all the random variables ξ^k encountered to that point of the algorithm. This measure is most appropriate when the objective f is strongly convex, so that the solution x^* is uniquely defined. The second measure of optimality is the gap between the current objective value and the optimal value, that is $f(x) - f(x^*)$. This measure can be used when f is convex but not necessarily strongly convex. In the strongly convex case, each of these two measures can be bounded in terms of the other, with the bound depending on the Lipschitz constant for ∇f and the modulus of convexity m.

We see that the suitable choices of steplengths α_k in (5.4) depend on L_g and B, and that the convergence rates also depend on these two quantities.

We begin by expanding the distance to the optimal solution, as follows:

$$
\begin{aligned}
\|x^{k+1} - x^*\|^2 &= \|x^k - \alpha_k g(x^k;\xi_k) - x^*\|^2 \\
&= \|x^k - x^*\|^2 - 2\alpha_k \langle g(x^k;\xi_k), x^k - x^*\rangle + \alpha_k^2 \|g(x^k;\xi_k)\|^2
\end{aligned}
\tag{5.23}
$$

We deal with each term in this expansion separately. We take the expectation of both sides with respect to all the random variables encountered by the algorithm up to and including iteration k, namely $i_0, i_1, \ldots, i_k$. By applying the law of iterated expectation, and noting that x^k depends on $\xi_0, \xi_1, \ldots, \xi_{k-1}$ *but not on* ξ_k, we obtain

$$
\begin{aligned}
\mathbb{E}[\langle g(x^k;\xi_k), x^k - x^*\rangle] &= \mathbb{E}\left[\mathbb{E}_{\xi_k}[\langle g(x^k;\xi_k), x^k - x^*\rangle \mid \xi_0, \xi_1, \ldots, \xi_{k-1}]\right] \\
&= \mathbb{E}\left[\langle \mathbb{E}_{\xi_k}[g(x^k;\xi_k) \mid \xi_0, \xi_1, \ldots, \xi_{k-1}], x^k - x^*\rangle\right] \\
&= \mathbb{E}\left[\langle \nabla f(x^k), x^k - x^*\rangle\right] .
\end{aligned}
$$

In the last step of this derivation, we used the fact that $g(x^k;\xi_k)$ depends on ξ_k, while x^k does not, so we took the expectation of $g(x^k;\xi_k)$ explicitly with respect to ξ_k, to obtain $\nabla f(x^k)$.

By a similar argument, we can bound the last term in (5.23) by using (5.19):

$$
\mathbb{E}[\|g(x^k;\xi_k)\|_2^2] = \mathbb{E}\left[\mathbb{E}_{\xi_k}[\|g(x^k;\xi_k)\|_2^2 \mid \xi_0, \xi_1, \ldots, \xi_{k-1}]\right] \le \mathbb{E}[L_g^2 \|x^k - x^*\|_2^2 + B^2].
$$

By defining the squared expected error as

$$
a_k := \mathbb{E}[\|x_k - x_\star\|^2], \tag{5.24}
$$

we obtain by substituting these relationships into (5.23) that

$$
a_{k+1} \le (1 + \alpha_k^2 L_g^2) a_k - 2\alpha_k \mathbb{E}\left[\langle \nabla f(x^k), x^k - x^*\rangle\right] + \alpha_k^2 B^2 . \tag{5.25}
$$

Our results follow from different manipulations of (9.2) for different settings of L_g and B. We proceed through several cases.

5.4.1 Case 1: $L_g = 0$.

When $L_g = 0$, the expression (9.2) reduces to

$$a_{k+1} \le a_k - 2\alpha_k \mathbb{E}\left[\langle \nabla f(x^k), x^k - x^* \rangle\right] + \alpha_k^2 B^2 \,. \tag{5.26}$$

Define λ_k to be the sum of all stepsizes up to and including iteration k, and $\bar{x}^k$ to be the average of all iterates so far, weighted by the stepsizes α_j, that is,

$$\lambda_k = \sum_{j=0}^{k} \alpha_j, \quad \bar{x}^k = \lambda_k^{-1} \sum_{j=0}^{k} \alpha_j x^j \,.$$

