16 STOCHASTIC OPTIMIZATION

We examined several types of algorithms in the last chapters including gradient descent, coordinate descent, subgradient, proximal gradient, projection gradient, and mirror descent algorithms. We applied the methods to the solution of general convex optimization problems of the form:

$$w^\star = \operatorname*{argmin}_{w\in\mathbb{R}^M} \; P(w) \tag{16.1}$$

with and without constraints on w, and for both smooth and nonsmooth risks. In this chapter, we are going to exploit the structure of $P(w)$ and the fact that it may correspond to an empirical or stochastic risk, in which case problem (16.1) takes either form:

$$w^\star \triangleq \operatorname*{argmin}_{w\in\mathbb{R}^M} \left\{ P(w) \triangleq \frac{1}{N}\sum_{m=0}^{N-1} Q(w;\gamma(m),h_m) \right\} \tag{16.2a}$$

$$w^o \triangleq \operatorname*{argmin}_{w\in\mathbb{R}^M} \left\{ P(w) \triangleq \mathbb{E}\, Q(w;\boldsymbol{\gamma},\boldsymbol{h}) \right\} \tag{16.2b}$$

where $Q(w;\cdot,\cdot)$ is some convex loss function, $\gamma(m)\in\mathbb{R}$ are target signals, and $h_m\in\mathbb{R}^M$ are observed vectors (also called feature vectors). The expectation in the second line is over the joint distribution of the data $\{\boldsymbol{\gamma},\boldsymbol{h}\}$. The exposition will address two main challenges:

(a) (**Amount of data**). The size N of the dataset in empirical risk minimization problems can be large, which leads to a serious computational burden on the optimization methods devised so far in our treatment.

(b) (**Unknown statistics**). The joint pdf of $\{\boldsymbol{\gamma},\boldsymbol{h}\}$ for stochastic risks is generally unknown and, therefore, the risk $P(w)$ is unavailable. As a result, it is not possible to evaluate gradients or subgradients of $P(w)$ and many of the methods devised in the previous chapters will not be applicable.

These two issues are recurrent in the context of inference and learning problems. We will provide ways to alleviate their effect by appealing to *stochastic approximation*. Under this approach, and independent of whether we are dealing with empirical or stochastic risks, the gradient or subgradient vectors of $P(w)$ will be approximated from a small amount of data samples, using instantaneous or mini-batch calculations. And the (sub)gradient estimates will then be used to

drive the updates of the weight iterates. Some degradation in performance will ensue from the gradient approximations, but it is generally tolerable. The resulting *stochastic approximation algorithms* form the backbone of most data-driven inference and learning methods: they do not require knowledge of the underlying signal statistics, and do not need to process the entire dataset repeatedly at every iteration.

16.1 STOCHASTIC GRADIENT ALGORITHM

We start our treatment by considering the gradient-descent method where the risk function is smooth in order to illustrate the main steps of the stochastic approximation method. Recall that we use the qualification "smooth" to refer to risks that are at least first-order differentiable everywhere.

Applying the gradient-descent method to the solution of either problem (16.2a) or (16.2b) leads to update relations of the form:

$$\textbf{(empirical risk)} \quad w_n = w_{n-1} - \mu \times \left(\frac{1}{N} \sum_{m=0}^{N-1} \nabla_{w^\mathsf{T}} Q(w_{n-1}; \gamma(m), h_m) \right) \tag{16.3a}$$

$$\textbf{(stochastic risk)} \quad w_n = w_{n-1} - \mu \times \Big(\mathbb{E}\, \nabla_{w^\mathsf{T}} Q(w_{n-1}, \boldsymbol{\gamma}, \boldsymbol{h}) \Big) \tag{16.3b}$$

where in the second line we exchanged the order of the differentiation and expectation operators, i.e., we used

$$\nabla_{w^\mathsf{T}} P(w) \triangleq \nabla_{w^\mathsf{T}} \mathbb{E}\, Q(w; \boldsymbol{\gamma}, \boldsymbol{h}) \overset{(a)}{=} \mathbb{E}\, \nabla_{w^\mathsf{T}} Q(w; \boldsymbol{\gamma}, \boldsymbol{h}) \tag{16.4}$$

Example 16.1 (**Switching gradient and expectation operators**) We will often encounter situations that require switching gradient and expectation operators, as already seen in (16.4). We explain in Appendix 16.A on the *dominated convergence theorem* that the switching is possible under some mild conditions that are generally valid for our cases of interest. For instance, the switching will be possible when the loss function $Q(w;\cdot)$ and its gradient are continuous functions of w. The following example considers a situation where we can verify the validity of the switching operation directly from first principles. Consider the quadratic loss:

$$Q(w; \boldsymbol{\gamma}, \boldsymbol{h}) = (\boldsymbol{\gamma} - \boldsymbol{h}^\mathsf{T} w)^2, \ \boldsymbol{\gamma} \in \mathbb{R}, \ \ \boldsymbol{h} \in \mathbb{R}^M \tag{16.5a}$$

$$\nabla_{w^\mathsf{T}} Q(w; \boldsymbol{\gamma}, \boldsymbol{h}) = -2\boldsymbol{h}(\boldsymbol{\gamma} - \boldsymbol{h}^\mathsf{T} w) \tag{16.5b}$$

and assume $(\boldsymbol{\gamma}, \boldsymbol{h})$ have zero means with second-order moments $\mathbb{E}\,\boldsymbol{\gamma}^2 = \sigma_\gamma^2$, $\mathbb{E}\,\boldsymbol{h}\boldsymbol{\gamma} = r_{h\gamma}$, and $\mathbb{E}\,\boldsymbol{h}\boldsymbol{h}^\mathsf{T} = R_h$. In this case, direct calculations show that

$$\mathbb{E}\, Q(w; \boldsymbol{\gamma}, \boldsymbol{h}) = \sigma_\gamma^2 - 2(r_{h\gamma})^\mathsf{T} w + 2w^\mathsf{T} R_h w \tag{16.6a}$$

$$\nabla_{w^\mathsf{T}} \mathbb{E}\, Q(w; \boldsymbol{\gamma}, \boldsymbol{h}) = -2r_{h\gamma} + 2R_h w \tag{16.6b}$$

$$\mathbb{E}\, \nabla_{w^\mathsf{T}} Q(w; \boldsymbol{\gamma}, \boldsymbol{h}) \overset{(16.5b)}{=} -2r_{h\gamma} + 2R_h w \tag{16.6c}$$

from which we conclude, by comparing the last two lines, that the switching operation is justified.

Returning to the gradient-descent recursions (16.3a)–(16.3b), we observe first that implementation (16.3a) in the empirical case employs the *entire* dataset $\{\gamma(m), h_m\}$ at every iteration n. This means that the data needs to be available beforehand, prior to running the algorithm. This also means that this particular implementation cannot be used in streaming scenarios where data pairs $\{\gamma(n), h_n\}$ arrive sequentially, one pair at every iteration n. Moreover, the same dataset is used repeatedly by the algorithm, from one iteration to the other, until sufficient convergence is attained. The data is needed to compute the gradient of $P(w)$ at w_{n-1}, and this calculation can be costly for large N.

For implementation (16.3b) in the stochastic case, the situation is different but related. The main problem here is that the gradient of $P(w)$ is not known because the statistical distribution of the data $\{\boldsymbol{\gamma}, \boldsymbol{h}\}$ is unavailable and, hence, the risk $P(w)$ itself is not known. Even if the joint distribution of the data were known, evaluation of $P(w)$ and its gradient in closed form can still be difficult. Therefore, this second implementation suffers from the unavailability of statistical information. And even if this information is available, calculations can still be analytically intractable.

16.1.1 Stochastic Approximation

Stochastic approximation helps address these challenges for both cases of empirical and stochastic risks. Interestingly, the method will lead to the same implementation for both types of risks (empirical or stochastic). We motivate the approach by treating both risks separately initially for the benefit of the reader, until it becomes clear that one can proceed thereafter by using a unified presentation.

Empirical risks

First, in the empirical case (16.2a), the true gradient vector is given by

$$\nabla_{w^{\mathsf{T}}} P(w) \;=\; \frac{1}{N}\sum_{m=0}^{N-1} \nabla_{w^{\mathsf{T}}} Q(w; \gamma(m), h_m) \tag{16.7}$$

The gradient in this case is *known* but costly to compute and involves the entire dataset. Two popular ways to approximate it are as follows:

(a) We can select one data pair $(\boldsymbol{\gamma}(n), \boldsymbol{h}_n)$ *at random* from the $N-$size dataset and use it to approximate the true gradient by the expression:

$$\begin{aligned} &(\textbf{instantaneous approximation}) \\ &\widehat{\nabla_{w^{\mathsf{T}}} P}(w) \;=\; \nabla_{w^{\mathsf{T}}} Q(w; \boldsymbol{\gamma}(n), \boldsymbol{h}_n) \end{aligned} \tag{16.8}$$

In other words, we select a single term from the right-hand side of (16.7) to approximate $\nabla_{w^{\mathsf{T}}} P(w)$. There are many ways by which the data sample can

be selected from the dataset (e.g., by sampling with or without replacement). We will discuss sampling strategies in the sequel.

(b) More generally, we can select more than one data pair at random, say, a *mini-batch* of size B, and use them to approximate the true gradient by means of a sample average:

(**mini-batch approximation**)

$$\widehat{\nabla_{w^\mathsf{T}} P}(w) \;=\; \frac{1}{B}\sum_{b=0}^{B-1} \nabla_{w^\mathsf{T}} Q(w;\boldsymbol{\gamma}(b),\boldsymbol{h}_b) \tag{16.9}$$

Again, there are different ways by which the samples in the mini-batch can be selected from the given dataset. And, the value of B is usually a power of 2.

Stochastic risks

For the case of stochastic risks in (16.2b), the situation is different but related. The true gradient vector of the risk function now has the form:

$$\nabla_{w^\mathsf{T}} P(w) = \mathbb{E}\, \nabla_{w^\mathsf{T}} Q(w;\boldsymbol{\gamma},h) \tag{16.10}$$

The main difficulty is that this gradient is generally unknown because the expectation on the right-hand side cannot be computed either because the statistical distribution of the data is unavailable or because the computation is analytically intractable. Stochastic approximation provides one useful way out:

(a') We assume we can sample (or observe) one data pair $(\boldsymbol{\gamma}(n),\boldsymbol{h}_n)$ from the underlying joint distribution for $\{\boldsymbol{\gamma},\boldsymbol{h}\}$, and use this data sample to approximate the true gradient by the expression:

(**instantaneous approximation**)

$$\widehat{\nabla_{w^\mathsf{T}} P}(w) \;=\; \nabla_{w^\mathsf{T}} Q(w;\boldsymbol{\gamma}(n),\boldsymbol{h}_n) \tag{16.11}$$

Comparing with (16.10), we observe that we are effectively dropping the expectation operator $\mathbb{E}$ altogether and evaluating the gradient of the loss at one realization for the data. We say that we are replacing the expectation $\mathbb{E}\,\nabla_{w^\mathsf{T}} Q(w;\boldsymbol{\gamma},h)$ by the *instantaneous approximation* $\nabla_{w^\mathsf{T}} Q(w;\boldsymbol{\gamma}(n),\boldsymbol{h}_n)$. The resulting approximation (16.11) has the same form as in the empirical case, shown in (16.8).

(b') We can alternatively sample (or observe) more than one data pair from the underlying joint distribution for $\{\boldsymbol{\gamma},\boldsymbol{h}\}$, say, a *mini-batch* of size B, and use the samples to compute:

(**mini-batch approximation**)

$$\widehat{\nabla_{w^\mathsf{T}} P}(w) \;=\; \frac{1}{B}\sum_{b=0}^{B-1} \nabla_{w^\mathsf{T}} Q(w;\boldsymbol{\gamma}(b),\boldsymbol{h}_b) \tag{16.12}$$

Again, this expression has the same form as the approximation (16.9) used in the empirical case — compare with (16.9).

Comparison

The gradient approximations in the empirical and stochastic risk cases have the same form, whether a single data point is used or a mini-batch of data samples. The difference between the constructions lies in their interpretation. In the empirical case, the data samples are extracted from the already given dataset, whereas in the stochastic case the data samples stream in and correspond to realizations from the underlying distribution for $\{\boldsymbol{\gamma}, \boldsymbol{h}\}$. Either way, for empirical or stochastic risks, the resulting stochastic gradient algorithm (often denoted by the letters SGA) takes the following form for instantaneous approximations:

(**stochastic gradient algorithm, SGA**)
for every iteration $n \geq 0$**:**

$$\begin{aligned} &\text{select or receive a random data pair } (\boldsymbol{\gamma}(n), \boldsymbol{h}_n) \\ &\text{update } \ \boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu \nabla_{w^{\mathsf{T}}} Q(\boldsymbol{w}_{n-1}; \boldsymbol{\gamma}(n), \boldsymbol{h}_n) \end{aligned} \tag{16.13a}$$

end

or, in the mini-batch case,

(**mini-batch stochastic gradient algorithm, mini-SGA**)
for every iteration $n \geq 0$**:**

$$\begin{aligned} &\text{select or receive } B \text{ random data pairs } (\boldsymbol{\gamma}(b), \boldsymbol{h}_b) \\ &\text{update } \ \boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu \times \left(\frac{1}{B} \sum_{b=0}^{B-1} \nabla_{w^{\mathsf{T}}} Q(\boldsymbol{w}_{n-1}; \boldsymbol{\gamma}(b), \boldsymbol{h}_b) \right) \end{aligned} \tag{16.13b}$$

end

Obviously, the first implementation (16.13a) is a special case of the mini-batch algorithm when $B = 1$. Note that in these listings, we are denoting the samples $\{\boldsymbol{\gamma}(b), \boldsymbol{h}_b\}$ as well as the iterates $\{\boldsymbol{w}_{n-1}, \boldsymbol{w}_n\}$ in boldface. This is because they are now random variables. Indeed, assume we run either algorithm for N iterations and obtain the sequence of realizations $\{w_0, w_1, w_2, \ldots, w_{N-1}\}$. If we re-run the same algorithm again, and even if we start from the same initial condition w_{-1}, the sequence of realizations that will result for the iterates $\{\boldsymbol{w}_n\}$ will generally be different from the previous run. This is because the samples $\{\boldsymbol{\gamma}(b), \boldsymbol{h}_b\}$ are selected at random in both runs. However, as the analysis will reveal, all the random trajectories for $\{\boldsymbol{w}_n\}$ will continue to converge close enough to $w^\star$ in some meaningful sense.

We will refer to algorithms that employ sample-based approximations for the true gradients (or subgradients) as *stochastic optimization* methods. We will also refer to these methods as online learning algorithms or simply *online algorithms* because they respond to streaming data.

16.1.2 Convergence Questions

The randomness in (16.13a) or (16.13b) raises several interesting questions. For example, we established earlier in Theorem 12.1 that the gradient-descent implementation (16.3a) in the empirical case generates iterates w_n that converge at some exponential rate λ^n to the true minimizer $w^\star$ of $P(w)$. Now, however, the true gradient vector of $P(w)$ is replaced by an *approximation* based on a randomly selected data point (or on a mini-batch of randomly selected data points). Some loss in performance is expected to occur due to the gradient approximation. Specifically, the sequence of iterates $\boldsymbol{w}_n$ (and the corresponding risk values $P(\boldsymbol{w}_n)$) need not converge any longer to the exact minimizer $w^\star$ (and the corresponding minimum value $P(w^\star)$). Even the convergence question becomes more subtle because we are not dealing anymore with a *deterministic* sequence of iterates $\{w_n\}$ converging towards a limit point $w^\star$. Instead, we are now dealing with a *random* sequence $\{\boldsymbol{w}_n\}$. Each run of the stochastic algorithm generates a trajectory for this random process, and these trajectories will generally be different over different runs. A well-designed stochastic algorithm should be able to ensure that these trajectories will approach $w^\star$ in some useful probabilistic sense. Using the *random* error vector:

$$\widetilde{\boldsymbol{w}}_n \;\triangleq\; w^\star - \boldsymbol{w}_n \tag{16.14}$$

we will examine in future chapters convergence questions such as the following:

$$\begin{cases} \text{does } \mathbb{E}\,\|\widetilde{\boldsymbol{w}}_n\|^2 \text{ approach zero?} & \textbf{(mean-square-error convergence)} \\ \text{does } \lim\limits_{n\to\infty} \mathbb{P}(\|\widetilde{\boldsymbol{w}}_n\|^2 > \epsilon) = 0? & \textbf{(convergence in probability)} \\ \text{does } \mathbb{P}\Big(\lim\limits_{n\to\infty} \|\widetilde{\boldsymbol{w}}_n\|^2 = 0\Big) = 1? & \textbf{(almost sure convergence)} \end{cases} \tag{16.15}$$

We will often examine the limiting value of the mean-square-error (also called mean-square deviation or MSD):

$$\limsup_{n\to\infty}\; \mathbb{E}\,\|\widetilde{\boldsymbol{w}}_n\|^2 \tag{16.16}$$

from which we will be able to comment on the behavior of the algorithms in probability. This is because mean-square-error convergence implies convergence in probability in view of Markov inequality (recall the discussion in Appendix 3.A):

$$\mathbb{P}\Big(\|\widetilde{\boldsymbol{w}}_n\|^2 \geq \epsilon\Big) \;\leq\; \mathbb{E}\,\|\widetilde{\boldsymbol{w}}_n\|^2/\epsilon, \quad \text{for any } \epsilon > 0 \tag{16.17}$$

The convergence analysis of stochastic optimization methods is demanding due to the degradation that is introduced by the stochastic gradient (or subgradient) approximations. Nevertheless, the conclusions will be reassuring in that the methods will be shown to perform well for small enough step-sizes.

16.1.3 Sample Selection

In stochastic risk minimization, the samples $(\gamma(n), h_n)$ stream in and the stochastic gradient algorithm or its mini-batch version respond to them accordingly. However, in empirical risk minimization, when a collection of N data samples $\{\gamma(m), h_m\}$ is already available, it is necessary to devise strategies to sample from this dataset. There are several ways by which the random samples can be selected.

For generality, we let $\boldsymbol{\sigma}$ denote a random integer index from within the range $0 \leq \boldsymbol{\sigma} \leq N-1$. The value of $\boldsymbol{\sigma}$ determines the data pair that is used by the algorithm at iteration n, namely, $(\boldsymbol{\gamma}(\boldsymbol{\sigma}), \boldsymbol{h}_{\boldsymbol{\sigma}})$. Clearly, the value of $\boldsymbol{\sigma}$ varies with the iteration index n, which means that, in principle, we should be writing $\boldsymbol{\sigma}(n)$. We lighten the notation and write $\boldsymbol{\sigma}$. Using this variable, we rewrite (16.13a) more explicitly in the equivalent form:

(**stochastic gradient algorithm, SGA**)
for every iteration $n \geq 0$**:**
select a random index $\boldsymbol{\sigma}$ from within $0 \leq \boldsymbol{\sigma} \leq N-1$
consider the random data pair $(\boldsymbol{\gamma}(\boldsymbol{\sigma}), \boldsymbol{h}_{\boldsymbol{\sigma}})$
update $\boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu \nabla_{w^{\mathsf{T}}} Q(\boldsymbol{w}_{n-1}; \boldsymbol{\gamma}(\boldsymbol{\sigma}), \boldsymbol{h}_{\boldsymbol{\sigma}})$
end (16.18)

The random index $\boldsymbol{\sigma}$ can be selected in a number of ways:

(a) (**Uniform sampling**). In this case, $\boldsymbol{\sigma}$ is selected uniformly from the discrete set of indexes $\{0, 1, \ldots, N-1\}$ so that, for each integer m in this set:

$$\mathbb{P}(\boldsymbol{\sigma} = m) = \frac{1}{N}, \quad m \in \{0, 1, 2, \ldots, N-1\} \tag{16.19}$$

This mode of operation amounts to sampling from the N data pairs $\{\gamma(m), h_m\}$ *with replacement*, which means that some sample points may be selected multiple times during the operation of the algorithm.

(b) (**Random reshuffling**). In this case, we sample from the N data pairs *without replacement*. Another way to describe the sampling process is to randomly reshuffle the N data points first, and then process the samples *sequentially*, one after the other, from the reshuffled data.

(c) (**Importance sampling**). In this case, a probability value p_m is assigned to every index $m \in \{0, 1, \ldots, N-1\}$ with their sum adding up to one:

$$\sum_m p_m = 1 \tag{16.20}$$

The probabilities $\{p_m\}$ need not be uniform. At every iteration n, the random index $\boldsymbol{\sigma}$ is selected according to this distribution, i.e.,

$$\mathbb{P}(\boldsymbol{\sigma} = m) = p_m, \quad m \in \{0, 1, 2, \ldots, N-1\} \tag{16.21}$$

This mode of operation amounts to sampling from the $N-$data pairs *with*

replacement, albeit one where some data points are more or less likely to be selected according to the probabilities $\{p_m\}$. This is in contrast to (16.19) where all data points are equally likely to be selected. While this description assumes the $\{p_m\}$ are known, there are ways for stochastic algorithms to learn what values to use for the $\{p_m\}$ — see future Sec. 19.6.

Likewise, for the mini-batch implementation (16.13b), the B samples can be chosen with or without replacement:

(i) We can sample *with replacement* one data point $(\boldsymbol{\gamma}(b), \boldsymbol{h}_b)$ at a time until all B samples have been selected. In this mode of operation, the samples within each mini-batch are selected independently of each other although some samples may appear repeated.

(ii) We can sample *without replacement* one point $(\boldsymbol{\gamma}(b), \boldsymbol{h}_b)$ at a time until all B samples have been selected. In this case, the samples within each mini-batch will be different. However, the samples are not independent anymore because the selection of one sample is dependent on the previously selected samples.

Example 16.2 (**Bias under importance sampling**) It is useful to remark that we can always transform a stochastic implementation that relies on importance sampling into an equivalent implementation that applies *uniform* sampling to a larger dataset. The original dataset has N samples $\{\gamma(m), h_m\}$. We extend it to size $N' > N$ as follows. We repeat each sample $(\gamma(m), h_m)$ a number N_m of times such that its proportion, measured by N_m/N', in the new dataset becomes equal to p_m. In this way, selecting *uniformly* from the $N'-$long dataset will correspond to selecting with probability p_m from the original $N-$long dataset. Clearly, when this is done, the empirical risk that we will be minimizing will not be (16.3a) any longer, which is defined over the original $N-$data samples, but rather the modified risk:

$$P'(w) \triangleq \frac{1}{N'} \sum_{m'=0}^{N'-1} Q\Big(w; \gamma(m'), h_{m'}\Big) \tag{16.22}$$

where the sum is over the enlarged N' data samples (which include sample repetitions). This modified risk function can be rewritten in a weighted form in terms of the original data samples:

$$P'(w) \triangleq \sum_{m=0}^{N-1} p_m \, Q(w; \gamma(m), h_m) \tag{16.23}$$

where the loss values are scaled by the respective probabilities, $\{p_m\}$. This observation means that, under importance sampling, the stochastic gradient recursion, using either instantaneous or mini-batch gradient approximations, will be converging towards the minimizer of the weighted risk $P'(w)$ and not towards the desired minimizer $w^\star$ of the original risk $P(w)$. We say that importance sampling biases the solution. For this reason, in implementations that employ importance sampling, the instantaneous and mini-batch approximations for the gradient vector are usually redefined by incorporating an

additional scaling by $1/Np_\sigma$:

$$(\textbf{instantaneous}):\ \widehat{\nabla_{w^\mathsf{T}} P}(w) = \frac{1}{Np_\sigma}\nabla_{w^\mathsf{T}} Q(w;\gamma(\boldsymbol{\sigma}),h_{\boldsymbol{\sigma}}) \tag{16.24a}$$

$$(\textbf{mini-batch}):\ \widehat{\nabla_{w^\mathsf{T}} P}(w) = \frac{1}{B}\sum_{b=0}^{B-1}\frac{1}{Np_b}\nabla_{w^\mathsf{T}} Q(w;\boldsymbol{\gamma}(b),\boldsymbol{h}_b) \tag{16.24b}$$

The role of the scaling in removing the bias is explained in the next chapter.