We analyze the deviation of $f(\bar{x}_k)$ from optimality. Given the initial point x^0, which we assume to now be random, we define $D_0 := \|x^0 - x^*\|$ to be the initial squared error. (Note from (5.24) that $a_0 = D_0^2$.) After T iterations, we have the following estimate for $\bar{x}^T$:

$$\begin{aligned}
\mathbb{E}[f(\bar{x}^T) - f(x^*)] &\le \mathbb{E}\left[\lambda_T^{-1} \sum_{j=0}^{T} \alpha_j (f(x^j) - f(x^*))\right] && (5.27\text{a})\\
&\le \lambda_T^{-1} \sum_{j=0}^{T} \alpha_j \mathbb{E}[\langle \nabla f(x^j), x^* - x^j \rangle] && (5.27\text{b})\\
&\le \lambda_T^{-1} \sum_{j=0}^{T} \left[\tfrac{1}{2}(a_j - a_{j+1}) + \tfrac{1}{2}\alpha_j^2 B^2\right] && (5.27\text{c})\\
&= \frac{1}{2}\lambda_T^{-1} \left[a_0 - a_{T+1} + B^2 \sum_{j=0}^{T} \alpha_j^2\right] \\
&\le \frac{D_0^2 + B^2 \sum_{j=0}^{T} \alpha_j^2}{2 \sum_{j=0}^{T} \alpha_j}. && (5.27\text{d})
\end{aligned}$$

Here, (5.27a) follows from convexity of f and the definition of $\bar{x}^T$; (5.27b) again uses convexity of f; and (5.27c) follows from (5.26).

With the bound (5.27d) in hand, we can prove the following result for the case of constant stepsizes: $\alpha_k \equiv \alpha > 0$ for all k.

Proposition 5.1 ([20]). *Suppose we run the SGM on a convex f with $L_g = 0$ for T steps with constant stepsize $\alpha > 0$. Define*

$$\alpha_{\text{opt}} = \frac{D_0}{B\sqrt{T+1}} \quad \textit{and} \quad \theta := \frac{\alpha}{\alpha_{\text{opt}}}.$$

Then we have the bound

$$\mathbb{E}[f(\bar{x}^T) - f^\star] \le \left(\tfrac{1}{2}\theta + \tfrac{1}{2}\theta^{-1}\right) \frac{BD_0}{\sqrt{T+1}} \,. \tag{5.28}$$

Proof. The proof comes directly from setting $\alpha_j \equiv \alpha = \theta\alpha_{\text{opt}} = \theta\frac{D_0}{B\sqrt{T+1}}$ in (5.27d). We have

$$\mathbb{E}\left[f\left(\bar{x}^T\right) - f(x_\star)\right] \le \frac{D_0^2 + B^2(T+1)\alpha^2}{2(T+1)\alpha} = \left(\tfrac{1}{2}\theta^{-1} + \tfrac{1}{2}\theta\right)\frac{BD_0}{\sqrt{T+1}}.$$

□

The tightest bound is attained when $\theta = 1$, that is, $\alpha = \alpha_{\text{opt}}$. The bound approximately linearly in the error factor in our choice of α. That is, if our α differs by a factor of 2 (in either direction) from α_{opt}, the bound is worse by a factor of approximately 2. This means that to achieve the same bound as with the optimal step size, we need to take about *four times* as many iterations, because the bound also depends on the iteration counter T through a factor of approximately $1/\sqrt{T}$.

Other stepsize schemes could also be selected here, including choices of α_k that decrease with k. But the constant stepsize is optimal for an upper bound of this type.