16.1.4 Data Runs

Besides random sample selection, there is another element that arises in the implementation of stochastic approximation algorithms for *empirical risk minimization* when a collection of N data points $\{\gamma(m), h_m\}$ is available. It is customary to perform repeated *runs* (also called *epochs*) over the $N-$data points to enhance performance and smooth out the effect of errors due to the gradient approximations. During each run k, the algorithm starts from some initial condition, denoted by w_{-1}^k, and carries out N iterations by sampling from the data and performing stochastic gradient updates. At the end of the run, the final iterate is w_{N-1}^k. This iterate is chosen as the initial condition for the next run:

$$\underbrace{\boldsymbol{w}_{-1}^{k+1}}_{\substack{\text{start of}\\ \text{run } k+1}} = \underbrace{\boldsymbol{w}_{N-1}^{k}}_{\substack{\text{end of}\\ \text{run } k}} \tag{16.25}$$

and the process repeats. The algorithm starts from $\boldsymbol{w}_{-1}^{k+1}$, and carries out N iterations by sampling from the same data again and performing stochastic gradient updates. At the end of the run, the final iterate is $\boldsymbol{w}_{N-1}^{k+1}$, which is set to $\boldsymbol{w}_{-1}^{k+2}$ and so forth. The sampling of the data is random during each run so that the data are generally covered in different orders between runs. Within each run, the random samples are selected either with or without replacement.

We can describe the multiple runs explicitly by using the subscript $k \geq 1$ to index the iterates within the $k-$th run, such as writing $\boldsymbol{w}_n^k$. Using this notation, and incorporating multiple runs, the stochastic gradient algorithm (16.18) applied to the minimization of an empirical risk of the form (16.2a) can be as shown in (16.26); we can similarly write a mini-batch version:

(**stochastic gradient algorithm with multiple runs**)

given N data pairs $\{\gamma(m), h_m\}, m = 0, 1, \ldots, N-1$;
given a desired number of epochs, K;
start with an arbitrary initial condition $\boldsymbol{w}_{N-1}^0$.
for each epoch $k = 1, 2, \ldots, K$:
  set initial condition to $\boldsymbol{w}_{-1}^k = \boldsymbol{w}_{N-1}^{k-1}$;
  repeat for $n = 0, 1, 2, \ldots, N-1$:
    select a random index $0 \leq \boldsymbol{\sigma} \leq N-1$;
    $\boldsymbol{w}_n^k = \boldsymbol{w}_{n-1}^k - \mu \nabla_{w^{\mathsf{T}}} Q(\boldsymbol{w}_{n-1}^k; \boldsymbol{\gamma}(\boldsymbol{\sigma}), \boldsymbol{h}_{\boldsymbol{\sigma}})$
  end
end
return $w^\star \leftarrow \boldsymbol{w}_{N-1}^K$.

(16.26)

Since each run starts from the iterate obtained at the end of the previous run then, for all practical purposes, the above implementation can be described more succinctly as one standard iteration running continuously over the data as listed in (16.27) and (16.28) for instantaneous and mini-batch gradient approximations. The notation $(\boldsymbol{\gamma}(n), \boldsymbol{h}_n)$ in listing (16.27) refers to the data sample that is selected at iteration n. For example, in empirical risk minimization, this sample is selected at random from the $N-$size dataset $\{\gamma(m), h_m\}$ according to some selection policy (e.g., uniform sampling, random reshuffling) and then used to update $\boldsymbol{w}_{n-1}$ to $\boldsymbol{w}_n$. Similar remarks apply to the mini-batch version (16.28). From the discussion in the previous section, we already known that *the same algorithm is applicable to the minimization of stochastic risks* of the form (16.2b). The main difference is that the dataset $\{\gamma(m), h_m\}$ will not be available beforehand. Instead, the samples $(\boldsymbol{\gamma}(n), \boldsymbol{h}_n)$ will stream in successively over time. The description (16.27) accounts for this possibility as well and is applicable to the minimization of both empirical and stochastic risks.

Stochastic gradient algorithm for minimizing (16.2a) or (16.2b)

given dataset $\{\gamma(m), h_m\}_{m=0}^{N-1}$ or streaming data $(\gamma(n), h_n)$;
start from an arbitrary initial condition, $\boldsymbol{w}_{-1}$.
repeat until convergence over $n \geq 0$:
  select at random or receive a sample $(\boldsymbol{\gamma}(n), \boldsymbol{h}_n)$ at iteration n;
  $\boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu \nabla_{w^{\mathsf{T}}} Q(\boldsymbol{w}_{n-1}; \boldsymbol{\gamma}(n), \boldsymbol{h}_n)$
end
return $w^\star \leftarrow \boldsymbol{w}_n$.

(16.27)

Mini-batch stochastic gradient algorithm for minimizing (16.2a) or (16.2b)

given dataset $\{\gamma(m), h_m\}_{m=0}^{N-1}$ or streaming data $(\gamma(n), h_n)$;
given a mini-batch size, B;
start from an arbitrary initial condition, $\boldsymbol{w}_{-1}$.
repeat until convergence over $n \geq 0$:

select at random or receive B samples $\{\boldsymbol{\gamma}(b), \boldsymbol{h}_b\}_{b=0}^{B-1}$ at iteration n;

$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu \left(\frac{1}{B} \sum_{b=0}^{B-1} \nabla_{w^{\mathsf{T}}} Q(\boldsymbol{w}_{n-1}; \boldsymbol{\gamma}(b), \boldsymbol{h}_b) \right)$$

end
return $w^\star \leftarrow \boldsymbol{w}_n$.

(16.28)

Example 16.3 (**Delta rule and adaline**) Consider the ℓ_2-regularized least-squares risk:

$$P(w) = \rho \|w\|^2 \; + \; \frac{1}{N} \sum_{m=0}^{N-1} (\gamma(m) - h_m^{\mathsf{T}} w)^2 \tag{16.29}$$

For an arbitrary data point (γ, h) we have:

$$Q(w; \gamma, h) = \rho \|w\|^2 \; + \; (\gamma - h^{\mathsf{T}} w)^2 \tag{16.30a}$$

$$\nabla_{w^{\mathsf{T}}} Q(w; \gamma, h) = 2\rho w \; - \; 2h(\gamma - h^{\mathsf{T}} w) \tag{16.30b}$$

so that the stochastic gradient iteration (16.27) reduces to:

$$\boldsymbol{w}_n \; = \; (1 - 2\mu\rho)\, \boldsymbol{w}_{n-1} + 2\mu \boldsymbol{h}_n (\boldsymbol{\gamma}(n) - \boldsymbol{h}_n^{\mathsf{T}} \boldsymbol{w}_{n-1}), \quad n \geq 0 \tag{16.31}$$

where $\mu > 0$ is a small step-size. This recursion is known as the leaky LMS (least-mean-squares) algorithm in the adaptive filtering literature — the reason for the designation "least-mean-squares" is explained in the next example, where we re-derive the same algorithm as the solution to a stochastic risk minimizing problem involving the mean-square-error criterion. When $\rho = 0$, the recursion reduces to the plain LMS algorithm:

$$\boldsymbol{w}_n \; = \; \boldsymbol{w}_{n-1} + 2\mu \boldsymbol{h}_n (\boldsymbol{\gamma}(n) - \boldsymbol{h}_n^{\mathsf{T}} \boldsymbol{w}_{n-1}), \quad n \geq 0 \tag{16.32}$$

If we focus on a single entry of the weight vector, say, $\boldsymbol{w}_n[m']$ for the $m'-$th entry, and if we introduce the quantities

$$\delta \boldsymbol{w}_n[m'] \; \triangleq \; \boldsymbol{w}_n[m'] - \boldsymbol{w}_{n-1}[m'], \quad \text{(change due to update)} \tag{16.33}$$

$$\boldsymbol{e}(n) \; \triangleq \; \boldsymbol{\gamma}(n) - \boldsymbol{h}_n^{\mathsf{T}} \boldsymbol{w}_{n-1} \tag{16.34}$$

then recursion (16.32) gives, for the individual entries of the weight vector:

$$\delta \boldsymbol{w}_n[m'] = 2\mu \boldsymbol{e}(n) \boldsymbol{h}_n[m'] \tag{16.35}$$

In other words, the change in the individual entries of the weight vector is proportional to the observation entry, $\boldsymbol{h}_n[m']$, scaled by the error signal and step-size. This form of the LMS recursion is known as the *delta rule* in the machine learning and neural network literature. In the particular case when the $\boldsymbol{\gamma}(n)$ are binary variables assuming the values ± 1, recursion (16.32) is also referred to as the *adaline* algorithm, where "adaline" stands for "adaptive linear" solution.

Example 16.4 (**Delta rule from stochastic risk minimization**) We re-derive the same delta rule by considering instead the stochastic risk minimization problem:

$$P(w) = \rho\|w\|^2 \;+\; \mathbb{E}\,(\boldsymbol{\gamma} - \boldsymbol{h}^{\mathsf{T}}w)^2 \tag{16.36}$$

where the loss function is now given by

$$Q(w;\boldsymbol{\gamma},\boldsymbol{h}) = \rho\|w\|^2 \;+(\boldsymbol{\gamma} - \boldsymbol{h}^{\mathsf{T}}w)^2 \tag{16.37a}$$

$$\nabla_{w^{\mathsf{T}}}\, Q(w;\boldsymbol{\gamma},\boldsymbol{h}) = 2\rho w \;-\; 2\boldsymbol{h}(\boldsymbol{\gamma} - \boldsymbol{h}^{\mathsf{T}}w) \tag{16.37b}$$

In this case, the stochastic gradient iteration (16.27) leads to the same delta rule:

$$\boldsymbol{w}_n \;=\; (1 - 2\mu\rho)\,\boldsymbol{w}_{n-1} + 2\mu\boldsymbol{h}_n(\boldsymbol{\gamma}(n) - \boldsymbol{h}_n^{\mathsf{T}}\boldsymbol{w}_{n-1}) \tag{16.38}$$

We illustrate the performance of the algorithm in Fig. 16.1, which shows the learning curve in linear scale using $\rho = 0.5$, $\mu = 0.001$, and $M = 10$. The simulation generates random pairs of data $\{\gamma(m), h_m\}$ according to a linear model. First, a random parameter model $w^a \in \mathbb{R}^{10}$ is selected, and a random collection of feature vectors $\{h_m\}$ are generated with zero-mean unit-variance Gaussian entries. Likewise, a collection of N independent noise Gaussian entries $\{v(m)\}$ with zero mean and variance $\sigma_v^2 = 0.0001$ is generated. Then, each $\gamma(m)$ is set to

$$\gamma(m) = h_m^{\mathsf{T}}w^a + v(m) \tag{16.39}$$

The minimizer w^o for the risk (16.36) can be determined in closed form and is given by

$$w^o = (\rho I_M + R_h)^{-1} r_{h\gamma} \tag{16.40}$$

where, for the simulated data,

$$R_h \;=\; \mathbb{E}\,\boldsymbol{h}\boldsymbol{h}^{\mathsf{T}} \;=\; I_M, \quad \text{(by construction)} \tag{16.41a}$$

$$r_{h\gamma} \;\triangleq\; \mathbb{E}\,\boldsymbol{\gamma}\boldsymbol{h} = \mathbb{E}\,(\boldsymbol{h}^{\mathsf{T}}w^a + \boldsymbol{v})\boldsymbol{h} = R_h w^a = w^a \tag{16.41b}$$

In the stochastic-gradient implementation (16.38), data $(\boldsymbol{\gamma}(n), \boldsymbol{h}_n)$ stream in, one pair at a time. The *learning curve* of the stochastic algorithm is denoted by $P(n)$ and defined as

$$P(n) \;\triangleq\; \mathbb{E}\,P(\boldsymbol{w}_{n-1}), \qquad (\textbf{learning curve}) \tag{16.42}$$

where the expectation is over the randomness in the weight iterates. The learning curve shows how the risk value evolves on average over time. We can simulate the learning curve by using repeated experiments with each experiment having its own data and starting from the same initial condition. We construct the learning curve in this example as follows. We generate L replicas of $N = 3000-$long data sequences $\{\gamma(m), h_m\}$ arising from the same linear model to ensure they have the *same* statistical distribution. We subsequently run the stochastic-gradient algorithm on each of these datasets, always starting from the *same* initial condition w_{-1}. In the simulations we perform $L = 500$ experiments. Each experiment ℓ results in a realization for a risk curve of the form:

$$\widehat{P}(w_{n-1}^{\ell}) = \rho\|w_{n-1}^{\ell}\|^2 \;+\; \left(\gamma(n) - h_n^{\mathsf{T}}w_{n-1}^{\ell}\right)^2, \;\; 0 \le n \le N-1 \tag{16.43}$$

This curve is evaluated at the successive weight iterates during the $\ell-$th experiment. By averaging the curves over all $L-$experiments we obtain an ensemble average approximation for the true risk value as follows:

$$P(n) \approx \frac{1}{L}\sum_{\ell=1}^{L}\widehat{P}(w_{n-1}^{\ell}), \;\; 0 \le n \le N-1 \tag{16.44}$$

where $P(n)$ denotes the estimate for the risk value $P(w)$ at the $n-$th iteration of the algorithm. This is the curve that is shown in the left plot of Fig. 16.4.

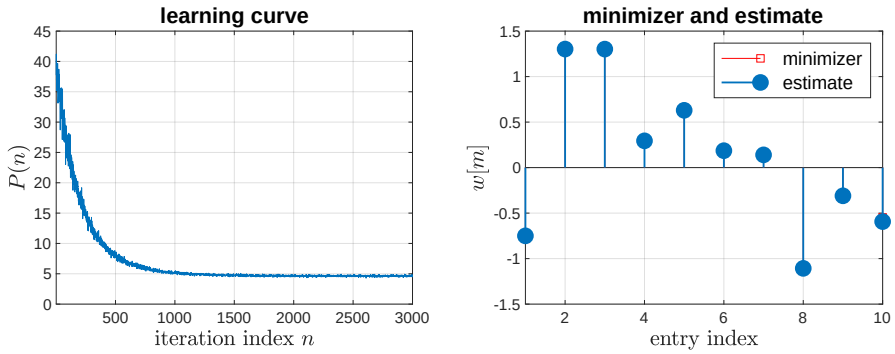

Figure 16.1 (*Left*) Ensemble-average learning curve $P(n)$ for the stochastic gradient implementation (16.27) in linear scale obtained by averaging over repeated experiments. (*Right*) Comparison of the minimizer w^o and the limit iterate w_n obtained at the end of one experiment.

Example 16.5 (**Logistic regression**) Consider the ℓ_2-regularized logistic regression empirical risk:

$$P(w) = \rho\|w\|^2 + \frac{1}{N}\sum_{m=0}^{N-1} \ln\left(1 + e^{-\gamma(m)h_m^{\mathsf{T}}w}\right) \tag{16.45}$$

for which

$$Q(w;\gamma(n),h_n) = \rho\|w\|^2 \;+\; \ln\left(1 + e^{-\gamma(n)h_n^{\mathsf{T}}w}\right) \tag{16.46a}$$

$$\nabla_{w^{\mathsf{T}}} Q(w;\gamma(n),h_n) = 2\rho w \;-\; \frac{\gamma(n)h_n}{1 + e^{\gamma(n)h_n^{\mathsf{T}}w}} \tag{16.46b}$$

The stochastic gradient iteration (16.27) becomes

$$\boldsymbol{w}_n \;=\; (1 - 2\mu\rho)\,\boldsymbol{w}_{n-1} + \mu\frac{\boldsymbol{\gamma}(n)\boldsymbol{h}_n}{1 + e^{\boldsymbol{\gamma}(n)\boldsymbol{h}_n^{\mathsf{T}}\boldsymbol{w}_{n-1}}}, \quad n \geq 0 \tag{16.47}$$

We illustrate the performance of this algorithm in Fig. 16.2, which shows the normalized learning curves in logarithmic scale under uniform sampling and random reshuffling, in addition to the learning curve for a mini-batch implementation.

The simulation uses $\rho = 1$, $\mu = 0.0001$, and $M = 10$. It generates $N = 500$ random pairs of data $\{\gamma(m), h_m\}$ according to a logistic model. First, a random parameter model $w^a \in \mathbb{R}^{10}$ is selected, and a random collection of feature vectors $\{h_m\}$ are generated, say, with zero-mean unit-variance Gaussian entries. Then, for each h_m, the label $\gamma(m)$ is set to either $+1$ or -1 according to the following construction:

$$\gamma(m) = +1 \;\text{ if }\; \left(\frac{1}{1 + e^{-h_m^{\mathsf{T}}w^a}}\right) \geq 0.5; \;\text{ otherwise } \gamma(m) = -1 \tag{16.48}$$

A total of $K = 500$ epochs are run over the data. The learning curves are plotted in normalized logarithmic scale in line with construction (11.65), namely,

$$\ln\left(\frac{P(w_n) - P(w^\star)}{\max_n\{P(w_n) - P(w^\star)\}}\right) \tag{16.49}$$

where $w^\star$ is approximated by the limit value of the weight iterate after sufficient convergence. The mini-batch implementation employs mini-batches of size $B = 5$ samples.

Each learning curve is generated by plotting the value of the risk function at the start of each epoch, namely, $P(w_{-1}^k)$.

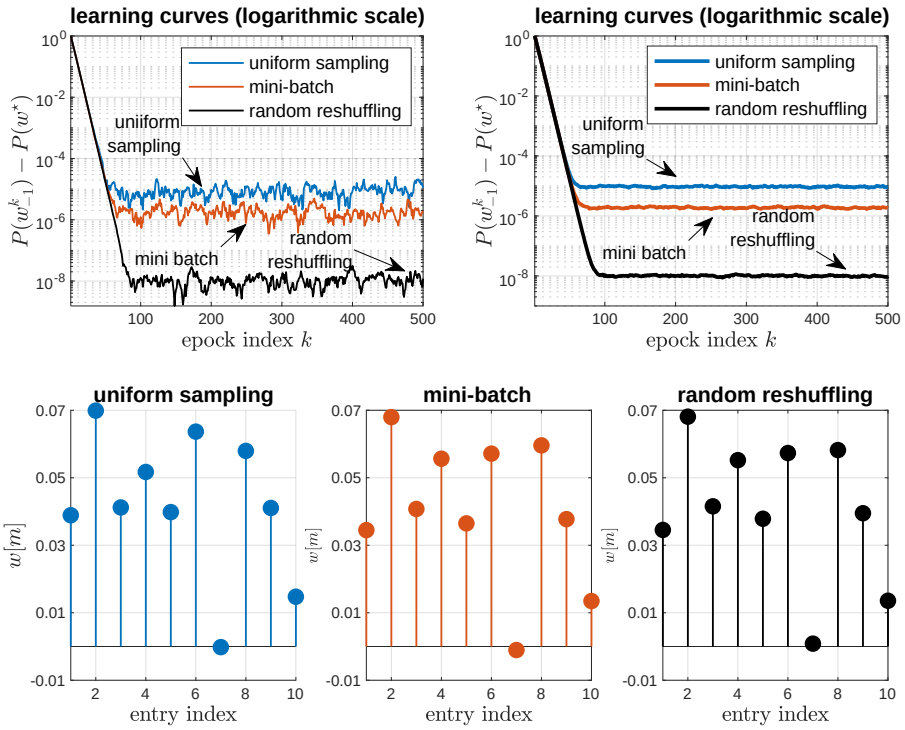

Figure 16.2 (*Top left*) Learning curves $P(w_{-1}^k)$ relative to the minimum risk value $P(w^\star)$ in normalized logarithmic scale for the stochastic-gradient implementation (16.47) under uniform sampling and random reshuffling, in addition to the learning curve for the mini-batch implementation. (*Top right*) The learning curves are smoothed over $L = 100$ experiments. (*Bottom*) Limiting values for the weight iterates under three data sampling policies.

The plot on the left in the top row of the figure shows the evolution of these values relative to the minimum risk value $P(w^\star)$ for one experiment using $K = 500$ epochs. The noisy variations in these learning curves are a reflection of the stochastic nature of the updates. We repeat this experiment for a total of $L = 100$ times and average the learning curves; with each experiment starting from the same initial condition $\boldsymbol{w}_{-1}$. The result leads to the smoother curves shown in the plot on the right in the top row of the figure. The curves illustrate the improved performance that is delivered by the mini-batch and random reshuffling versions of the stochastic-gradient algorithm; these observations will be established analytically in our future derivations — see Table 16.2. The plots in the bottom row show the limiting value of the weight iterates under the three data sampling policies at the end of one experiment involving $K = 500$ runs.

Example 16.6 (**Polyak-Ruppert averaging**) Sometimes a running average step is coupled with the stochastic gradient implementation (16.27) in order to smooth the iter-

ates. Let

$$\bar{\boldsymbol{w}}_{n-2} \triangleq \frac{1}{n}\sum_{m=0}^{n-1} \boldsymbol{w}_{m-1}, \quad n \geq 0 \tag{16.50}$$

which averages all iterates up to $\boldsymbol{w}_{n-1}$. The smoothed variable $\bar{\boldsymbol{w}}_n$ can be computed recursively as follows:

$$\bar{\boldsymbol{w}}_n = \bar{\boldsymbol{w}}_{n-1} + \frac{1}{n+2}(\boldsymbol{w}_n - \bar{\boldsymbol{w}}_{n-1}), \quad \bar{\boldsymbol{w}}_{-1} = \boldsymbol{w}_{-1} \tag{16.51}$$

andthe stochastic gradient recursion (16.27) is adjusted to

(**stochastic-gradient algorithm with Polyak-Ruppert averaging**)

$$\begin{cases} \boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu\,\nabla_{w^{\mathsf{T}}} Q(\boldsymbol{w}_{n-1}; \boldsymbol{\gamma}(n), \boldsymbol{h}_n), \quad n \geq 0 \\ \bar{\boldsymbol{w}}_n = \bar{\boldsymbol{w}}_{n-1} + \dfrac{1}{n+2}(\boldsymbol{w}_n - \bar{\boldsymbol{w}}_{n-1}) \end{cases} \tag{16.52}$$

where $\bar{\boldsymbol{w}}_n$ is taken as the output variable.

Example 16.7 **(Recommender systems and matrix factorization)** We provide an example with a non-convex risk function, which is widely used in the design of *recommender systems*. These are automated systems used to suggest recommendations to users for products based on their past preferences and the preferences of other similar users. Such systems are widely used by online business sites. We motivate the approach by considering the example of a streaming movie service.