5.4.2 Case 2: $B = 0$

When $B = 0$, we obtain a *linear* rate of convergence in the expected-error measure a_k. The expression (9.2) simplifies in this case to

$$a_{k+1} \le (1 + \alpha_k^2 L_g^2)a_k - 2\alpha_k \mathbb{E}\left[\langle \nabla f(x^k), x^k - x^* \rangle\right]. \tag{5.29}$$

Supposing that f is strongly convex, with modulus of convexity $m > 0$, we have that

$$\langle \nabla f(x), x - x^\star \rangle \ge m\|x - x^\star\|^2. \tag{5.30}$$

By substituting into (5.29), we obtain

$$a_{k+1} \le (1 - 2m\alpha_k + L_g^2\alpha_k^2)a_k. \tag{5.31}$$

By choosing a constant steplength $\alpha_k \equiv \alpha$, for any α in the range $(0, 2m/L_g^2)$, we obtain a linear rate of convergence. The optimal choice of α is the one that minimizes the factor $(1-2m\alpha+L_g^2\alpha^2)$ in the right-hand side of (5.31), that is, $\alpha = 2/L_g^2$. For this choice, we obtain from (5.31) that $a_{k+1} \le (1 - m^2/L_g^2)a_k$, $k = 0, 1, 2, \ldots$, so that

$$a_k \le \left(1 - \frac{m^2}{L_g^2}\right)^k D_0^2. \tag{5.32}$$

We can use this expression to bound the number of iterations T required to guarantee that the expected error $\mathbb{E}\left[\|x^T - x^\star\|^2\right] = a_T$ falls below a specified threshold $\epsilon > 0$. By applying the technique in Section A.3 to (5.32) we find that

$$T = \left\lceil \frac{L_g^2}{m^2}\log\left(\frac{D_0^2}{\epsilon}\right)\right\rceil.$$

5.4.3 Case 3: B and L_g both nonzero

In the general case in which both B and L_g are nonzero, but f is strongly convex, we have by using (5.30) in (9.2) that

$$a_{k+1} \le (1 - 2m\alpha_k + \alpha_k^2 L_g^2)a_k + \alpha_k^2 B^2 \tag{5.33}$$

Constant Stepsize. First, consider the case of a constant stepsize. Assuming that $\alpha \in (0, 2m/L_g^2)$, we can roll out the recursion (5.33) to obtain

$$a_k \le (1 - 2m\alpha + \alpha^2 L_g^2)^k D_0^2 + \frac{\alpha B^2}{2m - \alpha L_g^2} \,. \tag{5.34}$$

No matter how many iterations k are taken, the bound on the right-hand size never falls below the threshold value

$$\frac{\alpha B^2}{2m - \alpha L_g^2} \,. \tag{5.35}$$

We see this behavior in practice. The iterates converge to a ball around the optimal solution, whose radius is bounded by (5.35), but from that point forward, they bounce around inside this ball.

We can reduce the radius of the ball by decreasing α. But this has the effect of slowing the linear rate of convergence indicated by the first term in the right-hand side of (5.34): the quantity $1 - 2m\alpha + \alpha^2 L_g^2$ moves closer to 1.

One way to balance these two effects is to use *epoch doubling.* The idea is to run with an aggressively large stepsize α for a certain number of iterations T (called an "epoch"). Then we halve the stepsize — replacing α by $\alpha/2$ — and continue to iterate, for another $2T$ iterations. Scheme such as this one can guarantee eventual convergence to x^*, at an overall rate that is reasonable.

Many variants that use more flexible, possibly adaptive rules for choosing the lengths of epochs (within which stepsizes are held constant) and the factor by which steplength is decreased between epochs are used in practice. (Typical factors are .8 or .9.) Indeed, tuning of these "hyperparameters" is one of the most important issues in practical implementation of the SGM.

Diminishing Stepsize. The scheme just described suggests another approach, one in which we decrease the stepsize α_k at a rate approximately proportional to $1/k$. (The epoch-doubling scheme is a piece-constant approximation to this. At the last iterate of epoch S, we will have taken about $(2^S - 1)T$ total iterations, and the current steplength will be $\alpha/2^{S-1}$.)

Suppose we choose the stepsize to satisfy

$$\alpha_k = \frac{\gamma}{k_0 + k},$$

where γ and k_0 are constants to be determined. We will show that suitable choices of these constants lead to an error bound of the form

$$a_k \le \frac{Q}{k_0 + k} \,,$$

for some Q. The following proposition can be proved by induction.