Assume there are U users, labeled $u = 1, 2, \ldots, U$, and I items (such as movies), labeled $i = 1, 2, \ldots, I$. Based on past interactions between the users and the service provider, the users have provided ratings for different movies (say, on a numerical scale from 1=poor to 5=excellent). Table 16.1 shows one example of the type of information that is available. Each row corresponds to a user u, and each column corresponds to a movie item, i. The table is showing data for 9 movies and 7 users. Usually, users rate only some of the movies; they may not have watched all movies and they may not provide feedback on all movies they watch. For this reason, some entries in the table are marked with question marks to indicate that these ratings are missing.

Table 16.1 Ratings provided by users for some movie items from the service provider.

user	M1	M2	M3	M4	M5	M6	M7	M8	M9
U1	3	?	4	?	1	5	?	?	1
U2	3	4	?	?	2	?	4	4	?
U3	?	5	2	1	1	2	4	5	3
U4	?	?	?	4	?	?	?	2	5
U5	3	2	4	?	3	3	3	3	?
U6	?	3	3	2	2	?	1	?	1
U7	?	1	?	?	?	3	?	2	4

We collect all ratings into a $U \times I$ user-item (or ratings) matrix $R = [r_{ui}]$; it contains all the scores from the table with r_{ui} representing the score given by user u for item i. Some entries in the matrix R will be missing and we mark them by a question mark. We denote the set of available entries in R by $\mathcal{R}$. Thus, when we write $(u, i) \in \mathcal{R}$ we mean that entry r_{ui} has a valid ratings score. The objective of a recommender system is to predict what ratings users are likely to provide in place of the question marks. For example, referring to the table, we would like to know what ratings user U1 is likely to provide to movies M2, M4, M7, and M8. Based on these predictions, the service

provider will then recommend some movies to the user. There are several methods that can be used to predict the numerical values for the missing entries (i.e., to perform what is known as *imputation* or matrix completion). Here, we follow one approach known as *collaborative filtering*. It is based on exploiting relationships between users and using a convenient matrix factorization.

We assume that each item i can be represented by a feature vector $h_i \in \mathbb{R}^M$; the entries of this vector are called *latent* variables because they are hidden and will need to be discovered or learned. For example, for the case of movies, the entries of h_i could be some explicit attributes that relate to the type of movie (comedy, action, thriller, ...), the duration of the movie, if the movie has won any awards, or other more implicit attributes. Since the latent variables will be discovered by the matrix factorization approach, they may not relate directly to explicit attributes.

We further assume that each user u employs a weight vector $w_u \in \mathbb{R}^M$ to arrive at its ratings. The entries of this vector scale different attributes for the item (or movie) differently. Some users prefer comedy movies over suspense movies, or shorter movies over longer movies. If we happen to know the feature representation h_i for some item i, then we model the rating process used by user u as computing the inner product

$$r_{ui} \approx h_i^{\mathsf{T}} w_u - \theta \tag{16.53}$$

where θ models some bias term. For example, some users may provide consistently higher-than-average ratings, or some items (movies) may be perceived consistently as being superior to other movies. These perceptions can bias the rating process. To be more specific, we should split the bias into two sources: one arises from user-related biases (users behave differently) and the second arises from item-related biases (some items elicit different types of reactions from users; perhaps because they are promoted more strongly than other movies). For this reason, it is common to replace the above ratings generation model by one of the form

$$r_{ui} \approx h_i^{\mathsf{T}} w_u - \theta_u - \alpha_i, \quad u = 1, 2, \ldots, U, \quad i = 1, 2, \ldots I \tag{16.54}$$

with two scalar bias terms $\{\theta_u, \alpha_i\}$; one by the user and the other by the item. If we collect the feature vectors into an $M \times I$ matrix H and all user models into a $U \times M$ matrix W, a $U \times 1$ vector θ_U, and an $I \times 1$ vector α:

$$H = \begin{bmatrix} h_1 & h_2 & \ldots & h_I \end{bmatrix}, \; W = \begin{bmatrix} w_1^{\mathsf{T}} \\ w_2^{\mathsf{T}} \\ \vdots \\ w_U^{\mathsf{T}} \end{bmatrix}, \; \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_U \end{bmatrix}, \; \alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_I \end{bmatrix} \tag{16.55}$$

then expression (16.54) amounts to assuming that the ratings matrix R is generated according to the model:

$$R \approx WH - \theta \mathbb{1}_I^{\mathsf{T}} - \mathbb{1}_U \alpha^{\mathsf{T}} = \begin{bmatrix} W & -\theta & \mathbb{1}_U \end{bmatrix} \begin{bmatrix} H \\ \mathbb{1}_I^{\mathsf{T}} \\ -\alpha^{\mathsf{T}} \end{bmatrix} \tag{16.56}$$

This expression factors R into the product of two matrices: the first has $M+2$ columns and the second has $M+2$ rows. If we succeed in determining the quantities $\{W, H, \theta, \alpha\}$, then we can use relation (16.56) to predict the ratings at all locations in R. For this

purpose, we will minimize the following regularized least-squares risk function:

$$\left\{\widehat{w}_u, \widehat{h}_i, \widehat{\theta}_u, \widehat{\alpha}_i\right\} = \underset{\{w_u, h_i, \theta_u, \alpha_i\}}{\operatorname{argmin}} \left\{ \sum_{u=1}^{U} \rho \|w_u\|^2 + \sum_{i=1}^{I} \rho \|h_i\|^2 + \sum_{(u,i)\in\mathcal{R}} \left(r_{ui} - h_i^{\mathsf{T}} w_u + \theta_u + \alpha_i\right)^2 \right\} \tag{16.57}$$

where the last sum is over the valid indexes $(u, i) \in \mathcal{R}$. The above risk function is nonconvex because of the products $h_i^{\mathsf{T}} w_u$. We can approximate the solution by means of a stochastic gradient implementation, which takes the form shown in listing (16.58). The entries of the initial iterates $\boldsymbol{w}_{u,-1}$ and $\boldsymbol{h}_{i,-1}$ are selected at random from a uniform distribution in the range $[0, 1/\sqrt{M}]$.

Stochastic gradient algorithm applied to recommender problem (16.57)

given valid ratings in locations $(u, i) \in \mathcal{R}$;
start from arbitrary $\{\boldsymbol{w}_{u,-1}, \boldsymbol{h}_{i,-1}, \boldsymbol{\theta}_u(-1), \boldsymbol{\alpha}_i(-1)\}$.

repeat until convergence over $m \geq 0$
  select a random entry $(\boldsymbol{u}, \boldsymbol{i}) \in \mathcal{R}$

$$\begin{aligned}
&\boldsymbol{e}(m) = \boldsymbol{r_{ui}} - \boldsymbol{h}_{i,m-1}^{\mathsf{T}} \boldsymbol{w}_{u,m-1} + \boldsymbol{\theta_u}(m-1) + \boldsymbol{\alpha_i}(m-1) \\
&\boldsymbol{w}_{\boldsymbol{u},m} = (1 - 2\mu\rho)\boldsymbol{w}_{\boldsymbol{u},m-1} + 2\mu \boldsymbol{h}_{\boldsymbol{i},m-1} \boldsymbol{e}(m) \\
&\boldsymbol{h}_{\boldsymbol{i},m} = (1 - 2\mu\rho)\boldsymbol{h}_{\boldsymbol{i},m-1} + 2\mu \boldsymbol{w}_{\boldsymbol{u},m-1} \boldsymbol{e}(m) \\
&\boldsymbol{\theta_u}(m) = \boldsymbol{\theta_u}(m-1) - 2\mu \boldsymbol{e}(m) \\
&\boldsymbol{\alpha_i}(m) = \boldsymbol{\alpha_i}(m-1) - 2\mu \boldsymbol{e}(m)
\end{aligned} \tag{16.58}$$

end
return $\{w_u^\star, h_i^\star, \theta_u^\star, \alpha_i^\star\}$.

In the above listing, the term $w_{u,m}$ represents the estimate for w_u at iteration m; likewise for $h_{i,m}$, $\theta_u(m)$, and $\alpha_i(m)$. It is useful to note that although the recursions for updating θ_u and α_i look similar, these variables will generally be updated at different instants. This is because the same i will appear under different u values, and the same u will appear with different i values. Later in Example 50.6 we revisit this problem and solve it by applying an alternating least-squares solution. We also revisit the same problem in Example 68.2 and solve it by employing variational autoencoders.

We simulate recursions (16.58) by generating a random ratings matrix R with $U = 10$ users and $I = 10$ items. The scores are integer numbers in the range $1 \leq r \leq 5$, and unavailable scores are indicated by the symbol ?:

$$R = \begin{bmatrix}
5 & 3 & 2 & 2 & ? & 3 & 4 & ? & 3 & 3 \\
5 & 4 & 1 & 3 & 1 & 4 & 4 & ? & 3 & ? \\
3 & 5 & ? & 2 & 1 & 5 & 4 & 1 & 4 & 1 \\
? & 2 & 3 & 4 & 4 & 5 & 2 & 5 & 1 & 1 \\
2 & 1 & 2 & 2 & 1 & 5 & 1 & 4 & 1 & ? \\
? & 2 & 1 & 3 & ? & ? & 5 & 3 & 3 & 5 \\
3 & 4 & ? & 2 & 5 & 5 & 3 & 2 & ? & 4 \\
4 & 5 & 3 & 4 & 2 & 2 & 1 & ? & 5 & 5 \\
2 & 4 & 2 & 5 & ? & 1 & 1 & 3 & 1 & 4 \\
? & 1 & 4 & 4 & 3 & ? & 5 & 2 & 4 & 3
\end{bmatrix} \tag{16.59}$$

We set $M = 5$ (feature vectors of size 5) and generate uniform random initial conditions for the variables $\{\boldsymbol{w}_{u,-1}, \boldsymbol{h}_{i,-1}, \boldsymbol{\theta}_u(-1), \boldsymbol{\alpha}_i(-1)\}$ in the open interval $(0, 1)$. We set $\mu = 0.0001$ and $\rho = 0$ (no regularization). We normalize the entries of R to lie in the

range $[0,1]$ by replacing each numerical entry r by the value $r \leftarrow (r-1)/4$ where the denominator is the score range (highest value minus smallest value) and the smallest score is subtracted from the numerator. We run a large number of iterations until sufficient convergence is attained. Specifically, we run $K = 50,000$ epochs with the data randomly reshuffled at the beginning of each run. At the end of the simulation, we use the parameters $\{w_u^\star, h_i^\star, \theta_u^\star, \alpha_i^\star\}$ to estimate each entry of R using

$$\widehat{r}_{ui} = (h_i^\star)^{\mathsf{T}} w_u^\star - \theta_u^\star - \alpha_i^\star \tag{16.60}$$

We undo the normalization by replacing each of these predicted values by $\widehat{r}_{ui} \leftarrow 4\widehat{r}_{ui}+1$ and rounding each value to the closest integer; scores above 5 are saturated at 5 and scores below 1 are fixed at 1. The result is the matrix $\widehat{R}$ shown below, where we indicate the scores predicted for the unknown entries in red; we also indicate in blue those locations where the estimated scores differ by one level from the original scores:

$$\widehat{R} = \begin{bmatrix} 5 & 3 & 2 & 2 & \mathbf{2} & 3 & 4 & \mathbf{2} & 3 & 3 \\ 5 & 4 & 1 & 3 & 1 & 4 & 4 & \mathbf{3} & 3 & \mathbf{3} \\ 3 & \mathbf{4} & \mathbf{4} & 2 & 1 & 5 & \mathbf{3} & 1 & 4 & 1 \\ \mathbf{2} & 2 & 3 & 4 & 4 & 5 & 2 & 5 & 1 & 1 \\ 2 & \mathbf{2} & 2 & 2 & 1 & 5 & \mathbf{2} & 4 & 1 & \mathbf{1} \\ \mathbf{4} & 2 & 1 & 3 & \mathbf{5} & \mathbf{3} & 5 & 3 & 3 & 5 \\ 3 & 4 & \mathbf{2} & 2 & 5 & 5 & 3 & 2 & \mathbf{4} & 4 \\ \mathbf{3} & 5 & 3 & 4 & 2 & 2 & \mathbf{2} & \mathbf{1} & 5 & 5 \\ 2 & 4 & 2 & 5 & \mathbf{2} & 1 & 1 & 3 & 1 & 4 \\ \mathbf{4} & 1 & 4 & 4 & 3 & \mathbf{1} & 5 & 2 & 4 & 3 \end{bmatrix} \tag{16.61}$$

Figure 16.3 provides a color-coded representation of the entries of the original matrix R on the left with the locations of the missing entries highlighted by red squares, and the recovered matrix $\widehat{R}$ on the right.

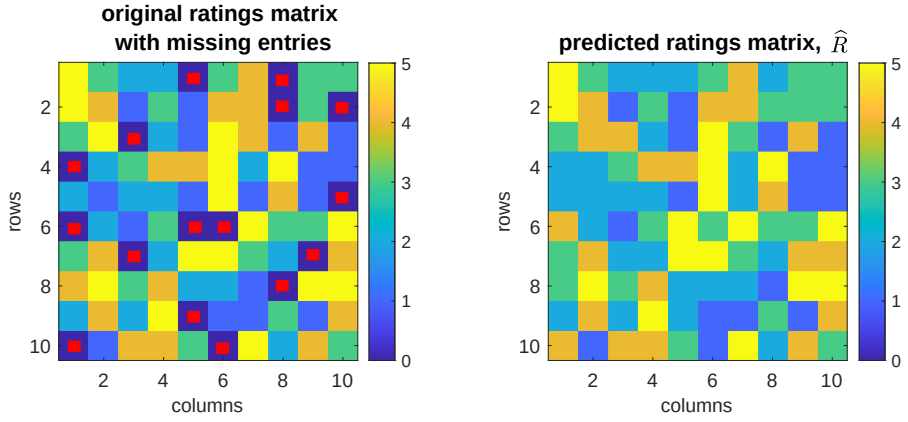

Figure 16.3 Color coded representation of the entries of the original matrix R with missing entries (*left*) and the recovered matrix $\widehat{R}$ (*right*).

We further denote the risk value at the start of each epoch of index k by

$$P(k) \triangleq \sum_{u=1}^{U} \rho\|w_u\|^2 + \sum_{i=1}^{I} \rho\|h_i\|^2 + \sum_{(u,i)\in\mathcal{R}} \Big(r_{ui} - h_i^{\mathsf{T}} w_u + \theta_u + \alpha_i\Big)^2 \tag{16.62}$$

where the parameters on the right-hand side are set to the values at the start of epoch

k. Figure 16.4 plots the evolution of the risk curve (normalized by its maximum value so that its peak value is set to one).

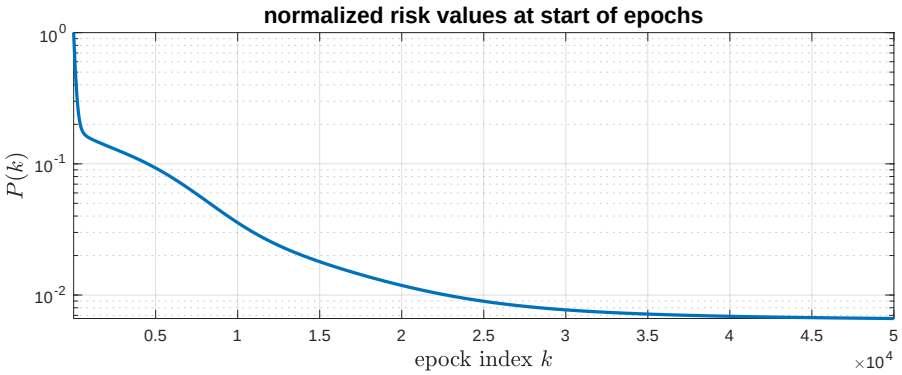

Figure 16.4 Evolution of the risk curve (16.62) with its peak value normalized to one.

16.2 STOCHASTIC SUBGRADIENT ALGORITHM

The same arguments used for the derivation of the stochastic gradient algorithm and its mini-batch version can be applied to nonsmooth risks (i.e., risks with points of non-differentiability) to arrive at the *stochastic subgradient algorithm*. The main difference is that gradients will now be replaced by subgradients:

$$\nabla_{w^{\mathsf{T}}} Q(w;\gamma,h) \;\text{ replaced by }\; s_Q(w;\gamma,h) \tag{16.63}$$

where $s_Q(w;\gamma,h)$ refers to a subgradient construction for the loss function $Q(w,\cdot)$ evaluated at the data point (γ,h). The substitution leads to listings (16.27) and (16.65) for the minimization of empirical or stochastic risks using instantaneous or mini-batch subgradient approximations.

Stochastic subgradient algorithm for minimizing (16.2a) or (16.2b)

given dataset $\{\gamma(m),h_m\}_{m=0}^{N-1}$ or streaming data $(\gamma(n),h_n)$;
start from an arbitrary initial condition, $\boldsymbol{w}_{-1}$.
repeat until convergence over $n \geq 0$:
  select at random or receive a sample $(\boldsymbol{\gamma}(n),\boldsymbol{h}_n)$ at iteration n;
  $\boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu\, s_Q(\boldsymbol{w}_{n-1};\boldsymbol{\gamma}(n),\boldsymbol{h}_n)$
end
return $w^\star \leftarrow \boldsymbol{w}_n$.

(16.64)

Mini-batch stochastic subgradient alg. for minimizing (16.2a) or (16.2b)

given dataset $\{\gamma(m), h_m\}_{m=0}^{N-1}$ or streaming data $(\gamma(n), h_n)$;
given a mini-batch size, B;
start from an arbitrary initial condition, $\boldsymbol{w}_{-1}$.
repeat until convergence over $n \geq 0$:
  select at random or receive B samples $\{\boldsymbol{\gamma}(b), \boldsymbol{h}_b\}_{b=0}^{B-1}$ at iteration n;

$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu \left(\frac{1}{B} \sum_{b=0}^{B-1} s_Q(\boldsymbol{w}_{n-1}; \boldsymbol{\gamma}(b), \boldsymbol{h}_b) \right)$$

end
return $w^\star \leftarrow \boldsymbol{w}_n$.

(16.65)

If desired, and was explained earlier in (14.99), we can incorporate exponential smoothing into the operation of the stochastic subgradient algorithm. The result is shown in listing (16.66) where the value of the positive scalar κ is smaller than but close to one.

Mini-batch stochastic subgradient algorithm with exponential smoothing for minimizing (16.2a) or (16.2b)

given dataset $\{\gamma(m), h_m\}_{m=0}^{N-1}$ or streaming data $(\gamma(n), h_n)$;
given a mini-batch size, B;
select a positive scalar κ close to but smaller than one;
start from an arbitrary initial condition, $\boldsymbol{w}_{-1}$;
start from $\bar{\boldsymbol{w}}_0 = \boldsymbol{w}_{-1}$;
start from $S_0 = 0$;
repeat until convergence over $n \geq 0$:
  select at random or receive B samples $\{\boldsymbol{\gamma}(b), \boldsymbol{h}_b\}_{b=0}^{B-1}$ at iteration n;

$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu \left(\frac{1}{B} \sum_{b=0}^{B-1} s_Q(\boldsymbol{w}_{n-1}; \boldsymbol{\gamma}(b), \boldsymbol{h}_b) \right)$$

$$S_{n+1} = \kappa S_n + 1;$$

$$\bar{\boldsymbol{w}}_{n+1} = \left(1 - \frac{1}{S_{n+1}}\right) \bar{\boldsymbol{w}}_n + \frac{1}{S_{n+1}} \boldsymbol{w}_n$$

end
return $w^\star \leftarrow \bar{\boldsymbol{w}}_{n+1}$.

(16.66)

Example 16.8 (ℓ_2-**regularized hinge loss**) Consider the ℓ_2-regularized hinge loss function:

$$Q(w;\gamma,h) = \rho\|w\|^2 \;+\; \max\Big\{0, 1-\gamma h^{\mathsf{T}} w\Big\} \tag{16.67}$$

We already know that one subgradient construction for it is

$$s_Q(w;\gamma,h) \;=\; 2\rho w - \gamma h\,\mathbb{I}\Big[\gamma h^{\mathsf{T}} w \le 1\Big] \tag{16.68}$$

Substituting into (16.64), we arrive at the stochastic subgradient implementation:

$$\boldsymbol{w}_n = (1-2\mu\rho)\boldsymbol{w}_{n-1} + \mu\,\boldsymbol{\gamma}(n)\boldsymbol{h}_n\,\mathbb{I}\Big[\boldsymbol{\gamma}(n)\boldsymbol{h}_n^{\mathsf{T}}\boldsymbol{w}_{n-1} \le 1\Big] \tag{16.69}$$

We illustrate the performance of algorithm (16.69) in Fig. 16.5, which shows the normalized learning curves in logarithmic scale under both random reshuffling and uniform sampling with and without smoothing, in addition to the mini-batch implementation.

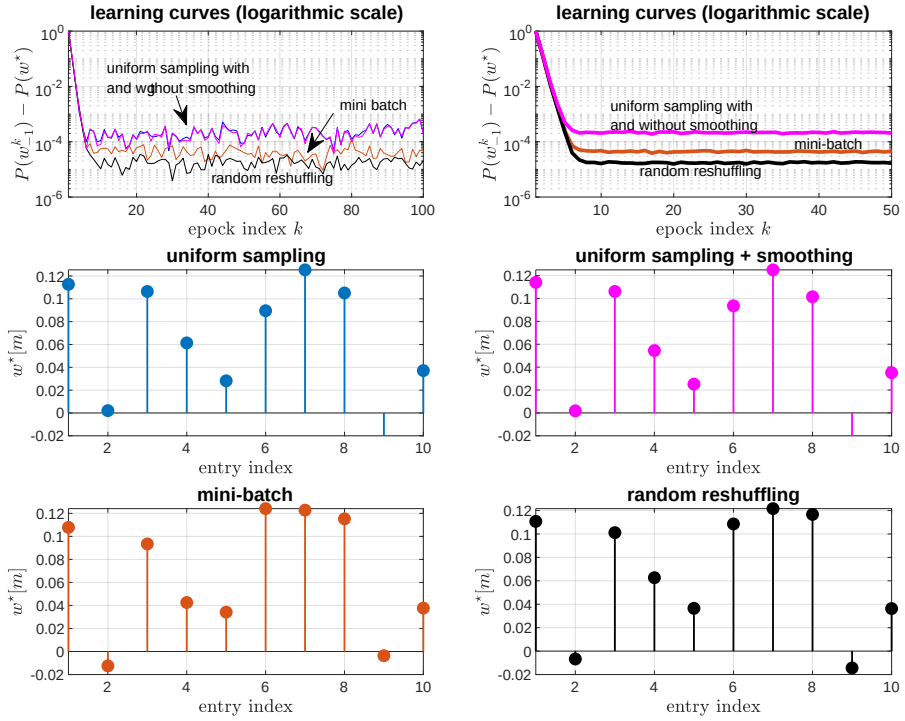

Figure 16.5 (*Top row*) Learning curves $P(w_{-1}^k)$ relative to the minimum risk value $P(w^\star)$ in normalized logarithmic scale for the stochastic subgradient implementation (16.64) under random reshuffling and uniform sampling with and without smoothing, in addition to a mini-batch implementation. (*Bottom rows*) Limit values for the weight iterates obtained under different data sampling policies.

The simulation uses $\rho = 1$, $\mu = 0.001$, $\kappa = 0.95$, and $M = 10$. It generates $N = 500$ random pairs of data $\{\gamma(m), h_m\}$ according to the logistic model described earlier in Example 16.5. A total of $K = 500$ epochs are run over the data. The learning curves are

plotted in normalized logarithmic scale in line with construction (16.49), where $w^\star$ is approximated by the limit value of the weight iterate after sufficient convergence. The mini-batch implementation employs mini-batches of size $B = 5$ samples. Each learning curve is generated by plotting the values of the risk function at the start of each epoch, namely, $P(w_{-1}^k)$. The plot on the left in the top row of the figure shows the evolution of these values relative to the minimum risk value $P(w^\star)$ for one run of the algorithm over the first 100 epochs. The noisy variations in these learning curves are a reflection of the stochastic nature of the updates. We repeat this experiment for a total of $L = 100$ times and average the learning curves over these experiments. Each experiment starts from the same initial condition w_{-1}. The result leads to the smoother curves shown in the plot on the right in the top row of the figure (for the first 50 epochs). The curves illustrate that the mini-batch and random reshuffling versions of the stochastic subgradient algorithm lead to improved steady-state performance. The plots in the bottom row show the limiting value of the weight iterates under four data sampling policies.

Consider next a variation of the hinge loss that incorporates an offset parameter θ:

$$Q(w,\theta;\gamma,h) \;=\; \rho\|w\|^2 \;+\; \max\Big\{0,\, 1-\gamma(h^{\mathsf{T}}w-\theta)\Big\} \tag{16.70}$$

If we compute subgradients relative to θ and w and write down the corresponding subgradient iterations we would get:

$$\boldsymbol{\theta}(n) = \boldsymbol{\theta}(n-1) \;-\; \mu\boldsymbol{\gamma}(n)\,\mathbb{I}\Big[\boldsymbol{\gamma}(n)\Big(\boldsymbol{h}_n^{\mathsf{T}}\boldsymbol{w}_{n-1}-\boldsymbol{\theta}(n-1)\Big)\le 1\Big] \tag{16.71a}$$

$$\boldsymbol{w}_n = (1-2\mu\rho)\,\boldsymbol{w}_{n-1} \;+\; \mu\boldsymbol{\gamma}(n)\boldsymbol{h}_n\,\mathbb{I}\Big[\boldsymbol{\gamma}(n)\Big(\boldsymbol{h}_n^{\mathsf{T}}\boldsymbol{w}_{n-1}-\boldsymbol{\theta}(n-1)\Big)\le 1\Big] \tag{16.71b}$$

We can combine these iterations by appealing to the augmented notation $w' \leftarrow \mathrm{col}\{-\theta, w\}$ and $h' \leftarrow \mathrm{col}\{1, h\}$ to arrive at:

$$\boldsymbol{w}_n' = \begin{bmatrix} 1 & 0 \\ 0 & (1-2\mu\rho)I_M \end{bmatrix}\boldsymbol{w}_{n-1}' \;+\; \mu\boldsymbol{\gamma}(n)\boldsymbol{h}_n'\,\mathbb{I}\Big[\boldsymbol{\gamma}(n)\,(\boldsymbol{h}_n')^{\mathsf{T}}\,\boldsymbol{w}_{n-1}'\le 1\Big] \tag{16.72}$$

Example 16.9 (**Perceptron recursion**) Consider the ℓ_2-regularized Perceptron loss:

$$Q(w;\boldsymbol{\gamma},\boldsymbol{h}) \;=\; \rho\|w\|^2 \;+\; \max\Big\{0,\, -\boldsymbol{\gamma}\boldsymbol{h}^{\mathsf{T}}w\Big\} \tag{16.73}$$

One subgradient construction for it is

$$s_Q(w;\gamma,h) \;=\; 2\rho w - \gamma h\,\mathbb{I}\Big[\gamma h^{\mathsf{T}}w\le 0\Big] \tag{16.74}$$

so that substituting into (16.64) we arrive at the stochastic subgradient implementation:

$$\boldsymbol{w}_n = (1-2\mu\rho)\boldsymbol{w}_{n-1} + \mu\,\boldsymbol{\gamma}(n)\boldsymbol{h}_n\,\mathbb{I}\Big[\boldsymbol{\gamma}(n)\boldsymbol{h}_n^{\mathsf{T}}\boldsymbol{w}_{n-1}\le 0\Big] \tag{16.75}$$

We can also consider a variation with an offset parameter:

$$Q(w,\theta;\boldsymbol{\gamma},\boldsymbol{h}) \;=\; \rho\|w\|^2 \;+\; \max\Big\{0,\, -\boldsymbol{\gamma}(\boldsymbol{h}^{\mathsf{T}}w-\theta)\Big\} \tag{16.76}$$

If we compute subgradients relative to θ and w and write down the corresponding subgradient iterations we get:

$$\boldsymbol{\theta}(n) = \boldsymbol{\theta}(n-1) \;-\; \mu\boldsymbol{\gamma}(n)\,\mathbb{I}\Big[\boldsymbol{\gamma}(n)\Big(\boldsymbol{h}_n^{\mathsf{T}}\boldsymbol{w}_{n-1}-\boldsymbol{\theta}(n-1)\Big)\le 0\Big] \tag{16.77a}$$

$$\boldsymbol{w}_n = (1-2\mu\rho)\,\boldsymbol{w}_{n-1} + \mu\boldsymbol{\gamma}(n)\boldsymbol{h}_n\,\mathbb{I}\Big[\boldsymbol{\gamma}(n)\Big(\boldsymbol{h}_n^{\mathsf{T}}\boldsymbol{w}_{n-1}-\boldsymbol{\theta}(n-1)\Big)\le 0\Big] \tag{16.77b}$$

We can combine these iterations by appealing to the augmented notation $w' \leftarrow \mathrm{col}\{-\theta, w\}$ and $h' \leftarrow \mathrm{col}\{1, h\}$ to arrive at:

$$\boldsymbol{w}'_n = \begin{bmatrix} 1 & 0 \\ 0 & (1-2\mu\rho)I_M \end{bmatrix} \boldsymbol{w}'_{n-1} \;+\; \mu\boldsymbol{\gamma}(n)\boldsymbol{h}'_n \,\mathbb{I}\left[\boldsymbol{\gamma}(n)\left(\boldsymbol{h}'_n\right)^{\mathsf{T}} \boldsymbol{w}'_{n-1} \le 0\right] \tag{16.78}$$

Example 16.10 (**LASSO or basis pursuit**) Consider the ℓ_1-regularized quadratic loss:

$$Q(w;\gamma,h) = \alpha\|w\|_1 \;+\; (\gamma - h^{\mathsf{T}}w)^2 \tag{16.79}$$

One subgradient construction for it is

$$s_Q(w;\gamma,h) \;=\; \alpha\,\mathrm{sign}(w) - 2h(\gamma - h^{\mathsf{T}}w) \tag{16.80}$$

so that substituting into (16.64), we arrive at:

$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu\,\mathrm{sign}(\boldsymbol{w}_{n-1}) + 2\mu\boldsymbol{h}_n(\boldsymbol{\gamma}(n) - \boldsymbol{h}_n^{\mathsf{T}}\boldsymbol{w}_{n-1}) \tag{16.81}$$

Example 16.11 (**Switching expectation and sub-differentiation**) The stochastic algorithms of this section are applicable to empirical and stochastic risks, as in (16.3a) and (16.3b). In the latter case, we need to justify switching the order of the expectation and sub-differentiation operators in order to write (in a manner similar to (16.4)):

$$\partial_w P(w) \;= \partial_w\Big(\mathbb{E}\,Q(w;\boldsymbol{\gamma},\boldsymbol{h})\Big) \;\overset{(a)}{=}\; \mathbb{E}\Big(\partial_w Q(w;\boldsymbol{\gamma},\boldsymbol{h})\Big) \tag{16.82}$$

Step (a) is possible under conditions that are generally valid for our cases of interest — see the explanation in Lemma 16.1 in the appendix. In essence, the switching is possible when the loss function $Q(w;\cdot)$ is convex and bounded in neighborhoods where the subgradients are evaluated.

16.3 STOCHASTIC PROXIMAL GRADIENT ALGORITHM

We motivate next stochastic approximations for the proximal gradient method by following similar arguments to those used for gradient-descent and the subgradient method in the last two sections. First, recall from the discussion in Sec. 15.1 that the proximal gradient algorithm is suitable for minimizing risk functions $P(w)$ that can be split into the sum of two convex components:

$$P(w) = q(w) + E(w) \tag{16.83}$$

where $E(w)$ is first-order differentiable and $q(w)$ is non-smooth. In the empirical case, the component $E(w)$ is expressed as the sample average

$$E(w) = \frac{1}{N}\sum_{m=0}^{N-1} Q_u\Big(w;\gamma(m),h_m\Big) \tag{16.84}$$

in terms of some convex loss function $Q_u(w,\cdot)$. The proximal gradient algorithms was listed in (15.10) and had the following form:

$$\left\{\begin{array}{rcl} z_n &=& w_{n-1} \;-\; \mu\,\nabla_{w^{\mathsf{T}}}\,E(w_{n-1}) \\ w_n &=& \mathrm{prox}_{\mu q}(z_n) \end{array}\right. \tag{16.85a}$$

where

$$\nabla_{w^{\mathsf{T}}}\,E(w) \;\triangleq\; \frac{1}{N}\sum_{m=0}^{N-1}\nabla_{w^{\mathsf{T}}}\,Q_u\Big(w;\gamma(m),h_m\Big) \tag{16.85b}$$

When the size N of the dataset is large, we observe again that it becomes impractical to evaluate the gradient of $E(w)$. We therefore resort to *stochastic* approximations, where this gradient is approximated either by an instantaneous value or by a mini-batch calculation. In the first case, we arrive at the stochastic proximal gradient algorithm listed in (16.87), and in the second case we arrive at (16.88). The same listings apply to the minimization of stochastic risks when $E(w)$ is defined instead as

$$E(w) = \mathbb{E}\,Q_u(w;\boldsymbol{\gamma},\boldsymbol{h}) \tag{16.86}$$

Stochastic proximal gradient for minimizing $P(w) = q(w) + E(w)$

given dataset $\{\gamma(m),h_m\}_{m=0}^{N-1}$ or streaming data $(\gamma(n),h_n)$;
start from an arbitrary initial condition, $\boldsymbol{w}_{-1}$.
repeat until convergence over $n \geq 0$:
  select at random or receive a sample $(\boldsymbol{\gamma}(n),\boldsymbol{h}_n)$ at iteration n;
  $\boldsymbol{z}_n = \boldsymbol{w}_{n-1} - \mu\,\nabla_{w^{\mathsf{T}}}\,Q_u(\boldsymbol{w}_{n-1};\boldsymbol{\gamma}(n),\boldsymbol{h}_n)$
  $\boldsymbol{w}_n = \mathrm{prox}_{\mu q}(\boldsymbol{z}_n)$
end
return $w^\star \leftarrow \boldsymbol{w}_n$.

(16.87)

Mini-batch stochastic proximal gradient for minimizing $P(w) = q(w) + E(w)$

given dataset $\{\gamma(m),h_m\}_{m=0}^{N-1}$ or streaming data $(\gamma(n),h_n)$;
given a mini-batch size, B;
start from an arbitrary initial condition, $\boldsymbol{w}_{-1}$.
repeat until convergence over $n \geq 0$:
  select at random or receive B samples $\{\boldsymbol{\gamma}(b),\boldsymbol{h}_b\}_{b=0}^{B-1}$ at iteration n;
  $\boldsymbol{z}_n = \boldsymbol{w}_{n-1} - \mu\left(\dfrac{1}{B}\displaystyle\sum_{b=0}^{B-1}\nabla_{w^{\mathsf{T}}}\,Q_u(\boldsymbol{w}_{n-1};\boldsymbol{\gamma}(b),\boldsymbol{h}_b)\right)$
  $\boldsymbol{w}_n = \mathrm{prox}_{\mu q}(\boldsymbol{z}_n)$
end
return $w^\star \leftarrow \boldsymbol{w}_n$.

(16.88)

Example 16.12 (**LASSO or basis pursuit**) Consider the quadratic risk with elastic-net regularization:

$$P(w) \;=\; \alpha\|w\|_1 \;+\; \rho\|w\|^2 \;+\; \frac{1}{N}\sum_{m=0}^{N-1}\Big(\gamma(m) - h_m^{\mathsf{T}}w\Big)^2, \quad w \in \mathbb{R}^M \tag{16.89}$$

so that

$$q(w) = \alpha\|w\|_1, \quad Q_u(w;\gamma(n),h_n) \;=\; \rho\|w\|^2 + (\gamma(n) - h_n^{\mathsf{T}}w)^2 \tag{16.90}$$

In this case, the stochastic proximal recursion (16.87) becomes

$$\boldsymbol{z}_n = (1-2\mu\rho)\boldsymbol{w}_{n-1} + 2\mu\boldsymbol{h}_n(\boldsymbol{\gamma}(n) - \boldsymbol{h}_n^{\mathsf{T}}\boldsymbol{w}_{n-1}) \tag{16.91a}$$

$$\boldsymbol{w}_n = \mathbb{T}_{\mu\alpha}(\boldsymbol{z}_n) \tag{16.91b}$$

in terms of the soft-thresholding operator applied to $\boldsymbol{z}_n$.

We illustrate the performance of the algorithm in Fig. 16.6, which shows the normalized learning curve in logarithmic scale under uniform sampling using $\rho = 0$, $\alpha = 1$, $\mu = 0.001$, and $M = 10$. The simulation generates $N = 500$ random pairs of data $\{\gamma(m), h_m\}$ according to the same linear model (16.39). The minimizer $w^\star$ for the risk (16.89) is estimated by running the batch proximal algorithm (16.85a), which employs the full gradient vector of $P(w)$, for a sufficient number of iterations. The plot on the left in the top row of the figure shows the normalized learning curve in logarithmic scale, where $P(w)$ is evaluated at the start of each epoch. The plot on the right in the first row averages these learning curves over $L = 100$ experiments to generate a smoother curve. The lower plot in the figure shows the limit value of w_n resulting from (16.91a)–(16.91b) and obtained after running $K = 300$ epochs over the data. It is seen that w_n approaches $w^\star$ and that it also exhibits sparsity.

Example 16.13 (**Logistic regression**) Consider an empirical logistic regression risk with elastic-net regularization:

$$P(w) \;=\; \alpha\|w\|_1 \;+\; \rho\|w\|^2 \;+\; \frac{1}{N}\sum_{n=0}^{N-1}\ln\Big(1 + e^{-\gamma(n)h_n^{\mathsf{T}}w}\Big), \quad w \in \mathbb{R}^M \tag{16.92}$$

so that

$$q(w) = \alpha\|w\|_1, \quad Q_u(w;\gamma(n),h_n) \;=\; \rho\|w\|^2 + \ln\Big(1 + e^{-\gamma(n)h_n^{\mathsf{T}}w}\Big) \tag{16.93}$$

In this case, the stochastic proximal recursion (16.87) becomes

$$\boldsymbol{z}_n = (1-2\mu\rho)\boldsymbol{w}_{n-1} + \mu\frac{\boldsymbol{\gamma}(n)\boldsymbol{h}_n}{1 + e^{\boldsymbol{\gamma}(n)\boldsymbol{h}_n^{\mathsf{T}}\boldsymbol{w}_{n-1}}} \tag{16.94a}$$

$$\boldsymbol{w}_n = \mathbb{T}_{\mu\alpha}(\boldsymbol{z}_n) \tag{16.94b}$$

in terms of the soft-thresholding operator applied to $\boldsymbol{z}_n$.

Example 16.14 (**Stochastic projection gradient**) Consider the constrained optimization problem

$$w^\star = \underset{w\in\mathbb{R}^M}{\text{argmin}}\left\{E(w) \;\triangleq\; \frac{1}{N}\sum_{m=0}^{N-1} Q_u(w;\gamma(m),h_m)\right\}, \quad \text{subject to } w \in \mathcal{C} \tag{16.95}$$

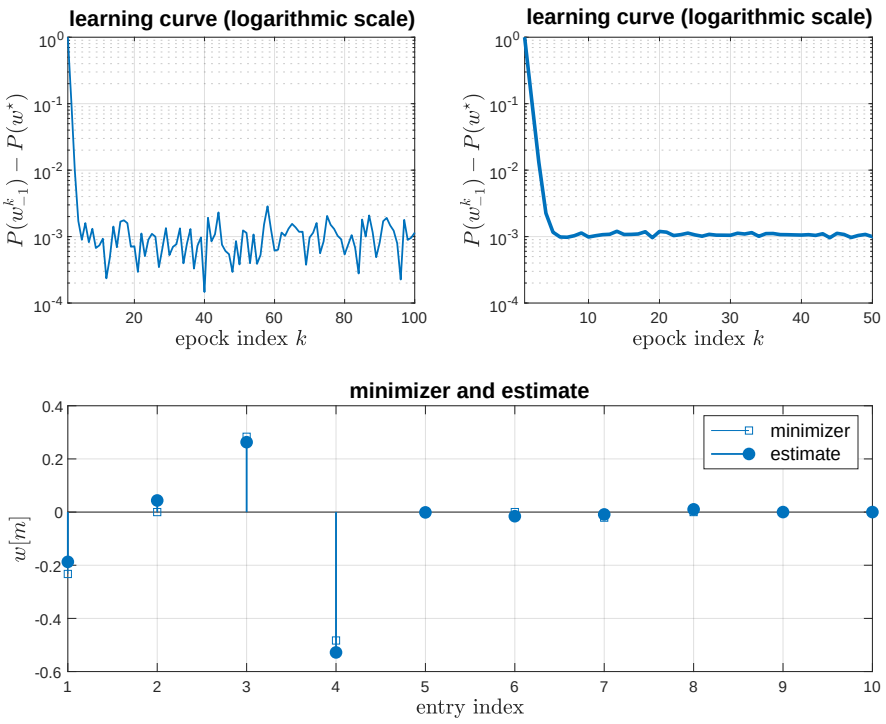

Figure 16.6 (*Top left*) Learning curve $P(w_{-1}^k)$ relative to the minimum risk value $P(w^\star)$ in normalized logarithmic scale for the stochastic proximal gradient implementation (16.91a)–(16.91b) under uniform sampling. (*Top right*) Learning curve obtained by averaging over 100 experiments. (*Bottom*) The limiting value of the weight iterate in comparison to the minimizer $w^\star$.

where $\mathcal{C}$ is a closed convex set. The projection gradient algorithm (15.51) was shown to be a special case of the proximal gradient method and it uses the gradient of $E(w)$. We can approximate this gradient by using either an instantaneous sample or a mini-batch calculation. The former case leads to listing (16.96), where $\mathcal{P}_C(x)$ denotes the projection of $x \in \mathbb{R}^M$ onto $\mathcal{C}$.

Stochastic projection gradient algorithm for solving (16.95) and (16.97)

given dataset $\{\gamma(m), h_m\}_{m=0}^{N-1}$ or streaming data $(\gamma(n), h_n)$;
start from any initial condition, w_{-1}.
repeat until convergence over $n \geq 0$:
  select at random or receive $(\boldsymbol{\gamma}(n), \boldsymbol{h}_n)$
  $\boldsymbol{w}_n = \mathcal{P}_C\Big(\boldsymbol{w}_{n-1} - \mu \nabla_{w^\mathsf{T}} Q_u(\boldsymbol{w}_{n-1}; \boldsymbol{\gamma}(n), \boldsymbol{h}_n)\Big)$
end
return $w^\star \leftarrow \boldsymbol{w}_n$.

(16.96)

This algorithm can also handle stochastic risks of the form

$$w^o = \operatorname*{argmin}_{w\in\mathbb{R}^M} \left\{ E(w) \;\triangleq\; \mathbb{E}\, Q_u(w;\boldsymbol{\gamma},\boldsymbol{h}) \right\}, \quad \text{subject to } w\in\mathcal{C} \tag{16.97}$$

In this case, the data samples $(\gamma(n), h_n)$ would stream in successively over time.

Example 16.15 (**Stochastic mirror descent**) Consider the same constrained optimization problems (16.95) or (16.97). The mirror-descent algorithm (15.103) also relies on the gradient of $E(w)$. We can again approximate this gradient by using either an instantaneous sample or a mini-batch calculation. The former case leads to listing (16.98). The listing can be specialized for particular choices of the mirror function $\phi(x)$, such as choosing it as the negative entropy function or as a quadratic function. In a similar manner, we can write down stochastic versions for the lazy mirror descent and mirror prox algorithms.

Stochastic mirror-descent for solving (16.95) or (16.97)

given dataset $\{\gamma(m), h_m\}_{m=0}^{N-1}$ or streaming data $(\gamma(n), h_n)$;
choose $\nu_\phi-$strongly convex mirror function $\phi(w):\mathcal{C}_\phi\to\mathbb{R}$;
let $\phi^\star(x)=\sup_w\{w^{\mathsf{T}}x-\phi(w)\}$ denote its conjugate function;
let $\mathcal{C}'=\mathcal{C}\cap\mathcal{C}_\phi$;
start from an initial condition, $\boldsymbol{w}_{-1}\in\mathcal{C}'$.
repeat until convergence over $n\geq 0$:
  select at random or receive $(\boldsymbol{\gamma}(n),\boldsymbol{h}_n)$
  $\boldsymbol{b}_n=\nabla_{w^{\mathsf{T}}}\,\phi(\boldsymbol{w}_{n-1})-\mu\nabla_{w^{\mathsf{T}}}\,Q_u(\boldsymbol{w}_{n-1};\boldsymbol{\gamma}(n),\boldsymbol{h}_n)$
  $\boldsymbol{z}_n=\nabla_{x^{\mathsf{T}}}\,\phi^\star(\boldsymbol{b}_n)$
  $\boldsymbol{w}_n=\mathcal{P}_{C',\phi}(\boldsymbol{z}_n)$, (Bregman projection)
end
return $w^\star\leftarrow\boldsymbol{w}_n$. (16.98)

Example 16.16 (**Stochastic coordinate-descent**) The same stochastic approximation approach can be applied to coordinate-descent algorithms. It is sufficient to illustrate the construction by considering the randomized proximal version listed earlier in (15.31); similar constructions apply to other variants of coordinate-descent. Thus, consider an empirical risk $P(w)$ that separates into the form:

$$P(w) \;=\; E(w)+\sum_{m=1}^{M} q_m(w_m) \tag{16.99a}$$

where the second component is separable over the individual coordinates of w denoted by $\{w_m\}$. Moreover, the smooth component $E(w)$ is expressed as the sample average of loss values:

$$E(w)=\frac{1}{N}\sum_{m=0}^{N-1} Q_u(w;\gamma_m,h_m) \tag{16.99b}$$

We approximate the gradient of $E(w)$ by using either an instantaneous approximation or a mini-batch calculation. The former case leads to (16.101), which is applicable to both empirical risks as in (16.99a)–(16.99b), or to stochastic risks where

$$E(w)=\mathbb{E}\, Q_u(w;\boldsymbol{\gamma},\boldsymbol{h}) \tag{16.100}$$

The main difference is that $(\boldsymbol{\gamma}(n),\boldsymbol{h}_n)$ will now stream in successively over time.