Proposition 5.2. *Suppose f is strongly convex with modulus of convexity m. If we run the SGM with stepsize*

$$\alpha_k = \frac{1}{2m(L_g^2/2m^2 + k)}, \quad k = 0, 1, 2, \ldots,$$

then we have for some numerical constant c_0,

$$\mathbb{E}[\|x^k - x^\star\|^2] \le \frac{c_0 B^2}{2m(L_g^2/2m^2 + k)}, \quad k = 0, 1, 2, \ldots \,.$$

5.5 Implementation Aspects

We mention here two techniques that are important elements of many practical implementations of the SGM.

5.5.1 Epochs

As mentioned in Section 5.4.3, a central concept in SGMs is the notion of *epochs*. In an epoch, some number of iterations are run, and then a choice is made about whether to change the stepsize. A common strategy is to run with a constant step size for some fixed number of iterations T, and then reduce the stepsize by a constant factor γ. Thus, if our initial stepsize is α, on the kth epoch, the stepsize is $\alpha\gamma^{k-1}$. This method is often more robust in practice than the diminishing stepsize rule. For this stepsize rule, a reasonable heuristic is to choose γ between 0.8 and 0.9.

Another popular rule is called *epoch doubling*. In this scheme, we run for T steps with stepsize α, then run $2T$ steps with stepsize $\alpha/2$, and then $4T$ steps with stepsize $\alpha/4$ and so on. Note that this provides a piecewise constant approximation to the function α/k.

5.5.2 Momentum

A popular variant of the SGM makes use of *momentum*, replacing the basic step (5.4) with one of the form

$$x^{k+1} = x^k - \alpha_k g(x^k, \xi^k) + \beta(x^k - x^{k-1}). \tag{5.36}$$

The inspiration for this approach comes, of course, from the accelerated gradient methods of Chapter 4. In practice, these variants are highly successful, with popular choices for β often falling in the range $[.8, .95]$.

In the case when $B = 0$, as in the randomized Kaczmarz method, the use of momentum can yield speedups comparable to those seen in the accelerated gradient methods of Chapter 4. The overhead of computing and maintaining the momentum term can cancel out the gains in speedup. (An exception is described in the Notes and References for this chapter.)

In the general case, the theoretical guarantees for momentum methods only demonstrate meager gains over the standard SGM. Essentially, we know that the function value will converge at a rate of $1/k$, but for certain instances, one can reduce the constant in front of the $1/k$ using momentum or acceleration. Regardless of the theoretical guarantees, one should always keep in mind that momentum can provide significant practical accelerations, and it should be considered an option in any implementation of the SGM.

Notes and References

Analysis of the case of $L_g = 0$, for both weakly and strongly convex cases, appears in [20].

A accelerated version of randomized Kaczmarz is described in [18]. A special technique is used to reduce the overhead associated with implementing the momentum step, so that some of the gains the convergence rate associated with the acceleration are preserved.

Exercises

1. Verify that the perceptron described in Section 5.1.3 is a special case of the SGM. In particular, verify the relationship between the steplength α_k and the parameters γ and η in the perceptron.

2. Consider the kth iteration (5.11) of the cyclic incremental gradient method applied to the function (5.10). Show that the minimizer is found exactly after n steps (that is $x^n = x^*$) and that $f(x^*)$ is one-half of the variance of the set $\{\omega_1, \omega_2, \dots, \omega_n\}$.

3. Verify the formula (5.13), given that the mean of the random variable ω is μ and its variance is σ^2. (The random variables ω_i, $i = 1, 2, \dots, k$ follow the same distribution, and all the random variables in this expression are independent.)

4. We showed that the unregularized support vector machine (5.21) admits a bound of the form (5.19) with $L_g = 0$. Find values of L_g and B such that the *regularized* support vector machine (5.7) satisfies (5.19). (Hint: Use the inequality $\|a+b\|^2 \le 2\|a\|^2 + 2\|b\|^2$.