Stochastic randomized proximal coordinate descent for minimizing (16.99a).

given dataset $\{\gamma(m), h_m\}_{m=0}^{N-1}$ or streaming data $(\gamma(n), h_n)$;
start with an arbitrary initial condition $\boldsymbol{w}_{-1}$.
repeat until convergence over $n \geq 0$:
  iterate is $\boldsymbol{w}_{n-1} = \text{col}\{\boldsymbol{w}_{n-1,m}\}_{m=1}^{M}$
  select at random or receive $(\boldsymbol{\gamma}(n), \boldsymbol{h}_n)$;
  select a random index $1 \leq m^o \leq M$;
  $\boldsymbol{z}_{n,m^o} = \boldsymbol{w}_{n-1,m^o} - \mu\, \partial Q_u(\boldsymbol{w}_{n-1}; \boldsymbol{\gamma}(n), \boldsymbol{h}_n)/\partial w_{m^o}$
  $\boldsymbol{w}_{n,m^o} = \text{prox}_{\mu q_{m^o}}(\boldsymbol{z}_{n,m^o})$
  keep $\boldsymbol{w}_{n,m} = \boldsymbol{w}_{n-1,m}$ for all $m \neq m^o$
end
return $w^\star \leftarrow \boldsymbol{w}_n$. (16.101)

Example 16.17 (**Stochastic conjugate gradient**) We can similarly devise a stochastic implementation for the Fletcher-Reeves conjugate gradient algorithm (13.87) by approximating the gradient of $P(w)$ using an instantaneous or mini-batch calculation. The former case is listed in (16.102), which can handle both empirical and stochastic risk minimization. In the latter case, the data samples $(\gamma(n), h_n)$ stream in successively over time.

Stochastic Fletcher-Reeves algorithm for minimizing (16.2a) or (16.2b)

given dataset $\{\gamma(m), h_m\}_{m=0}^{N-1}$ or streaming data $(\gamma(n), h_n)$;
start with an arbitrary initial condition $\boldsymbol{w}_{-1}$;
set $\boldsymbol{q}_{-1} = 0$;
repeat until convergence over $n \geq 0$:
  select at random or receive $(\boldsymbol{\gamma}(n), \boldsymbol{h}_n)$;
  $\boldsymbol{r}_{n-1} = -\nabla_{w^\mathsf{T}} Q(\boldsymbol{w}_{n-1}; \boldsymbol{\gamma}(n), \boldsymbol{h}_n)$
  if $n = 0$ then $\boldsymbol{\beta}_{-1} = 0$
    else $\boldsymbol{\beta}_{n-1} = \|\boldsymbol{r}_{n-1}\|^2 / \|\boldsymbol{r}_{n-2}\|^2$
  end
  $\boldsymbol{q}_n = \boldsymbol{r}_{n-1} + \boldsymbol{\beta}_{n-1}\, \boldsymbol{q}_{n-1}$
  find $\boldsymbol{\alpha}_n$ by solving $\min_{\alpha \in \mathbb{R}} Q(\boldsymbol{w}_{n-1} + \alpha \boldsymbol{q}_n)$ using line search.
  $\boldsymbol{w}_n = \boldsymbol{w}_{n-1} + \boldsymbol{\alpha}_n \boldsymbol{q}_n$
end
return $w^\star \leftarrow \boldsymbol{w}_n$ (16.102)

16.4 GRADIENT NOISE

In all stochastic algorithms studied in this chapter, the desired gradient or subgradient search direction is approximated by using either instantaneous or mini-batch calculations. For example, when $P(w)$ is smooth, we used approximations

of the form:

$$(\textbf{instantaneous}):\ \widehat{\nabla_{w^{\mathsf{T}}}\,P}(w)=\nabla_{w^{\mathsf{T}}}\,Q(w;\boldsymbol{\gamma},\boldsymbol{h}) \tag{16.103a}$$

$$(\textbf{mini-batch}):\ \widehat{\nabla_{w^{\mathsf{T}}}\,P}(w)=\frac{1}{B}\sum_{b=0}^{B-1}\nabla_{w^{\mathsf{T}}}\,Q\Big(w;\boldsymbol{\gamma}(b),\boldsymbol{h}_b\Big) \tag{16.103b}$$

and when $P(w)$ is nonsmooth we replaced the gradients of $Q(w;\cdot)$ by subgradients, $s_Q(w;\boldsymbol{\gamma},\boldsymbol{h})$. We continue with the smooth case for illustration purposes. The difference between the true gradient and its approximation is called *gradient noise* and denoted by

$$\boldsymbol{g}(w)\ \triangleq\ \widehat{\nabla_{w^{\mathsf{T}}}\,P}(w)-\nabla_{w^{\mathsf{T}}}\,P(w) \tag{16.104}$$

The presence of this noise alters the dynamics of optimization algorithms. To see this, the following two relations highlight the difference between the original gradient-descent method and its stochastic version:

$$(\textbf{gradient descent}):\ w_n=w_{n-1}-\mu\nabla_{w^{\mathsf{T}}}\,P(w_{n-1}) \tag{16.105a}$$

$$(\textbf{stochastic version}):\ \boldsymbol{w}_n=\boldsymbol{w}_{n-1}-\mu\nabla_{w^{\mathsf{T}}}\,P(\boldsymbol{w}_{n-1})-\mu\boldsymbol{g}(\boldsymbol{w}_{n-1}) \tag{16.105b}$$

where the gradient noise process appears as a driving term in the second recursion. The noise seeps into the operation of the algorithm and some degradation in performance is expected. For instance, while we were able to show in a previous chapter that the gradient-descent implementation (16.105a) converges to the exact minimizer $w^\star$ for sufficiently small step-sizes, we will discover that the stochastic version (16.105b) can only approach a small neighborhood around $w^\star$. Specifically, we will prove in later chapters that, for this case, the mean-square-deviation $\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_n\|^2$ approaches $O(\mu)$:

$$\limsup_{n\to\infty}\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_n\|^2=O(\mu) \tag{16.106}$$

Obviously, the smaller μ is, the smaller the size of the limiting error will be. However, small step-sizes slow down convergence and there is a need to strike a balance between convergence rate and error size. One way to reduce the size of the error is to employ decaying step-sizes; in this case, future results will show that it is possible for stochastic algorithms to converge to the exact minimizer even in the presence of gradient noise. Nevertheless, vanishing step-sizes reduce the ability of an optimization algorithm to continue to update and learn, which is problematic for scenarios involving drifting minimizers. Another way to reduce the limiting error, even with constant step-sizes, is to resort to variance-reduction techniques. We will discuss these various elements in future chapters and establish performance limits. The derivations are demanding and are not necessary at this stage.

For illustration and comparison purposes, we collect in Tables 16.2 and 16.3 some of the results that will be derived in future analyses for strongly-convex

risks. The first table assumes constant step-sizes, while the second table assumes a vanishing step-size of the form $\mu(k) = \tau/k$ with its value decaying with the epoch index $k \geq 1$. The tables also list the conditions under which the results will be derived; these conditions are generally satisfied for our problems of interest. The tables further indicate the locations in the text where the results can be found. Observe that all algorithms deliver exponential (i.e., linear) convergence rates under strong convexity conditions, and that the size of the limiting neighborhood depends on several factors including whether uniform sampling or random reshuffling is used, whether the risk function is smooth or not, and on whether mini-batch processing is employed. More results will be discussed in future Chapters 19 through 22.

Remark 16.1. (Big-O and little-o notation) The results in the tables employ the Big-O notation with its argument being either a sequence, such as λ^k, or a function of the step-size parameter μ. We already explained in Remark 12.3 what the notation means for sequences, namely, it compares the asymptotic growth rate of two sequences. Thus, writing $a_k = O(b_k)$, with a big O for a sequence b_k with positive entries, means that there exists some constant $c > 0$ and index k_o such that $|a_k| \leq cb_k$ for $k > k_o$. This also means that the decay rate of a_k is at least as fast or faster than b_k. On the other hand, the notation $f(\mu) \in O(g(\mu))$ for some positive function $g(\mu)$ means that there exists a constant c independent of μ such that $\lim_{\mu \to 0} |f(\mu)|/g(\mu) \leq c$. In a similar vein, using the little-o notation and writing $f(\mu) \in o(g(\mu))$ means that $\lim_{\mu \to 0} |f(\mu)|/g(\mu) = 0$. Using these definition, we note for example that

$$\mu \in O(\sqrt{\mu}), \quad 10\mu \in O(\mu), \quad \mu \in o(\sqrt{\mu}), \quad \mu^2 \in o(\mu), \quad \mu^{-1/2} \in o(\mu^{-1}) \tag{16.107}$$

We will use the compact symbols $O(g(\mu))$ and $o(g(\mu))$ as placeholders for the more explicit notation $f(\mu) \in O(g(\mu))$ and $f(\mu) \in o(g(\mu))$, respectively. Also, writing $a(\mu) \leq b(\mu) + o(g(\mu))$ means that there exists $f(\mu) \in o(g(\mu))$ such that $a(\mu) \leq b(\mu) + f(\mu)$. ■

16.5 REGRET ANALYSIS

We will examine the convergence performance of stochastic approximation algorithms in some detail in future chapters by studying the behavior of the mean-square-error, $\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_n\|^2$, and the mean excess risk, $\mathbb{E}\,P(\boldsymbol{w}_n) - P(w^\star)$. In this section, we comment on another approach for the study of these algorithms, which is based on regret analysis. We explain the value of this type of analysis and also some of its limitations.

We have already encountered the regret measure in earlier chapters — see, e.g., expression (12.58) for the gradient-descent algorithm. The definition of the regret needs to be adjusted for *stochastic approximation methods*. It is sufficient to illustrate the construction for the case of smooth loss functions and for the stochastic gradient algorithm.

Table 16.2 Convergence properties of stochastic optimization algorithms with *constant* step-sizes. The risk $P(w)$ is $\nu-$strongly convex and the loss is $Q(w;\cdot,\cdot)$. For proximal implementations, we assume $P(w) = q(w) + E(w)$ and denote the loss associated with $E(w)$ by $Q_u(w;\cdot,\cdot)$.

algorithm	conditions on risk and loss functions	asymptotic convergence property	reference
Stochastic gradient algorithm (16.27) (uniform sampling)	$P(w)$: $\nu-$strongly convex $Q(w)$: $\delta-$Lipschitz gradients (18.10a)–(18.10b)	$\mathbb{E}\,P(\boldsymbol{w}_n^k) - P(w^\star) \leq O(\lambda^k) + O(\mu)$ $k \geq 1$: epoch index $n \geq 0$: iteration index	Thm. 19.1
Stochastic gradient algorithm (16.27); (random reshuffling)	same as above	$\mathbb{E}\,P(\boldsymbol{w}_n^k) - P(w^\star) \leq O(\lambda^k) + O(\mu^2)$	Thm. 19.5
Stochastic gradient algorithm (16.27); (importance sampling)	same as above	$\mathbb{E}\,P(\boldsymbol{w}_n^k) - P(w^\star) \leq O(\lambda^k) + O(\mu)$	Thm. 19.1 Sec. 19.6
Stochastic gradient algorithm (16.27) (data streaming)	$P(w)$: $\nu-$strongly convex $Q(w)$: $\delta-$Lipschitz gradients in mean-square sense (18.13a)–(18.13b)	$\mathbb{E}\,P(\boldsymbol{w}_n^k) - P(w^o) \leq O(\lambda^k) + O(\mu)$	Thm. 19.1
Stochastic subgradient algorithm with smoothing (16.66), $B = 1$ (uniform sampling)	$P(w)$: $\nu-$strongly convex $Q(w)$: affine-Lipschitz subgradients (18.66a)–(18.66b)	$\mathbb{E}\,P(\boldsymbol{w}_n^k) - P(w^\star) \leq O(\kappa^k) + O(\mu)$ (using κ close to 1)	Eq. (20.68)
Stochastic proximal gradient (16.87) (uniform sampling)	$E(w)$: $\nu-$strongly convex $Q_u(w)$: $\delta-$Lipschitz gradients (21.11a)–(21.11b) and (21.13) $P(w) = q(w) + E(w)$	$\mathbb{E}\,P(\boldsymbol{w}_n^k) - P(w^\star) \leq O(\lambda^{k/2}) + O(\sqrt{\mu})$	Thm. 21.1
Mini-batch stochastic gradient (16.28)	$P(w)$: $\nu-$strongly convex $Q(w)$: $\delta-$Lipschitz gradients (18.10a)–(18.10b)	$\mathbb{E}\,P(\boldsymbol{w}_n) - P(w^\star) \leq O(\lambda^n) + O(\mu/B)$	Thm. 19.2
Mini-batch stochastic subgradient algorithm with smoothing (16.66)	$P(w)$: $\nu-$strongly convex $Q(w)$: affine-Lipschitz subgradients (18.66a)–(18.66b)	$\mathbb{E}\,P(\boldsymbol{w}_n) - P(w^\star) \leq O(\kappa^n) + O(\mu/B)$ (using κ close to 1)	Prob. 20.2
Mini-batch proximal gradient (16.88)	$E(w)$: $\nu-$strongly convex $Q_u(w)$: $\delta-$Lipschitz gradients (21.11a)–(21.11b) and (21.13) $P(w) = q(w) + E(w)$	$\mathbb{E}\,P(\boldsymbol{w}_n) - P(w^\star) \leq O(\lambda^{n/2}) + O(\sqrt{\mu/B}\,)$	Thm. 21.2

Table 16.3 Convergence properties of various stochastic optimization algorithms with step-size $\mu(k) = \tau/k$, which decays with the epoch index, k. In the listing, n refers to the iteration index and $k \geq 1$ refers to the epoch index. In this table, the risk $P(w)$ is assumed to be $\nu-$strongly convex and the loss function is denoted by $Q(w;\cdot,\cdot)$. For proximal implementations, we assume the risk is split as $q(w) + E(w)$ and denote the loss associated with $E(w)$ by $Q_u(w;\cdot,\cdot)$.

algorithm	conditions on risk and loss functions	asymptotic convergence property	reference
Stochastic gradient algorithm (16.27) (uniform sampling) (decaying step-size)	$P(w)$: $\nu-$strongly convex $Q(w)$: $\delta-$Lipschitz gradients (18.10a)–(18.10b) $\mu(k) = \tau/k$	$\mathbb{E}\, P(\boldsymbol{w}_n^k) - P(w^\star) \leq O(1/k)$ $k \geq 1$: epoch index $n \geq 0$: iteration index	Thm. 19.4
Stochastic subgradient algorithm (16.64) (uniform sampling) (decaying step-size)	$P(w)$: $\nu-$strongly convex $Q(w)$: affine-Lipschitz subgradients (18.66a)–(18.66b) $\mu(k) = \tau/k$	$\mathbb{E}\, P(\boldsymbol{w}_n^k) - P(w^\star) \leq O(1/\sqrt{k}\,)$	Thm. 20.4
Stochastic proximal gradient (16.87) (uniform sampling) (decaying step-size)	$E(w)$: $\nu-$strongly convex $Q_u(w)$: $\delta-$Lipschitz gradients (21.11a)–(21.11b) and (21.13) $\mu(k) = \tau/k$	$\mathbb{E}\, P(\boldsymbol{w}_n^k) - P(w^\star) \leq O(1/\sqrt{k}\,)$	Thm. 21.4

Consider the empirical risk minimization problem:

$$w^\star \triangleq \operatorname*{argmin}_{w\in\mathcal{C}} \left\{ P(w) \triangleq \frac{1}{N}\sum_{m=0}^{N-1} Q(w;\gamma(m),h_m) \right\} \tag{16.108}$$

where $Q(w,\cdot)$ is a smooth convex function over w, and where we are now adding a constraint by requiring w to belong to some convex set $\mathcal{C}$. The reason for the addition of this constraint will become evident soon. One way to solve constrained problems of this type is to resort to the stochastic projection method (16.96):

$$\boldsymbol{w}_n = \mathcal{P}_C\Big\{ \boldsymbol{w}_{n-1} - \mu(n)\,\nabla_{w^\mathsf{T}} Q(\boldsymbol{w}_{n-1};\boldsymbol{\gamma}(n),\boldsymbol{h}_n) \Big\} \tag{16.109}$$

where, for generality, we are allowing for an iteration-dependent step-size, $\mu(n)$. In this description, the operator $\mathcal{P}_C(x)$ projects the vector $x\in\mathbb{R}^M$ onto the convex set $\mathcal{C}$. We already know from the non-expansive property (9.70) of projection operators that

$$\|w-\mathcal{P}_C(x)\| \le \|w-x\|, \quad \text{for any } x\in\mathbb{R}^M, w\in\mathcal{C} \tag{16.110}$$

in terms of Euclidean distances.

Observe next that the gradient vector in (16.109) changes with n. We simplify the notation and introduce

$$Q_n(w) \triangleq Q(w;\gamma(n),h_n) \tag{16.111}$$

so that the stochastic algorithm can be rewritten more compactly as

$$\boxed{\boldsymbol{w}_n = \mathcal{P}_C\Big\{ \boldsymbol{w}_{n-1} - \mu(n)\,\nabla_{w^\mathsf{T}} Q_n(\boldsymbol{w}_{n-1}) \Big\}, \quad n\ge 0} \tag{16.112}$$

For such *stochastic* optimization procedures, the average *regret* is defined directly in terms of the loss function $Q_n(w;\cdot)$; this is in contrast to the earlier definition in the gradient-descent case (12.57) where the regret was defined in terms of the risk function itself. This is because the updates there involved the gradient of the risk function whereas the updates here involve the gradient of the loss function and *sampled* data points $(\gamma(n),h_n)$. Thus, for stochastic optimization algorithms, we define the average regret that is based on a dataset of size N as the difference between the accumulated *loss* and the smallest possible value for it:

$$\boxed{\boldsymbol{\mathcal{R}}(N) \triangleq \frac{1}{N}\sum_{n=0}^{N-1} Q_n(\boldsymbol{w}_{n-1}) - \min_{w\in\mathcal{C}} \left\{ \frac{1}{N}\sum_{n=0}^{N-1} Q_n(w) \right\}} \tag{16.113}$$

where the iterate $\boldsymbol{w}_{n-1}$ changes with the iteration index within the first sum. In a manner similar to (16.108), we denote the minimizer for the rightmost term in the above expression by $w^\star$. Observe that this term involves the samples

$\{\gamma(n), h_n\}$ that were used during the N iterations of the stochastic algorithms. We then have

$$\boldsymbol{\mathcal{R}}(N) \triangleq \frac{1}{N}\sum_{n=0}^{N-1}\Big\{Q_n(\boldsymbol{w}_{n-1}) - Q_n(w^\star)\Big\} \tag{16.114}$$

The boldface notation for $\boldsymbol{\mathcal{R}}$ is meant to reflect the random nature of the regret due to its dependence on the random iterates, $\{\boldsymbol{w}_{n-1}\}$. We will show below that, in the process of bounding the regret, the weight iterates will disappear and we will be able to bound the regret by some deterministic value.

The purpose of regret analysis is to examine how the regret evolves with N, for any sequence of iterates $\{\boldsymbol{w}_{n-1}\}$. For example, if it can be shown that the average regret decays to zero at the rate $\boldsymbol{\mathcal{R}}(N) = O(1/\sqrt{N})$, then this would imply the desirable conclusion that, asymptotically, the average accumulated loss by the algorithm is able to approach the smallest possible risk value. We can bound the regret as follows.

Regret bound

First, we rewrite the stochastic gradient algorithm (16.112) in the following equivalent form involving an intermediate variable $\boldsymbol{z}_n$:

$$\begin{cases} \boldsymbol{z}_n = \boldsymbol{w}_{n-1} - \mu(n)\nabla_{w^\mathsf{T}} Q_n(\boldsymbol{w}_{n-1}) \\ \boldsymbol{w}_n = \mathcal{P}_C(\boldsymbol{z}_n) \end{cases} \tag{16.115}$$

where, using property (16.110), it holds that

$$\|w - \boldsymbol{w}_n\| \leq \|w - \boldsymbol{z}_n\|, \quad \text{for any } w \in \mathcal{C} \tag{16.116}$$

Subtracting $w^\star$ from both sides of the first relation in (16.115) gives

$$(w^\star - \boldsymbol{z}_n) = (w^\star - \boldsymbol{w}_{n-1}) + \mu(n)\nabla_{w^\mathsf{T}} Q_n(\boldsymbol{w}_{n-1}) \tag{16.117}$$

Let $\widetilde{\boldsymbol{w}}_n = w^\star - \boldsymbol{w}_n$. Squaring both sides of the above equation and using (16.116) leads to

$$\begin{aligned} \|\widetilde{\boldsymbol{w}}_n\|^2 \leq \|\widetilde{\boldsymbol{w}}_{n-1}\|^2 + 2\mu(n)\Big(\nabla_{w^\mathsf{T}} Q(\boldsymbol{w}_{n-1})\Big)^\mathsf{T}\widetilde{\boldsymbol{w}}_{n-1} + \\ \mu^2(n)\|\nabla_{w^\mathsf{T}} Q_n(\boldsymbol{w}_{n-1})\|^2 \end{aligned} \tag{16.118}$$

Invoking the convexity of $Q(w,\cdot)$ over w and using property (8.5) for convex functions we have

$$Q_n(\boldsymbol{w}_{n-1}) - Q_n(w^\star) \leq -\Big(\nabla_{w^\mathsf{T}} Q_n(\boldsymbol{w}_{n-1})\Big)^\mathsf{T}\widetilde{\boldsymbol{w}}_{n-1} \tag{16.119}$$

Substituting into the regret expression (16.114), we obtain

$$\begin{aligned} N\mathcal{R}(N) &\leq -\sum_{n=0}^{N-1}\Big(\nabla_{w^{\mathsf{T}}} Q_n(\boldsymbol{w}_{n-1})\Big)^{\mathsf{T}}\widetilde{\boldsymbol{w}}_{n-1} \\ &\overset{(16.118)}{\leq} \sum_{n=0}^{N-1}\left\{\frac{1}{2\mu(n)}\Big(\|\widetilde{\boldsymbol{w}}_{n-1}\|^2 - \|\widetilde{\boldsymbol{w}}_n\|^2\Big) + \frac{\mu(n)}{2}\|\nabla_{w^{\mathsf{T}}} Q_n(\boldsymbol{w}_{n-1})\|^2\right\} \\ &= \frac{1}{2\mu(0)}\|\widetilde{\boldsymbol{w}}_{-1}\|^2 - \frac{1}{2\mu(N-1)}\|\widetilde{\boldsymbol{w}}_{N-1}\|^2 + \\ &\quad \frac{1}{2}\sum_{n=1}^{N-1}\Big(\frac{1}{\mu(n)} - \frac{1}{\mu(n-1)}\Big)\|\widetilde{\boldsymbol{w}}_{n-1}\|^2 + \\ &\quad \frac{1}{2}\sum_{n=0}^{N-1}\mu(n)\|\nabla_{w^{\mathsf{T}}} Q_n(\boldsymbol{w}_{n-1})\|^2 \end{aligned} \tag{16.120}$$

Introduce the two constants:

$$d \triangleq \max_{x,y\in\mathcal{C}} \|x-y\| \tag{16.121a}$$

$$c \triangleq \max_{w\in\mathcal{C},0\leq n\leq N-1} \|\nabla_{w^{\mathsf{T}}} Q_n(w)\| \tag{16.121b}$$

where d is the largest distance between any two points in the convex set $\mathcal{C}$, and c is the largest norm of the gradient of the loss function over both $\mathcal{C}$ and the data. Then, we can simplify the bound (16.120) on the regret function by noting that

$$N\mathcal{R}(N) \leq \frac{d^2}{2}\left\{\frac{1}{\mu(0)} + \sum_{n=1}^{N-1}\left(\frac{1}{\mu(n)} - \frac{1}{\mu(n-1)}\right)\right\} + \frac{c^2}{2}\sum_{n=0}^{N-1}\mu(n) \tag{16.122}$$

and, consequently,

$$\boxed{\mathcal{R}(N) \leq \frac{1}{N}\left\{\frac{d^2}{2\mu(N-1)} + \frac{c^2}{2}\sum_{n=0}^{N-1}\mu(n)\right\}} \tag{16.123}$$

Vanishing step-size

For illustration purposes, assume the step-size sequence decays as $\mu(n) = 1/\sqrt{n+1}$. Then,

$$\sum_{n=0}^{N-1}\mu(n) = \sum_{n=1}^{N}\frac{1}{\sqrt{n}} \;\leq\; 1 + \int_1^N \frac{1}{\sqrt{x}}dx \;=\; 2\sqrt{N} - 1 \tag{16.124}$$

and we arrive at

$$\mathcal{R}(N) \leq \frac{1}{N}\left\{\frac{d^2\sqrt{N}}{2} + \frac{c^2}{2}\Big(2\sqrt{N} - 1\Big)\right\} \tag{16.125}$$

It follows that the average regret converges to zero as $N \to \infty$ at the rate

$$\boxed{\boldsymbol{\mathcal{R}}(N) \leq O(1/\sqrt{N})} \tag{16.126}$$

Constant step-size

On the other hand, when the step-size is constant, say, $\mu(n) = \mu$, then the bound (16.123) leads to

$$\boxed{\boldsymbol{\mathcal{R}}(N) \leq O(1/N) \;+\; O(\mu)} \tag{16.127}$$

which shows that the average regret approaches a small value on the order of μ and the convergence rate towards this region is $O(1/N)$. We will encounter a more detailed example of regret analysis in future Appendix 17.A when we apply it to examine the performance of adaptive gradient algorithms.

There are at least two differences in the regret analysis approach in comparison to the mean-square-error analysis performed in future chapters; see also Prob. 16.2. First, the regret argument relies on a worst-case scenario in the sense that the effect of the random trajectory $\{\widetilde{\boldsymbol{w}}_{-1}, \widetilde{\boldsymbol{w}}_0, \ldots, \widetilde{\boldsymbol{w}}_{N-1}\}$ is removed completely by replacing their norms by d. Second, the size of the constants d and c can be very large. For example, if we were to remove the constraint $w \in \mathcal{C}$ and replace $\mathcal{C}$ by $\mathbb{R}^M$, then the above argument would not carry through since d or c will become unbounded.

16.6 COMMENTARIES AND DISCUSSION

Stochastic approximation theory. The idea of using data realizations to approximate actual gradient or subgradient vectors is at the core of *stochastic approximation theory*. According to Tsypkin (1971, p. 70) and Lai (2003), the pioneering work in the field is the landmark paper by Robbins and Monro (1951), which developed a recursive method for finding roots of functions, i.e., points $w^\star$ where $P(w^\star) = 0$. Their procedure was a variation of a scheme developed two decades earlier by von Mises and Pollaczek-Geiringer (1929), and it can be succinctly described as follows. Consider a risk function $P(w)$, of a *scalar* parameter w, and assume $P(w)$ is represented as the mean of some loss function, say, in the form:

$$P(w) \;\triangleq\; \mathbb{E}\, Q(w; \boldsymbol{x}) \tag{16.128}$$

Robbins and Monro (1951) argued that the root $w^\star$ can be approximated by evaluating the loss function at successive realizations x_n and employing the update relation:

$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu(n) Q(\boldsymbol{w}_{n-1}; \boldsymbol{x}_n), \;\; n \geq 0 \tag{16.129}$$

where $\mu(n)$ is a step-size sequence that satisfies:

$$\sum_{n=0}^{\infty} \mu(n) = \infty, \quad \sum_{n=0}^{\infty} \mu^2(n) < \infty \tag{16.130}$$

They showed that the algorithm converges in the mean-square-error sense, and also in probability, namely, $\mathbb{E}\,\widetilde{\boldsymbol{w}}_n^2 \to 0$ and, for any $\epsilon > 0$:

$$\lim_{n\to\infty} \mathbb{P}\Big(|\widetilde{\boldsymbol{w}}_n| \geq \epsilon\Big) = 0 \tag{16.131}$$

Stronger almost-sure convergence results were later given by Blum (1954), Dvoretzky (1956), and Gladyshev (1965), among others, showing that under certain technical conditions:

$$\mathbb{P}\left(\lim_{n\to\infty} \widetilde{\boldsymbol{w}}_n = 0\right) = 1 \tag{16.132}$$

The same construction can be extended from root-finding to the solution of minimization problems. Assume $P(w)$ is convex and has a unique minimum at some location $w^\star$. Then, finding $w^\star$ is equivalent to finding the root of $dP(w)/dw = 0$, which suggests using the stochastic gradient recursion:

$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu(n)\,\frac{dQ(w;\boldsymbol{x}_n)}{dw}\bigg|_{w=\boldsymbol{w}_{n-1}} \tag{16.133}$$

Motivated by the work of Robbins and Monro (1951), an alternative stochastic construction was proposed by Kiefer and Wolfowitz (1952) to solve minimization problems from noisy measurements. Their procedure relies on the same general concept of using data to approximate unknown quantities but took a different form, namely, they proposed using the recursion:

$$\boldsymbol{w}_{n-1}^{+} = \boldsymbol{w}_{n-1} + \tau(n) \tag{16.134a}$$

$$\boldsymbol{w}_{n-1}^{-} = \boldsymbol{w}_{n-1} - \tau(n) \tag{16.134b}$$

$$\Delta Q(n) = Q(\boldsymbol{w}_{n-1}^{+};\boldsymbol{x}_n) - Q(\boldsymbol{w}_{n-1}^{-};\boldsymbol{x}_n) \tag{16.134c}$$

$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu(n)\,\frac{\Delta Q(n)}{\tau(n)} \tag{16.134d}$$

In this recursion, a first-order finite difference calculation is used to approximate the derivative of $Q(w)$ with $\tau(n)$ denoting the interval width. The nonnegative sequences $\{\mu(n), \tau(n)\}$ are chosen to tend asymptotically to zero and to satisfy the conditions:

$$\sum_{n=0}^{\infty} \mu(n) = \infty, \quad \sum_{n=0}^{\infty} \left(\frac{\mu(n)}{\tau(n)}\right)^2 < \infty \tag{16.135}$$

The work by Robbins and Monro (1951) generated tremendous interest in the statistical, optimization, and engineering literature and led to many subsequent studies and extensions. While their work dealt primarily with a *scalar* weight w, Blum (1954) and Schmetterer (1961) extended the procedure to weight *vectors*. A description of these developments can be found in the book by Wetherhill (1966). Further discussions on stochastic approximation methods, including some detailed treatments of their convergence properties, can be found in the works by Albert and Gardner (1967), Wasan (1969), Mendel and Fu (1970), Tsypkin (1971), Ljung (1977), Kushner and Clark (1978), Kushner (1984), Polyak (1987), Benveniste, Métivier, and Priouret (1990), Bertsekas and Tsitsiklis (1997,2000), Kushner and Yin (2003), Spall (2003), Marti (2005), and Sayed (2003,2008,2014a). The use of averaged iterates as in (16.52) was proposed independently by Ruppert (1988) and Polyak and Juditsky (1992).

Stochastic gradient and subgradient algorithms have become commonplace in online inference and learning solutions for at least three reasons. First, the explosive interest in large-scale and big data scenarios favors the use of simple algorithmic structures, of which these methods are prime examples. Second, as shown in future chapters, these algorithms are able to deliver solid performance guarantees, with the mean-square error $\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_n\|^2$ and the excess risk $\mathbb{E}\,P(\boldsymbol{w}_n) - P(w^\star)$ approaching small neighborhoods on the order of $O(\mu)$. Third, and importantly, it is increasingly evident that employing more

sophisticated optimization techniques do not necessarily ensure improved performance — see, e.g., Bousquet and Bottou (2008) and Bottou (2012). This is because the assumed data models or risk functions do not always capture faithfully the underlying data structure anyway. In addition, the presence of noise in the data generally implies that a solution that may be perceived to be optimal is actually sub-optimal due to the perturbations in the data and models.

Adaline and Perceptron. During the 1950s, stochastic approximation theory did not receive much attention in the engineering community until the landmark work by Widrow and Hoff (1960) in which they developed the delta or adaline recursion (16.32). Using a target sequence $\boldsymbol{\gamma}(n)$, the algorithm enabled the adjusting of the weight parameter $\boldsymbol{w}_n$ in order to close the gap between the target signal and its prediction given by $\widehat{\boldsymbol{\gamma}}(n) = \boldsymbol{h}_n^{\mathsf{T}}\boldsymbol{w}_{n-1}$. Their filter launched the design of adaptive systems with adjustable structures and has found applications in a remarkable range of areas — see, e.g., the treatments by Widrow and Stearns (1985), Haykin (2001), and Sayed (2003,2008). A useful interpretation for the LMS algorithm (or delta rule) as the solution to a min-max optimization problem was given by Hassibi, Sayed, and Kailath (1994a, 1996). The algorithm was further studied by Sayed and Rupp (1996) and Sayed (2003,2008) using energy arguments to establish several robustness properties.

Besides adaline and LMS, there have been other notable works on stochastic gradient algorithms in the early 1960s. One example is the Perceptron algorithm (16.75), which was developed by Rosenblatt (1957,1958,1962) for pattern classification problems and which we will study in greater detail in future Chapter 60.

Early examples of stochastic approximation structures. There are other examples of adjustable designs from the 1950s that bear resemblance to stochastic approximation constructions. One such contribution are the works by Mattson (1959a,b) in the context of pattern classifiers. According to Widrow and Hoff (1960, p. 97), these works were among the first to apply adjustable structures to classification problems. However, unlike Rosenblatt (1957), the construction proposed in Mattson (1959b) was a trial-and-error procedure based on varying the weight entries and a threshold value until satisfactory performance is attained. There was no explicit optimization problem guiding the design procedure. This is reflected in the description by Mattson (1959b), where it is stated that *"it is the purpose of this paper to define a model for a self-organizing logical system which determines, by an iterative trial-and-error procedure, the proper Boolean function for a process."*

According to the presentation in Sayed (2003,2008), another example of early work on stochastic algorithms is a procedure that minimizes the mean-square error between an input signal and a reference signal developed by Gabor, Wilby and Woodcock (1961); their filter is described in Tsypkin (1971, p. 156). This latter reference also contains on pages 172–173 commentaries on works on adaptation and learning during the early sixties, including a description of a stochastic gradient algorithm by Sefl (1960) that is the continuous-time counterpart of the delta or LMS rule; it employs a differential update equation of the form

$$\frac{d\boldsymbol{w}(t)}{dt} = 2\mu(t)\boldsymbol{h}(t)\left(\boldsymbol{\gamma}(t) - (\boldsymbol{h}(t))^{\mathsf{T}}\,\boldsymbol{w}(t)\right) \tag{16.136}$$

with continuous-time vector variables $\{\boldsymbol{w}(t), \boldsymbol{h}(t)\}$ and scalar variables $\{\boldsymbol{\gamma}(t), \mu(t)\}$. Other noteworthy works on stochastic gradient algorithms in the 1960s are those by Applebaum (1966) and Widrow *et al.* (1967) on adaptive antenna arrays and Amari (1967) on pattern classification. In Applebaum (1966), a stochastic gradient algorithm is derived that is based on maximizing a signal-to-noise ratio measure, while Widrow *et al.* (1967) focus on mean-square error performance and use the LMS algorithm. The work by Amari (1967) uses a stochastic gradient recursion to learn the weight vector in a pattern classification problem.

For further readings and discussion on online learning techniques, the reader may

consult the books by Sayed (2003,2008, 2014a), Cesa-Bianchi and Lugosi (2006), Shalev-Shwartz (2011), and Theodoridis (2015), and the articles by Bottou (1998,2012), Cesa-Bianchi, Conconi, and Gentile (2004), Bottou and Bousquet (2008), Bach and Moulines (2011), and Agarwal *et al.* (2012).

Zeroth-order learning algorithms. We can also develop stochastic versions for the zeroth-order optimization algorithms described earlier in Appendix 12.A. In this case, we would sample directional vectors u in order to approximate the gradient vector of the loss (rather than the risk) function by using either instantaneous or mini-batch calculations. For instance, the stochastic gradient algorithm (16.27) would be replaced by listing (16.137) for instantaneous gradient approximations or by listing (16.138) for mini-batch approximations. In either listing, we denote the distribution from which the directional vectors $\boldsymbol{u}$ are sampled by $f_{\boldsymbol{u}}(u)$; it can refer to either the Gaussian distribution $\mathcal{N}_{\boldsymbol{u}}(0, I_M)$ or the uniform distribution $\mathcal{U}(\mathbb{S})$, as described by (12.213a)–(12.213b).

Zeroth-order stochastic gradient algorithm for minimizing (16.2a) or (16.2b)

given dataset $\{\gamma(m), h_m\}_{m=0}^{N-1}$ or streaming data $(\gamma(n), h_n)$;
given a small smoothing factor $\alpha > 0$;
select the sampling distribution $f_{\boldsymbol{u}}(u)$ and set $\beta \in \{1, M\}$;
start from an arbitrary initial condition, $\boldsymbol{w}_{-1}$.
repeat until sufficient convergence over $n \geq 0$:

$$\begin{aligned}
&\text{select at random or receive a sample } (\boldsymbol{\gamma}(n), \boldsymbol{h}_n) \text{ at iteration } n;\\
&\text{sample } \boldsymbol{u} \sim f_{\boldsymbol{u}}(u);\\
&\widehat{\nabla_{w^{\mathsf{T}}} Q}(\boldsymbol{w}_{n-1}; \boldsymbol{u}) = \frac{\beta}{\alpha}\left\{ Q\Big(\boldsymbol{w}_{n-1} + \alpha \boldsymbol{u}; \boldsymbol{\gamma}(n), \boldsymbol{h}_n\Big) - Q\Big(\boldsymbol{w}_{n-1}; \boldsymbol{\gamma}(n), \boldsymbol{h}_n\Big) \right\} \boldsymbol{u}\\
&\boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu\, \widehat{\nabla_{w^{\mathsf{T}}} Q}(\boldsymbol{w}_{n-1}; \boldsymbol{u})
\end{aligned}$$

end
return $w^{\star} \leftarrow \boldsymbol{w}_n$.

(16.137)

Zeroth-order mini-batch stochastic algorithm for minimizing (16.2a) or (16.2b)

given dataset $\{\gamma(m), h_m\}_{m=0}^{N-1}$ or streaming data $(\gamma(n), h_n)$;
given a mini-batch size, B;
given a small smoothing factor $\alpha > 0$;
select the sampling distribution $f_{\boldsymbol{u}}(u)$ and set $\beta \in \{1, M\}$;
start from an arbitrary initial condition, $\boldsymbol{w}_{-1}$.
repeat until sufficient convergence over $n \geq 0$:

$$\begin{aligned}
&\text{select at random or receive } B \text{ samples } \{\boldsymbol{\gamma}(b), \boldsymbol{h}_b\}_{b=0}^{B-1} \text{ at iteration } n;\\
&\text{sample direction } \boldsymbol{u} \sim f_{\boldsymbol{u}}(u)\\
&\widehat{\nabla_{w^{\mathsf{T}}} Q}(\boldsymbol{w}_{n-1}; \boldsymbol{u}) \;\triangleq\; \frac{\beta}{\alpha}\Big(Q(\boldsymbol{w}_{n-1} + \alpha \boldsymbol{u}; \boldsymbol{\gamma}(b), \boldsymbol{h}_b) - Q(\boldsymbol{w}_{n-1}; \boldsymbol{\gamma}(b), \boldsymbol{h}_b) \Big) \boldsymbol{u}\\
&\boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu \left(\frac{1}{B} \sum_{b=0}^{B-1} \widehat{\nabla_{w^{\mathsf{T}}} Q}(\boldsymbol{w}_{n-1}; \boldsymbol{u}) \right)
\end{aligned}$$

end
return $w^{\star} \leftarrow \boldsymbol{w}_n$.

(16.138)

There are many variations by which the gradient vector of the loss function can be ap-

proximated. The mini-batch listing assume that, for each iteration n, a single direction u is sampled and used for all entries in the minibatch. Alternatively, we could consider sampling J directions u_j, computing the minibatch gradient approximation for each one of them, and then averaging over the J directions, namely,

$$\left\{\begin{array}{l}
\text{sample } J \text{ directions } \boldsymbol{u}_j \sim f_{\boldsymbol{u}}(u),\ j=0,1,\ldots,J-1;\\
\textbf{for each } j\textbf{, compute } B \textbf{ gradients:}\\
\quad \left|\begin{array}{l}
\widehat{\nabla_{w^{\mathsf{T}}}Q}(\boldsymbol{w}_{n-1};\boldsymbol{u}_j,b)=\dfrac{\beta}{\alpha}\left\{Q\Big(\boldsymbol{w}_{n-1}+\alpha\boldsymbol{u}_j;\boldsymbol{\gamma}(b),\boldsymbol{h}_b\Big)-Q\Big(\boldsymbol{w}_{n-1};\boldsymbol{\gamma}(b),\boldsymbol{h}_b\Big)\right\}\boldsymbol{u}_j\\
b=0,1,\ldots,B-1
\end{array}\right.\\
\textbf{end}\\
\text{set } \widehat{\nabla_{w^{\mathsf{T}}}Q}(\boldsymbol{w}_{n-1})=\dfrac{1}{J}\displaystyle\sum_{j=0}^{J-1}\left\{\frac{1}{B}\sum_{b=0}^{B-1}\widehat{\nabla_w Q}(\boldsymbol{w}_{n-1};\boldsymbol{u}_j,b)\right\};\\
\text{update } \boldsymbol{w}_n=\boldsymbol{w}_{n-1}-\mu\widehat{\nabla_{w^{\mathsf{T}}}Q}(\boldsymbol{w}_{n-1}).
\end{array}\right. \tag{16.139}$$

Other constructions are possible.

Collaborative filtering. The stochastic gradient implementation (16.58) for solving the matrix factorization problem (16.57) is motivated by the works of Funk (2006), Paterek (2007), Takacs *et al.* (2007), and Koren (2008). The alternating least-squares version described later in future Example 50.6 is motivated by Bell and Koren (2007a), Hu, Koren, and Volinsky (2008), Zhou *et al.* (2008), and Pilaszy, Zibriczky, and Tikk (2010). Most of these works were driven by the Netflix prize challenge, which was an open competition during the period 2006-2009 offering a prize of 1 million US Dollars for the best collaborative filtering solution to predict user ratings of movies — overviews appear in Bell, Koren, and Volinsky (2007) and Bell and Koren (2007b). Netflix provided training data consisting of over 100 million ratings from close to 500 thousand users for about 18 thousand movies. For more details on the Netflix challenge and on matrix factorization methods, readers may consult the tutorial by Koren, Bell, and Volinsky (2009) and the text by Symeonidis and Zioupos (2017). The stochastic gradient algorithm (16.58) was also among the top solutions in the KDDCup 2011 challenge, which dealt with the problem of recommending music items to users from the Yahoo music dataset. This challenge released about 250 million ratings from over 1 million anonymized users — see the account by Dror *et al.* (2012).

PROBLEMS

16.1 Consider an empirical risk minimization problem with uniform data sampling. Given a finite number of data samples $\{(\gamma(0),h_0),(\gamma(1),h_1),\ldots,(\gamma(N-1),h_{N-1})\}$, we define discrete random variables $\{\boldsymbol{\gamma},\boldsymbol{h}\}$ that are generated according to the following probability distribution:

$$\mathbb{P}(\boldsymbol{\gamma}=\gamma,\boldsymbol{h}=h)=\left\{\begin{array}{ll}
1/N, & \text{if } \gamma=\gamma(0),h=h_0\\
1/N, & \text{if } \gamma=\gamma(1),h=h_1\\
\vdots & \vdots\\
1/N, & \text{if } \gamma=\gamma(N-1),h=h_{N-1}
\end{array}\right.$$

Verify that, under this construction, it holds:

$$\mathbb{E}\,Q(w;\boldsymbol{\gamma},\boldsymbol{h}) \;=\; \frac{1}{N}\sum_{m=0}^{N-1} Q(w;\gamma(m),h_m)$$

Conclude that the solutions to the following stochastic and empirical risk problems coincide:

$$w^o \;\triangleq\; \operatorname*{argmin}_{w\in\mathbb{R}^M} \mathbb{E}\,Q(w;\boldsymbol{\gamma},\boldsymbol{h}) \;=\; \operatorname*{argmin}_{w\in\mathbb{R}^M} \frac{1}{N}\sum_{m=0}^{N-1} Q(w;\gamma(m),h_m) \;\triangleq\; w^\star$$

16.2 Refer to the empirical risk minimization problem (16.2a) with minimizer denoted by $w^\star$. Assume the minimization is restricted over a bounded convex set, $w\in\mathcal{C}$. Refer further to the rightmost term in (16.113) in the definition of the average regret. Assuming a stochastic gradient implementation, how does the minimizer of this rightmost term compare to $w^\star$?

16.3 Consider the ℓ_1-regularized mean-square-error risk:

$$w^o = \operatorname*{argmin}_{w\in\mathbb{R}^M} \;\Big\{\alpha\|w\|_1 \;+\; \mathbb{E}\,(\boldsymbol{\gamma}-\boldsymbol{h}^{\mathsf{T}}w)^2\Big\}$$

(a) Write down a stochastic subgradient algorithm for its solution.
(b) Write down a stochastic proximal algorithm for the same problem.
(c) Write down a stochastic coordinate descent solution.

16.4 The total-variation denoising problem involves solving a regularized least-squares problem of the following form:

$$\min_w \left\{\alpha\sum_{m'=1}^{M-1}\Big|w[m'+1]-w[m']\Big| \;+\; \frac{1}{N}\sum_{m=0}^{N-1}\Big(\gamma(m)-h_m^{\mathsf{T}}w\Big)^2\right\}$$

where the $\{w[m']\}$, for $m'=1,2,\ldots,M$, denote the individual entries of $w\in\mathbb{R}^M$.
(a) Derive a stochastic subgradient solution for this problem.
(b) Derive a stochastic proximal solution for the same problem.