5. (a) Consider the finite-sum objective (5.5) with additive Gaussian noise model on the component functions f_i, that is,

$$[\nabla f_i(x)]_j = [\nabla f(x)]_j + \epsilon_{ij}, \quad \text{for all } i = 1, 2, \dots, m \text{ and } j = 1, 2, \dots, d,$$

where $\epsilon_{ij} \sim N(0, \sigma^2)$ for all i, j. Show that when we estimate the gradient using a minibatch $\mathcal{S}$, that is,

$$g = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \nabla f_i(x),$$

then we have

$$\mathbb{E}(\|g - \nabla f(x)\|^2) = \frac{d}{|\mathcal{S}|}\sigma^2, \quad \mathbb{E}(\|g\|^2) = \|\nabla f(x)\|^2 + \frac{d}{|\mathcal{S}|}\sigma^2.$$

(b) Consider a minibatch strategy for the additive Gaussian noise model (5.3) for the general formulation (5.1). That is, the gradient estimate is

$$g(x; \xi_1, \xi_2, \dots, \xi_s) := \nabla f(x) + \frac{1}{s} \sum_{j=1}^{s} \xi_j,$$

where each ξ_j is i.i.d. with distribution $N(0, \sigma^2 I)$, and $s \ge 1$. Show that

$$\mathbb{E}_{\xi_1, \xi_2, \dots, \xi_s}\left(\|g(x; \xi_1, \xi_2, \dots, \xi_s)\|^2\right) = \|\nabla f(x)\|^2 + \frac{d}{s}\sigma^2.$$

6. **Dropout.** A popular heuristic in training neural networks is called *dropout*. Suppose we are running stochastic gradient descent on a function on $\mathbb{R}^d$. In each iteration of stochastic gradient descent, a set S of variables of size b is chosen at random. A stochastic gradient is

computed with those coordinates in S set to 0. Then, only the coordinates in S^c are updated. Suppose we are minimizing the least squares cost

$$f(x) = \frac{1}{2n} \sum_{i=1}^{n} (a_i^T x - b_i)^2 \,.$$

Find a function $\hat{f}(x)$ such that each iteration of dropout SGD corresponds to taking a valid step of the incremental gradient method applied to $\hat{f}$. Qualitatively, how does changing the size of S change the solution to which dropout SGD converges?

7. **Epoch SGD.** Let $f(x) = \mathbb{E}[F(x;\xi)]$ be a strongly convex function with parameter m. Assume that

$$\mathbb{E}[\|\nabla F(x;\xi)\|^2] \leq L_g^2 \|x - x_\star\|^2 + B^2 \,,$$

where $x_\star$ denotes the minimizer of f and L_g and B are constants. Suppose we run the stochastic gradient method on f by sampling ξ and taking steps along $\nabla F(x;\xi)$ using an *epoch doubling approach*. That is, we run for T steps with stepsize α, and then 2T steps with stepsize $\alpha/2$, and then $4T$ steps with stepsize $\alpha/4$ and so on. Let $\hat{x}_t$ be the average of all of the iterates in the tth epoch. How many epochs are required to guarantee that

$$\mathbb{E}[\|\hat{x}_t - x_\star\|^2] \leq \epsilon \,?$$

8. **Random line search.** Let $f : \mathbb{R}^d \to \mathbb{R}$ be a strongly convex function with L-Lipschitz gradients and strong convexity parameter m. Consider the following algorithm

   (a) Choose $x_0 \in \mathbb{R}^d$ and set $k = 0$.

   (b) Choose a direction v_k uniformly at random

   (c) Set $t_k = \arg\min_t f(x_k + t v_k)$

   (d) Set $x_{k+1} = x_k + t_k v_k$

   (e) Set $k = k + 1$ and repeat (b) through (d) until converged.

Prove that $\mathbb{E}[f(x_T) - f(x_\star)] \leq \epsilon$ provided

$$T \geq \frac{CdL}{m} \log\left(\frac{f(x_0) - f(x_\star)}{\epsilon}\right) \,,$$

where C is a small constant. What is the smallest value you can derive for C?

9. Consider applying stocchastic gradient with *constant stepsize* $\alpha \in (0,1)$ to (5.10), so that each iteration has the form

$$x^{k+1} = x^k - \alpha(x^k - \omega_{i_k})$$

for i_k drawn uniformly at random from $\{1, 2, \ldots, n\}$. Assuming that the initial point is $x^0 = 0$, write down an explicit expression for x^k, and find $\mathbb{E}_{i_0, i_1, \ldots, i_{k-1}}(x^k)$.