16.5 The fused LASSO problem adds ℓ_1-regularization to the total variation formulation in Prob. 16.4 and considers instead

$$\min_w \left\{\alpha_1\|w\|_1+\alpha_2\sum_{m'=1}^{M-1}\Big|w[m'+1]-w[m']\Big| \;+\; \frac{1}{N}\sum_{m=0}^{N-1}\Big(\gamma(m)-h_m^{\mathsf{T}}w\Big)^2\right\}$$

(a) Derive a stochastic subgradient solution for this problem.
(b) Derive a stochastic proximal solution for the same problem.

16.6 The group LASSO problem involves solving a regularized least-squares problem of the following form. We partition each observation vector into K sub-vectors, say, $h_m=\mathrm{col}\{h_{mk}\}$ for $k=1,2,\ldots,K$. We similarly partition the weight vector into K sub-vectors, $w=\mathrm{col}\{w_k\}$, of similar dimensions to h_{mk}. Now consider the problem:

$$\min_w \left\{\alpha\sum_{k=1}^{K}\|w_k\| \;+\; \frac{1}{N}\sum_{m=0}^{N-1}\left(\gamma(m)-\sum_{k=1}^{K}h_{mk}^{\mathsf{T}}w_k\right)^2\right\}$$

(a) Derive a stochastic subgradient solution for this problem.
(b) Derive a stochastic proximal solution for the same problem.

16.7 Consider a collection of $N-$data pairs $\{\gamma(m),h_m\}$ where $\gamma(m)\in\mathbb{R}$ and $h_m\in\mathbb{R}^M$, and formulate the least-squares problem:

$$w^\star = \operatorname*{argmax}_{w\in\mathbb{R}^M} \left\{\frac{1}{N}\sum_{m=0}^{N-1}\Big(\gamma(m)-h_m^{\mathsf{T}}w\Big)^2\right\}$$

One stochastic gradient method for minimizing this risk function can be devised as follows (this method is known as a *randomized* Kaczmarz method):

$$\begin{cases} \text{select an index } 0 \le n \le N-1 \text{ at random} \\ \text{consider the corresponding data pair } \{\boldsymbol{\gamma}(n), \boldsymbol{h}_n\} \\ \text{update } \boldsymbol{w}_n = \boldsymbol{w}_{n-1} + \dfrac{\boldsymbol{h}_n}{\|\boldsymbol{h}_n\|^2}(\boldsymbol{\gamma}(n) - \boldsymbol{h}_n^{\mathsf{T}}\boldsymbol{w}_{n-1}) \end{cases}$$

Collect all vectors $\{h_m\}$ into the $N \times M$ matrix $H = \mathrm{row}\{h_m^{\mathsf{T}}\}$ and all target values into the $N \times 1$ vector $d = \mathrm{col}\{\gamma(m)\}$. Assume H has full rank. Assume also the random index $\boldsymbol{n}$ is selected according to the following importance sampling procedure $\mathbb{P}(\boldsymbol{n} = m) = \|h_m\|^2/\|H\|_{\mathrm{F}}^2$. Note that the vectors $\boldsymbol{h}_n$ are selected independently of each other and of any other random variable in the problem. Let $w^\star$ denote the solution to the least-squares problem, i.e., it satisfies $H^{\mathsf{T}}Hw^\star = H^{\mathsf{T}}d$, and introduce the weight-error vector $\widetilde{\boldsymbol{w}}_n = w^\star - \boldsymbol{w}_n$. Show that

$$\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_n\|^2 \le \left(1 - \frac{\sigma_{\min}^2(H)}{\|H\|_{\mathrm{F}}^2}\right)\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_{n-1}\|^2, \quad n \ge 0$$

in terms of the smallest singular value of H. *Remark.* See the work by Strohmer and Vershynin (2009), who studied this randomized version of a popular method by Kaczmarz (1937) for the solution of linear systems of equations. The traditional Kaczmarz method studied earlier in Prob. 12.34 cycles through the rows of H, whereas the randomized version described here samples the rows of H at random as described above.

16.8 Continuing with Prob. 16.7, let $w^\star$ denote the minimum-norm solution in the over-parameterized case when $N < M$. Show that $\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_n\|^2$ converges to zero.

16.9 Let $\{\boldsymbol{x}(n),\ n = 1, \ldots, N\}$ denote N independent realizations with mean μ and and finite variance, $\sigma_x^2 = \mathbb{E}\,(\boldsymbol{x}(n) - \mu)^2 < \infty$. Introduce the sample average $\widehat{\boldsymbol{\mu}}_N = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}(n)$.

(a) Verify that $\widehat{\boldsymbol{\mu}}_N = \widehat{\boldsymbol{\mu}}_{N-1} + \frac{1}{N}\left(\boldsymbol{x}(N) - \widehat{\boldsymbol{\mu}}_{N-1}\right)$ with $\widehat{\boldsymbol{\mu}}_0 = 0$ and $N \ge 1$.

(b) Assume that the sample average is approximated recursively as follows (written now as $\boldsymbol{\mu}_N$ to distinguish it from $\widehat{\boldsymbol{\mu}}_N$):

$$\boldsymbol{\mu}_N = \boldsymbol{\mu}_{N-1} + \alpha(N)\left(\boldsymbol{x}(N) - \boldsymbol{\mu}_{N-1}\right), \quad \boldsymbol{\mu}_0 = 0,\ N \ge 1$$

where the scalar sequence $\{\alpha(n)\}$ satisfies

$$0 \le \alpha(N) < 1, \quad \lim_{N\to\infty}\alpha(N) = 0, \quad \lim_{N\to\infty}\sum_{n=1}^{N}\alpha(n) = \infty$$

We want to verify that $\boldsymbol{\mu}_N$ tends to μ in probability. Let σ_N^2 denote the variance of $\boldsymbol{\mu}_N$, i.e., $\sigma_N^2 = \mathbb{E}\,(\boldsymbol{\mu}_N - \mathbb{E}\,\boldsymbol{\mu}_N)^2$.

(b.1) Verify that $(\mathbb{E}\,\boldsymbol{\mu}_N - \mu) = (1 - \alpha(N))\,(\mathbb{E}\,\boldsymbol{\mu}_{N-1} - \mu),\ N \ge 1$.

(b.2) Show that σ_N^2 satisfies $\sigma_N^2 = (1-\alpha(N))^2\sigma_{N-1}^2 + \alpha^2(N)\sigma_x^2$.

(b.3) Compare the recursion in (b.2) with (14.136) and conclude that $\sigma_N^2 \to 0$ as $N \to \infty$. Conclude also that $\mathbb{E}\,\boldsymbol{\mu}_N \to \mu$ as $N \to \infty$.

16.10 Consider the regularized logistic risk

$$P(w) = \rho\|w\|^2 + \mathbb{E}\left\{\ln\left(1 + e^{-\boldsymbol{\gamma}\boldsymbol{h}^{\mathsf{T}}w}\right)\right\}$$

where $\boldsymbol{\gamma}$ is a binary random variable assuming the values ± 1 and $R_h = \mathbb{E}\,\boldsymbol{h}\boldsymbol{h}^{\mathsf{T}}$. Let w^o denote the minimizer of $P(w)$. Show that

(a) $\|w^o\| \le \mathbb{E}\,\|\boldsymbol{h}\|/2\rho$.

(b) $\|w^o\|^2 \le \mathrm{Tr}(R_h)/4\rho^2$.

16.11 Consider the mean-square-error cost $P(w) = \mathbb{E}\,(\boldsymbol{\gamma}(n) - \boldsymbol{h}_n^{\mathsf{T}} w)^2$, where $\boldsymbol{\gamma}(n)$ denotes a streaming sequence of zero-mean random variables with variance $\sigma_\gamma^2 = \mathbb{E}\,\boldsymbol{\gamma}^2(n)$ and $\boldsymbol{h}_n \in \mathbb{R}^M$ is a streaming sequence of independent zero-mean Gaussian random vectors with covariance matrix $R_h = \mathbb{E}\,\boldsymbol{h}_n \boldsymbol{h}_n^{\mathsf{T}} > 0$. Both processes $\{\boldsymbol{\gamma}(n), \boldsymbol{h}_n\}$ are assumed to be jointly wide-sense stationary. The cross-covariance vector between $\boldsymbol{\gamma}(n)$ and $\boldsymbol{h}_n$ is denoted by $r_{h\gamma} = \mathbb{E}\,\boldsymbol{\gamma}(n)\boldsymbol{h}_n$. The data $\{\boldsymbol{\gamma}(n), \boldsymbol{h}_n\}$ are assumed to be related via a linear regression model of the form $\boldsymbol{\gamma}(n) = \boldsymbol{h}_n^{\mathsf{T}} w^\bullet + \boldsymbol{v}(n)$, for some unknown parameter vector $w^\bullet$, and where $\boldsymbol{v}(n)$ is a zero-mean white-noise process with power $\sigma_v^2 = \mathbb{E}\,\boldsymbol{v}^2(n)$ and assumed independent of $\boldsymbol{h}_m$ for all n, m.

(a) Let w^o denote the minimizer for $P(w)$. Show that $w^o = w^\bullet$.

(b) Consider the stochastic gradient algorithm for estimating w^o from streaming data, $\boldsymbol{w}_n = \boldsymbol{w}_{n-1} + 2\mu \boldsymbol{h}_n(\boldsymbol{\gamma}(n) - \boldsymbol{h}_n^{\mathsf{T}} \boldsymbol{w}_{n-1})$. Let $\widetilde{\boldsymbol{w}}_n = w^o - \boldsymbol{w}_n$. Verify that

$$\widetilde{\boldsymbol{w}}_n = (1 - 2\mu \boldsymbol{h}_n \boldsymbol{h}_n^{\mathsf{T}})\widetilde{\boldsymbol{w}}_{n-1} + 2\mu \boldsymbol{h}_n \boldsymbol{v}(n)$$

(c) Determine a necessary and sufficient condition on μ to ensure convergence in the mean, i.e., for $\mathbb{E}\,\widetilde{\boldsymbol{w}}_n \to 0$ as $n \to \infty$.

(d) Determine a recursion for $\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_n\|^2$.

(e) Find a necessary and sufficient condition on μ to ensure that $\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_n\|^2$ converges. How does this condition compare to the one in part (c)?

(f) Find an expression for the limiting value of $\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_n\|^2$ (also referred to as the mean-square-deviation (MSD) of the algorithm).

16.12 Consider the same setting of Prob. 16.11 albeit with $R_h = \sigma_h^2 I_M$. Assume the limits exist and define the mean-square-deviation (MSD) and excess mean-square-error (EMSE) figures of merit:

$$\text{MSD} = \lim_{n\to\infty} \mathbb{E}\,\|\widetilde{\boldsymbol{w}}_n\|^2, \qquad \text{EMSE} = \lim_{n\to\infty} \mathbb{E}\,|\boldsymbol{h}_n^{\mathsf{T}} \widetilde{\boldsymbol{w}}_{n-1}|^2$$

(a) Verify that $P(w^o) = \sigma_v^2$ and $R(w_{n-1}) = R(w^o) + \text{EMSE}$. Justify the name "excess mean-square-error."

(b) Determine expressions for the MSD and EMSE.

(c) Define the convergence time, $\mathcal{K}$, as the number of iterations it takes for the mean-square-error, $P(w_{n-1})$, to be within $\epsilon\%$ of its steady-state value. Find a closed form expression for $\mathcal{K}$.

16.13 Consider the regret bound (16.123) and assume $\mu(n) = 1/(n+1)$. At what rate does the regret approach its limiting behavior?

16.14 Consider a stochastic gradient recursion of the form:

$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} + \boldsymbol{\mu}(n)\boldsymbol{h}_n \left(\boldsymbol{\gamma}(n) - \boldsymbol{h}_n^{\mathsf{T}} \boldsymbol{w}_{n-1}\right)$$

where $\boldsymbol{\gamma}(n) \in \mathbb{R}$ and $\boldsymbol{h}_n \in \mathbb{R}^M$. The step-size $\boldsymbol{\mu}(n)$ is an independent and identically distributed (i.i.d.) random process with mean $\bar{\mu}$ and variance σ_μ^2. The feature vectors $\{\boldsymbol{h}_n\}$ are i.i.d. Gaussian with zero mean and covariance matrix $R_h = \sigma_h^2 I_M > 0$. Moreover, the data $\{\boldsymbol{\gamma}(n), \boldsymbol{h}_n\}$ is assumed to arise from the stationary data model $\boldsymbol{\gamma}(n) = \boldsymbol{h}_n^{\mathsf{T}} w^o + \boldsymbol{v}(n)$, where $\boldsymbol{h}_n$ and $\boldsymbol{v}(m)$ are independent of each other for all n and m. The variance of the zero-mean process $\boldsymbol{v}(n)$ is denoted by σ_v^2. In addition, the step-size variable $\boldsymbol{\mu}(n)$ is assumed to be independent of all random variables in the learning algorithm for any time instant.

(a) Determine conditions to ensure mean convergence of $\boldsymbol{w}_n$ towards w^o.

(b) Determine a recursion for $\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_n\|^2$, where $\widetilde{\boldsymbol{w}}_n = w^o - \boldsymbol{w}_n$.

(c) Determine conditions to ensure the convergence of $\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_n\|^2$ to a steady-state value.

(d) Use the recursion of part (b) to determine an exact closed-form expression for the limiting mean-square-deviation (MSD) of the algorithm, which is defined as the limiting value of $\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_n\|^2$ as $n \to \infty$.

(e) Determine an approximation for the MSD metric to first-order in $\bar{\mu}$.

(f) Determine an approximation for the convergence rate to first-order in $\bar{\mu}$.

(g) Assume $\boldsymbol{\mu}(n)$ is Bernoulli and assumes the values μ and 0 with probabilities p and $1-p$, respectively. What are the values of $\bar{\mu}$ and σ_μ^2 in this case? Consider the alternative stochastic gradient implementation with $\boldsymbol{\mu}(n)$ replaced by a constant value μ. How do the MSD values for these two implementations, with $\boldsymbol{\mu}(n)$ and μ, compare to each other?

16.15 Consider the stochastic mirror descent algorithm and apply it to the solution of the following optimization problem:

$$w^\star = \operatorname*{argmin}_{w \in \mathcal{C} \cap \mathcal{C}_\phi} \frac{1}{N} \sum_{n=0}^{N-1} Q_n(w)$$

Here, each loss term $Q_n(w) : \mathbb{R}^M \to \mathbb{R}$ is convex over $\mathcal{C}$ and $\delta-$Lipschitz relative to some norm $\|\cdot\|$. Observe that in this case the objective function is the average of several loss values changing with n. Consider a mirror function $\phi(w) : \mathcal{C}_\phi \to \mathbb{R}$ that is $\nu-$strongly convex relative to the same norm, and where $\mathcal{C} \subset \bar{\mathcal{C}}_\phi$. Repeat the argument used in the proof of Theorem 15.5 to show that

$$0 \le \frac{1}{N} \sum_{n=0}^{N-1} \Big(Q_n(w_{n-1}) - Q_n(w^\star) \Big) \le \frac{1}{N\mu} D_\phi(w^\star, w_{-1}) + \frac{\mu}{2N\nu} \sum_{n=0}^{N-1} \|g_n\|_\star^2$$

where g_n is a subgradient for $Q_n(w)$ at w_{n-1} and $\|\cdot\|_\star$ is the dual norm.

16.16 Continuing with the setting of Prob. 16.15, assume $\|\cdot\|$ is the ℓ_1-norm and $\delta = 1$. Choose $\phi(x)$ as the negative entropy function for which we already know from the discussion in the body of the chapter that $\nu = 1$. Choose $w_{-1} = \frac{1}{M}\mathbb{1}$ and $\mu = \sqrt{(2/N)\ln M}$. Conclude that

$$0 \le \frac{1}{N} \sum_{n=0}^{N-1} \Big(Q_n(w_{n-1}) - Q_n(w^\star) \Big) \le \sqrt{\frac{2 \ln M}{N}}$$

16.A SWITCHING EXPECTATION AND DIFFERENTIATION

We encountered in the body of the chapter instances where it is necessary to switch the order of the expectation and differentiation operators. The switching can be justified by appealing to the *dominated convergence theorem* from measure theory, which we state under conditions that are sufficient for our purposes.

Dominated convergence theorem (e.g., Rudin (1976), Royden (1988)). *Assume $R_n(x)$ is a sequence of real-valued functions parameterized by n and converging pointwise to a limit as $n \to \infty$. Assume that $R_n(x)$ is dominated by another function independent of n, i.e., $|R_n(x)| \le a(x)$ for all $x \in \mathcal{D}$ in the domain of $R_n(x)$ and where $a(x)$ is integrable, i.e., $\int_{\mathcal{D}} a(x)dx < \infty$. Then, it holds that*

$$\lim_{n\to\infty} \left(\int_{\mathcal{D}} R_n(x) dx \right) = \int_{\mathcal{D}} \left(\lim_{n\to\infty} R_n(x) \right) dx \tag{16.140}$$

That is, we can switch the limit and integral signs.

We can use the above result to justify exchanging derivatives (which are limit operations) with expectations (which are integral operations). We provide three statement variations that lead to similar conclusions under related but different conditions. In

the proofs, we follow the same line of reasoning that is used to establish the classical Leibniz integral rule from calculus for the derivative of an integral expression by means of the dominated convergence theorem — see, e.g., Natanson (1961), Buck (1965), Hewitt and Stromberg (1969), Dieudonné (1969), Apostol (1974), Lewin (1987), Bartle, (1995), Troutman (1996), or Norris (2013).

Theorem 16.1. (Switching expectation and gradient operations I) *Consider a function $Q(w;\boldsymbol{x}) : \mathbb{R}^M \times \mathbb{R}^P \to \mathbb{R}$, where $\boldsymbol{x}$ is a random vector. Assume Q is first-order differentiable with respect to w, and that for any $w \in \operatorname{dom} Q(w;x)$ there exists a function $b(w;x) : \mathbb{R}^M \times \mathbb{R}^P \to [0,\infty)$ satisfying $\mathbb{E}\, b(w;\boldsymbol{x}) < \infty$ and*

$$\|\nabla_w Q(w+\delta w;x)\| \;\le\; b(w;x), \quad \text{for any } \|\delta w\| \le \epsilon \tag{16.141}$$

Then, assuming the expectations over the distribution of $\boldsymbol{x}$ exist, it holds that

$$\nabla_w \Big(\mathbb{E}\, Q(w;\boldsymbol{x})\Big) \;=\; \mathbb{E}\,\Big(\nabla_w Q(w;\boldsymbol{x})\Big) \tag{16.142}$$

Proof: The argument is motivated by the discussion in Hewitt and Stromberg (1969, pp.172–173), Bartle (1995, Corollary 5.7), and also by the derivation used in the proof of Theorem 3.5.1 from Norris (2013) extended to vector variables. Let $w[m]$ denote the $m-$th entry of $w \in \mathbb{R}^M$. Then, from the definition of the gradient vector of a multi-variable function we know that:

$$\nabla_{w^{\mathsf{T}}} \mathbb{E}\, Q(w;\boldsymbol{x}) \;\triangleq\; \operatorname{col}\left\{\frac{\partial \mathbb{E}\, Q(w;\boldsymbol{x})}{\partial w[1]},\; \frac{\partial \mathbb{E}\, Q(w;\boldsymbol{x})}{\partial w[2]},\; \ldots,\; \frac{\partial \mathbb{E}\, Q(w;\boldsymbol{x})}{\partial w[M]}\right\} \tag{16.143}$$

We establish result (16.142) by considering the individual entries of the gradient vector. Let $\alpha > 0$ denote a small positive scalar and let e_m denote the $m-$th basis vector in $\mathbb{R}^M$ with all its entries equal to zero except for the $m-$th entry, which is equal to one. According to the definition of the differentiation operation, we have

$$\begin{aligned}
\frac{\partial \mathbb{E}\, Q(w;\boldsymbol{x})}{\partial w[m]} &= \lim_{\alpha\to 0} \frac{1}{\alpha}\Big(\mathbb{E}\, Q(w+\alpha e_m;\boldsymbol{x}) \;-\; \mathbb{E}\, Q(w;\boldsymbol{x})\Big) \\
&= \lim_{\alpha\to 0} \frac{1}{\alpha}\mathbb{E}\,\Big(Q(w+\alpha e_m;\boldsymbol{x}) \;-\; Q(w;\boldsymbol{x})\Big) \\
&\overset{(a)}{=} \lim_{\alpha\to 0} \frac{1}{\alpha}\,\mathbb{E}\,\Big(\nabla_w Q(w+t_m\alpha e_m;\boldsymbol{x})\Big)^{\mathsf{T}} \alpha e_m \\
&\overset{(b)}{=} \lim_{\alpha\to 0}\; \mathbb{E}\,\left(\frac{\partial Q(w+t_m\alpha e_m;\boldsymbol{x})}{\partial w[m]}\right)
\end{aligned} \tag{16.144}$$

for some constant $t_m \in (0,1)$. Equality (a) holds because of the mean-value theorem, while equality (b) holds because the elements in e_m are all zero except for the $m-$th entry. We next introduce the function

$$g_m(w;x) \;\triangleq\; \frac{\partial Q(w;x)}{\partial w[m]} \tag{16.145}$$

so that (16.144) becomes

$$\frac{\partial \mathbb{E}\, Q(w;\boldsymbol{x})}{\partial w[m]} \;=\; \lim_{\alpha\to 0}\; \mathbb{E}\, g_m(w+t_m\alpha e_m;\boldsymbol{x}) \;=\; \lim_{\alpha\to 0}\left(\int_{x\in\mathcal{D}} g_m(w+t_m\alpha e_m;\boldsymbol{x}) f_{\boldsymbol{x}}(x)dx\right) \tag{16.146}$$

where $\mathcal{D}$ is the domain of $\boldsymbol{x}$. Now consider any scalar sequence $\alpha^{(n)} \to 0$. We define $\alpha^{(n)} e_m \in \mathbb{R}^M$ as the vector in which all the elements are zero except for the $m-$entry

set to $\alpha^{(n)}$. With this notation, expression (16.144) becomes

$$\frac{\partial \mathbb{E}\, Q(w;\boldsymbol{x})}{\partial w[m]} \;=\; \lim_{n\to+\infty}\left(\int_{x\in\mathcal{D}} g_m\Big(w+t_m\alpha^{(n)}e_m;\boldsymbol{x}\Big)f_{\boldsymbol{x}}(x)dx\right) \tag{16.147}$$

Next, we define

$$R_n(x;w) \;\triangleq\; g_m\Big(w+t_m\alpha^{(n)}e_m;x\Big)f_{\boldsymbol{x}}(x), \quad w\in\mathrm{dom}(Q) \tag{16.148}$$

In $R_n(x;w)$, the symbol x is the variable and w is a parameter. The function $R_n(x;w)$ is dominated. Indeed, using condition (16.141) we have

$$\begin{aligned}\Big|g_m\Big(w+t_m\alpha^{(n)}e_m;x\Big)\Big| &= \left|\frac{\partial Q\Big(w+t_m\alpha^{(n)}e_m;x\Big)}{\partial w[m]}\right| \\ &\le \Big\|\nabla_w\, Q\Big(w+t_m\alpha^{(n)}e_m;x\Big)\Big\| \\ &\le b(w;x)\end{aligned} \tag{16.149}$$

for any $n\ge N_o$ where N_o is sufficiently large. Therefore, for any $x\in\mathcal{D}$ and $w\in\mathrm{dom}(Q)$, it holds that

$$|R_n(x;w)| \;\le\; b(w;x)f_{\boldsymbol{x}}(x) \tag{16.150}$$

Since, by assumption, $\mathbb{E}\, b(w;\boldsymbol{x})<+\infty$, we know that $b(w;x)f_{\boldsymbol{x}}(x)$ is integrable. Finally, applying the dominated convergence theorem, we know that for any $w\in\mathrm{dom}\,(Q)$, it holds that

$$\begin{aligned}\lim_{n\to+\infty}\int_{x\in\mathcal{D}} R_n(x;w)dx &= \int_{x\in\mathcal{D}}\lim_{n\to+\infty} R_n(x;w)dx \\ &= \int_{x\in\mathcal{D}}\frac{\partial Q(w;x)}{\partial w[m]}f_{\boldsymbol{x}}(x)dx \\ &= \mathbb{E}\left(\frac{\partial Q(w;\boldsymbol{x})}{\partial w[m]}\right)\end{aligned} \tag{16.151}$$

which also implies from (16.147) that

$$\frac{\partial \mathbb{E}\, Q(w;\boldsymbol{x})}{\partial w[m]} \;=\; \mathbb{E}\left(\frac{\partial Q(w;\boldsymbol{x})}{\partial w[m]}\right) \tag{16.152}$$

From (16.143) and (16.152) we arrive at (16.142).

■

We state a second related variation of the theorem, which holds when $\nabla_w Q(w;\boldsymbol{x})$ is continuous in w and the distribution of $\boldsymbol{x}$ is such that the means of the absolute entries of $\nabla_w Q(w;\boldsymbol{x})$ are bounded — see, e.g., the statement for the dominated convergence theorem for bounded functions in Natanson (1961), Luxemburg (1971), Lewin (1987), and the brief note by Ene (1999). We continue with the same notation used in the proof of the previous version.

Theorem 16.2. (Switching expectation and gradient operations II) *Consider a function $Q(w;\boldsymbol{x}) : \mathbb{R}^M \times \mathbb{R}^P \to \mathbb{R}$, where $\boldsymbol{x}$ is a random vector. Assume Q is first-order differentiable with respect to w, $\nabla_w Q(w;x)$ is continuous over w, and*

$$\mathbb{E}\left|\frac{\partial Q(w;\boldsymbol{x})}{\partial w[m]}\right| \;<\; +\infty, \quad m=1,2,\ldots,M \tag{16.153}$$

Then, it holds that

$$\nabla_w\Big(\mathbb{E}\,Q(w;\boldsymbol{x})\Big) \;=\; \mathbb{E}\left(\nabla_w\,Q(w;\boldsymbol{x})\right) \tag{16.154}$$

where the expectations are over the distribution of $\boldsymbol{x}$.

Proof: We repeat similar arguments, We start from (16.146):

$$\frac{\partial \mathbb{E}\,Q(w;\boldsymbol{x})}{\partial w[m]} \;=\; \lim_{\alpha\to 0}\;\mathbb{E}\,g_m(w+t_m\alpha e_m;\boldsymbol{x}) \;=\; \lim_{\alpha\to 0}\left(\int_{x\in\mathcal{D}} g_m(w+t_m\alpha e_m;\boldsymbol{x})f_{\boldsymbol{x}}(x)dx\right) \tag{16.155}$$

Since $\nabla_w\,Q(w;x)$ is continuous with respect to w, we know that $g_m(w;x)$ is continuous with respect to w. Using the same sequence $\alpha^{(n)}$ defined in the previous proof, this property implies that

$$\lim_{\alpha\to 0} g_m\Big(w+t_m\alpha^{(n)}e_m;x\Big) \;=\; g_m(w;x) \tag{16.156}$$

which also means that for any $\epsilon>0$, there exists a positive integer N_o such that for all $n>N_o$, it holds that

$$\Big|g_m\Big(w+t_m\alpha^{(n)}e_m;x\Big) \;-\; g_m(w;x)\Big| < \epsilon \tag{16.157}$$

As a result, for any $n>N_o$, we have

$$\begin{aligned}
\Big|g_m\Big(w+t_m\alpha^{(n)}e_m;x\Big)\Big| \;&=\; \Big|g_m\Big(w+t_m\alpha^{(n)}e_m;x\Big) \;-\; g_m(w;x) \;+\; g_m(w;x)\Big| \\
&=\; \Big|g_m\Big(w+t_m\alpha^{(n)}e_m;x\Big) \;-\; g_m(w;x)\Big| \;+\; |g_m(w;x)| \\
&\overset{(16.157)}{\leq}\; |g_m(w;x)| \;+\; \epsilon
\end{aligned} \tag{16.158}$$

Next, we define, for any $w\in\mathrm{dom}(Q)$,

$$R_n(x;w) \;\triangleq\; g_m\Big(w+t_m\alpha^{(n+N)}e_m;x\Big)f_{\boldsymbol{x}}(x) \tag{16.159a}$$

$$b(w;x) \;\triangleq\; \left(|g_m(w;x)| \;+\; \epsilon\right)f_{\boldsymbol{x}}(x) \tag{16.159b}$$

It follows from (16.158) and the fact that $f_{\boldsymbol{x}}(x)\geq 0$ that $|R_n(x)|\leq b(w;x)$ for all n. Moreover, under assumption (16.153):

$$\begin{aligned}
\int_{x\in\mathcal{D}} b(w;x)dx &= \int_{x\in\mathcal{D}} \left(|g_m(w;x)| \;+\; \epsilon\right)f_{\boldsymbol{x}}(x)dx \\
&= \epsilon + \int_{x\in\mathcal{D}} |g_m(w;x)|\,f_{\boldsymbol{x}}(x)dx \\
&= \epsilon + \mathbb{E}\left|\frac{\partial Q(w;\boldsymbol{x})}{\partial w[m]}\right| < +\infty
\end{aligned} \tag{16.160}$$

which implies that $b(w;x)$ is integrable. We conclude that $R_n(x)$ is dominated by an

integrable function $b(w;x)$. Using (16.155) and the dominated convergence theorem we know that for any $w \in \operatorname{dom} Q$, it holds that

$$\begin{aligned}
\frac{\partial \mathbb{E}\, Q(w;\boldsymbol{x})}{\partial w[m]} &= \lim_{n\to+\infty} \int_{x\in\mathcal{D}} R_n(x;w)dx \\
&= \int_{x\in\mathcal{D}} \lim_{n\to+\infty} R_n(x;w)dx \\
&= \int_{x\in\mathcal{D}} \lim_{n\to+\infty} g_m\Big(w + t_m \alpha^{(n+N)} e_m; x\Big) f_{\boldsymbol{x}}(x)dx \\
&= \int_{x\in\mathcal{D}} g_m(w;x) f_{\boldsymbol{x}}(x)dx \\
&= \int_{x\in\mathcal{D}} \frac{\partial Q(w;x)}{\partial w[m]} f_{\boldsymbol{x}}(x)dx \\
&= \mathbb{E}\left(\frac{\partial Q(w;\boldsymbol{x})}{\partial w[m]}\right)
\end{aligned} \tag{16.161}$$

From this result and (16.143) we arrive at (16.154).

■

A straightforward corollary follows if the random variable $\boldsymbol{x}$ takes values in a *compact* (i.e., closed and bounded) set.

Corollary 16.1. (Switching expectation and gradient operations III) *Consider a function $Q(w;\boldsymbol{x}) : \mathbb{R}^M \times \mathbb{R}^P \to \mathbb{R}$, where $\boldsymbol{x}$ is a random vector variable taking on values in a compact set, $\mathcal{D}$. Assume Q is first-order differentiable with respect to w and $\nabla_w Q(w;\boldsymbol{x})$ is continuous with respect to w and $\boldsymbol{x}$, respectively. Then, it holds that*

$$\nabla_w\Big(\mathbb{E}\, Q(w;\boldsymbol{x})\Big) \;=\; \mathbb{E}\Big(\nabla_w Q(w;\boldsymbol{x})\Big) \tag{16.162}$$

where the expectations are over the distribution of $\boldsymbol{x}$.

Proof: Recall that $g_m(w;x) = \partial Q(w;x)/\partial w[m]$, which is continuous with respect to x since we are assuming that $\nabla_w Q(w;x)$ is continuous with respect to x. Now, given that x is defined over a compact set, we know that

$$|g_m(w;x)| < C, \quad \forall\, x \in \mathcal{D} \tag{16.163}$$

Therefore, it holds that

$$\int_{x\in\mathcal{D}} |g_m(w;x)| f_{\boldsymbol{x}}(x)dx \le C\left(\int_{x\in\mathcal{D}} f_{\boldsymbol{x}}(x)dx\right) = C < +\infty. \tag{16.164}$$

In other words, the expectation $\mathbb{E}\,|g_m(w;\boldsymbol{x})| = \mathbb{E}\,|\partial Q(w;x)/\partial w[m]|$ exists for any $m \in \{1,\cdots,M\}$. Since all conditions of Theorem 16.2 are satisfied, we conclude that (16.162) holds.

■

There are similar results allowing the exchange of expectation and *subgradient* operations when the function $Q(w;\boldsymbol{x})$ is non-differentiable at some locations. The proof of the following statement is given in Rockafellar and Wets (1981, Eq. 20) and also Wets (1989, Prop. 2.10).

Lemma 16.1. (Switching expectation and subgradient operations) *Consider a convex function $Q(w;\boldsymbol{x}) : \mathbb{R}^M \times \mathbb{R}^P \to \mathbb{R}$, where $\boldsymbol{x}$ is a random vector variable, and assume $\mathbb{E}\,Q(w;\boldsymbol{x})$ is finite in a neighborhood of w where the subgradient is computed. Then, it holds that*

$$\partial_w \Big(\mathbb{E}\,Q(w;\boldsymbol{x})\Big) \;=\; \mathbb{E}\,\Big(\partial_w\, Q(w;\boldsymbol{x})\Big) \tag{16.165}$$

where ∂_w refers to the subdifferential operator.

REFERENCES

Agarwal, A., P. L. Bartlett, P. Ravikumar, and M. J. Wainwright (2012), "Information-theoretic lower bounds on the oracle complexity of convex optimization," *IEEE Trans. Inf. Thy,*, vol. 58, no. 5, pp. 3235–3249.

Albert, A. E. and L. A. Gardner (1967), *Stochastic Approximation and Nonlinear Regression*, MIT Press, Cambridge, MA.

Amari, S. I. (1967), "A theory of adaptive pattern classifiers," *IEEE Trans. Elec. Comput.*, vol. 16, pp. 299–307.

Apostol, T. (1974), *Mathematical Analysis*, 2nd edition, Addison-Wesley, Reading, MA.

Applebaum, S. P. (1966), *Adaptive Arrays*, Rep. SPLTR 66-1, Syracuse University Research Corporation.

Bach, F. and E. Moulines (2011), "Non-asymptotic analysis of stochastic approximation algorithms for machine learning," *Proc. Advances Neural Information Processing Systems* (NIPS), pp. 451–459, Granada, Spain, 2011.

Bartle, R. G. (1995), *The Elements of Integration and Lebesgue Measure*, Wiley, NY.

Bell, R. and Y. Koren (2007a), "Scalable collaborative filtering with jointly derived neighborhood interpolation weights," *Proc. IEEE International Conference on Data Mining* (ICDM), pp. 43–52, Omaha, NE.

Bell, R. and Y. Koren (2007b), "Lessons from the Netflix prize challenge," *ACM SIGKDD Explorations Newsletter*, vol. 9, no. 2, pp. 75–79.

Bell, R., Y. Koren, and C. Volinsky (2007), "The BellKor solution to the Netflix Prize," *available at https://www.netflixprize.com/assets/ProgressPrize2007_KorBell.pdf*

Benveniste, A., M. Métivier, and P. Priouret (1987), *Adaptive Algorithms and Stochastic Approximations*, Springer-Verlag, NY.

Bertsekas, D. P. and J. N. Tsitsiklis (1996), *Neuro-Dynamic Programming*, Athena Scientific, MA.

Bertsekas, D. P. and J. N. Tsitsiklis (1997), *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, Singapore.

Bertsekas, D. P. and J. N. Tsitsiklis (2000), "Gradient convergence in gradient methods with errors," *SIAM J. Optim.*, vol. 10, no. 3, pp. 627–642.

Blum, J. R. (1954), "Multidimensional stochastic approximation methods," *Ann. Math. Stat.*, vol. 25, pp. 737–744.

Bottou, L. (1998), "Online algorithms and stochastic approximations," in *Online Learning and Neural Networks*, D. Saad, *Ed.*, Cambridge University Press.

Bottou, L. (2012), "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*, G. Montavon, G. B. Orr, and K-R. Muller, *Eds.*, 2nd edition, Lecture Notes in Computer Science, vol. 7700, pp. 421–436, Springer, NY.

Bottou, L. and O. Bousquet (2008),"The tradeoffs of large scale learning," *Proc. Advances Neural Information Processing Systems* (NIPS), vol. 20, pp. 161–168, Vancouver, Canada.

Bousquet, O. and L. Bottou (2008), "The tradeoffs of large scale learning," in *Proc. Advances Neural Information Processing Systems*, (NIPS), pp. 161–168, Vancouver, BC.

Buck, R. C. (1965), *Advanced Calculus*, McGraw-Hill, NY.

Cesa-Bianchi, N., A. Conconi, and C. Gentile (2004), "On the generalization ability of on-line learning algorithms," *IEEE Trans. Inf. Thy*, vol. 50, no. 9, pp. 2050–2057.
Cesa-Bianchi, N. and G. Lugosi (2006), *Prediction, Learning, and Games*, Cambridge University Press.
Dieudonné, J. (1969), *Foundations of Modern Analysis*, vol. 1, Academic Press, NY.
Dror, G., N. Koenigstein, Y. Koren, and M. Weimer (2012), "The Yahoo! music dataset and KDDCup 11," *Journal of Machine Learning Research*, vol. 18, pp. 8–18, 2012.
Dvoretzky, A. (1956), "On stochastic approximation," *Proc. 3rd Berkeley Symp. Math. Statist. Probab.*, vol. 1, pp. 39–56, University of California Press.
Ene, V. (1999), "Some queries concerning convergence theorems," *Real Anal. Exchange*, vol. 25, no. 2, pp. 955–958.
Funk, S. (2006), "Netflix update: Try this at home," *blog post available at the link https://sifter.org/ simon/journal/20061211.html*
Gabor, D., W. P. Z. Wilby, and R. Woodcock (1961), "An universal nonlinear filter, predictor, and simulator which optimizes itself by a learning process," *Proc. IEE*, vol. 108, no. 40, pp. 422-436.
Gladyshev, E. G. (1965), "On stochastic approximations," *Theory of Probability and its Applications*, vol. 10, pp. 275–278.
Hassibi, B., A. H. Sayed and T. Kailath (1994a), "$\mathcal{H}^\infty-$optimality criteria for LMS and backpropagation," *Proc. Advances Neural Information Processing Systems* (NIPS), vol. 6, pp. 351–358, Denver, CO.
Hassibi, B., A. H. Sayed, and T. Kailath (1996), "$\mathcal{H}^\infty$-optimality of the LMS algorithm," *IEEE Trans. Signal Processing*, vol. 44, no. 2, pp. 267–280.
Haykin, S. (2001), *Adaptive Filter Theory*, 4th edition, Prentice Hall, NJ.
Hewitt, E. and K. Stromberg (1969), *Real and Abstract Analysis*, Springer Verlag.
Hu, Y. F., Y. Koren, and C. Volinsky (2008), "Collaborative filtering for implicit feedback datasets," *Proc. IEEE International Conference on Data Mining* (ICDM), pp. 263–272, Pisa, Italy.
Kaczmarz, S. (1937), "Angenäherte Auflösung von Systemen linearer Gleichungen," *Bull. Int. Acad. Polon. Sci. Lett.* A, pp. 335–357.
Kiefer, J. and J. Wolfowitz (1952), "Stochastic estimation of the maximum of a regression function," *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462–466.
Koren, Y. (2008), "Factorization meets the neighborhood: A multifaceted collaborative filtering model," *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 426–434, Las Vegas, NV.
Koren, Y., R. Bell, and C. Volinsky (2009), "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37.
Kushner, H. J. (1984), *Approximation and Weak Convergence Methods for Random Processes, with Applications to Stochastic System Theory*, MIT Press, Cambridge, MA.
Kushner, H. J. and D. S. Clark (1978), *Stochastic Approximation for Constrained and Unconstrained Systems*, Springer-Verlag, NY.
Kushner, H. J. and G. G. Yin (2003), *Stochastic Approximation and Recursive Algorithms and Applications*, Springer, NY.
Lai, T. L. (2003), "Stochastic approximation," *Annals of Statistics*, vol. 31, no. 2, pp. 391–406.
Lewin, J. W. (1987), "Some applications of the bounded convergence theorem for an introductory course in analysis," *The American Mathematical Monthly*, vol. 94, no. 10, pp. 988–993.
Ljung, L. (1977), "Analysis of recursive stochastic algorithms," *IEEE Trans. Automat. Contr.*, vol. 22, pp. 551–575.
Luxemburg, W. A. J. (1971), "Arzelà's dominated convergence theorem for the Riemann integral," *Amer. Math. Monthly*, vol. 78, pp. 970–979.
Norris, J. R. (2013), *Probability and Measure*, unpublished notes. Available online at http://www.statslab.cam.ac.uk/ james/Lectures/pm.pdf
Marti, K. (2005), *Stochastic Optimization Methods*, Springer, NY.

Mattson, R. L. (1959a), *The Design and Analysis of an Adaptive System for Statitiscal Classification*, S. M. Thesis, MIT.

Mattson, R. L. (1959b), "A self-organizing binary system," *Proc. Eastern Joint IRE-AIEE-ACM Computer Conference*, pp. 212–217, Boston, MA.

Mendel, J. M. and K. S. Fu (1970), *Adaptive, Learning, and Pattern Recognition Systems: Theory and Applications*, Academic Press, NY.

Natanson, I. P. (1961), *Theory of Functions of a Real Variable*, 2nd edition, Frederick Ungar Publishing Co., NY.

Paterek, A. (2007), "Improving regularized singular value decomposition for collaborative filtering," *Proc. KDD Cup and Workshop*, pp. 39–42, ACM Press.

Pilaszy, I., D. Zibriczky, and D. Tikk (2010), "Fast ALS-based matrix factorization for explicit and implicit feedback datasets," *Proc. ACM conference on Recommender Systems*, pp. 71–78, Barcelona, Spain.

Polyak, B. T. (1987), *Introduction to Optimization*, Optimization Software, NY.

Polyak, B. T. and A. Juditsky (1992), "Acceleration of stochastic approximation by averaging," *SIAM J. Control and Optim.*, vol. 30, no. 4, pp. 838–855.

Robbins, H. and S. Monro (1951), "A stochastic approximation method," *Ann. Math. Stat.*, vol. 22, pp. 400–407.

Rockafellar, R. T. and R. Wets (1981), "On the interchange of sudifferentiation and conditional expectation for convex functionals," *International Institute for Applied Systems Analysis*, IIASA working paper WP-81-089, Laxenburg, Austria.

Rosenblatt, F. (1957), *The Perceptron: A Perceiving and Recognizing Automaton*, Technical Report 85-460-1, Project PARA, Cornell Aeronautical Lab.

Rosenblatt, F.(1958), "The Perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408.

Rosenblatt, F. (1962), *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan Press, Washington, DC.

Royden, H. L. (1988), *Real Analysis*, Prentice Hall, NJ.

Rudin, W. (1976), *Principles of Mathematical Analysis*, 3rd edition, McGraw-Hill, NY.

Ruppert, D. (1988), *Efficient Estimation From a Slowly Convergent Robbins-Monro Process*, Technical Report 781, Cornell University, School of Operations Research and Industrial Engineering.

Sayed, A. H. (2003), *Fundamentals of Adaptive Filtering*, Wiley, NJ.

Sayed, A. H. (2008), *Adaptive Filters*, Wiley, NJ.

Sayed, A. H. (2014a), *Adaptation, Learning, and Optimization over Networks,* Foundations and Trends in Machine Learning, NOW Publishers, vol. 7, no. 4–5, pp. 311–801.

Sayed, A. H. and M. Rupp (1996),"Error energy bounds for adaptive gradient algorithms," *IEEE Trans. Signal Processing*, vol. 44, no. 8, pp. 1982–1989.

Schmetterer, L. (1961), "Stochastic approximation," *Proc. Berkeley Symp. Math. Statist. Probab.*, pp. 587–609, Berkeley, CA.

Sefl, O. (1960), "Filters and predictors which adapt their values to unknown parameters of the input process," *Trans. 2nd Conference on Information Theory,* Czechoslovak Academy of Sciences, Prague.

Shalev-Shwartz, S. (2011), "Online learning and online convex optimization," *Foundations and Trends in Machine Learning*, vol. 4, no. 2, pp. 107–194.

Spall, J. C. (2003), *Introduction to Stochastic Search and Optimization: Estimation, Simulation and Control*, Wiley, NJ.

Strohmer, T. and R. Vershynin (2009), "A randomized Kaczmarz algorithm with exponential convergence," *J. Fourier Analysis and Applications*, vol. 15, no. 2, pp. 262–278.

Symeonidis, P. and A. Zioupos (2017), *Matrix and Tensor Factorization Techniques for Recommender Systems*, Springer, NY.

Takacs, G., I. Pilaszy, B. Nemeh, and D. Tikk (2007), "Major components of the gravity recommendation system," *SIGKDD Explorations*, vol. 9, pp. 80–84.

Theodoridis, S. (2015), *Machine Learning: A Bayesian and Optimization Perspective*, Academic Press.

Troutman, J. L. (1996), *Variational Calculus and Optimal Control*, Springer, NY.

Tsypkin, Y. Z. (1971), *Adaptation and Learning in Automatic Systems*, Academic Press, New York.

von Mises, R. and H. Pollaczek-Geiringer (1929), "Praktische verfahren der gleichungsauflösung," *Z. Agnew. Math. Mech.*, vol. 9.

Wasan, M. T. (1969), *Stochastic Approximation*, Cambridge University Press, London.

Wetherhill, G. B. (1966), *Sequential Methods in Statistics,* Methuen, London.

Wets, R. (1989), "Stochastic programming," in *Handbook for Operations Research and Management Sciences*, G. Nemhauser and A. Rinnnooy Kan, *Eds.*, vol. 1, pp. 573–629.

Widrow, B. and M. E. Hoff (1960), "Adaptive switching circuits," *IRE WESCON Conv. Rec.*, Institute of Radio Engineers, pt. 4, pp. 96–104.

Widrow, B., P. Mantey, L. J. Griffiths, and B. Goode (1967), "Adaptive antenna systems," *Proc. IEEE*, vol. 55, no. 12, pp. 2143–2159.

Widrow, B. and S. D. Stearns (1985), *Adaptive Signal Processing*, Prentice Hall, NJ.

Zhou, Y., D. Wilkinson, R. Schreiber, and R. Pan (2008), "Large-scale parallel collaborative filtering for the Netflix prize," in *Algorithmic Aspects in Information and Management*, pp. 337–348, Springer, NY.