CHAPTER

Asynchronous Adaptive Networks[1]

1

Ali H. Sayed,[*] **and Xiaochuan Zhao**[**]

[*]*École Polytechnique Fédérale de Lausanne, EPFL, Switzerland*
[**]*University of California at Los Angeles, UCLA, USA*
[**]*Corresponding author:* `ali.sayed@epfl.ch`

ABSTRACT

The overview article [1] surveyed advances related to adaptation, learning, and optimization over synchronous networks. Various distributed strategies were discussed that enable a collection of networked agents to interact locally in response to online streaming data and to continually learn and adapt to drifts in the data and models. Under reasonable technical conditions, the adaptive networks were shown to be mean-square stable in the slow adaptation regime, and their mean-square-error performance and convergence rate were characterized in terms of the network topology and data statistical properties [5]. Classical results for single-agent adaptation and learning were recovered as special cases. Following the works [2]-[4], this chapter complements the exposition from [1] and extends the results to asynchronous networks where agents are subject to various sources of uncertainties that influence their behavior, including randomly changing topologies, random link failures, random data arrival times, and agents turning on and off randomly. In an asynchronous environment, agents may stop updating their solutions or may stop sending or receiving information in a random manner and without coordination with other agents. The presentation will reveal that the mean-square-error performance of asynchronous networks remains largely unaltered compared to synchronous networks. The results justify the remarkable resilience of cooperative networks.

Keywords: Multi-agent networks, distributed optimization, diffusion strategy, asynchronous behavior, adaptive networks.

1.1 INTRODUCTION

Adaptive networks consist of a collection of agents with learning abilities. The agents interact with each other on a local level and diffuse information across the network to solve inference and optimization tasks in a decentralized manner. Such networks are scalable, robust to node and link failures, and are particularly suitable for learning from big data sets by tapping into the power of collaboration among distributed agents. The networks are also endowed with cognitive abilities due to the sensing abilities of their agents, their interactions with their neighbors, and the embedded feedback mechanisms for acquiring and refining information. Each agent is not only capable of sensing data and experiencing the environment directly, but it also receives information through interactions with its neighbors and processes and analyzes this information to drive its learning process.

As already indicated in [1, 5], there are many good reasons for the peaked interest in networked solutions, especially in this day and age when the word "network" has become commonplace whether one is referring to social networks, power networks, transportation networks, biological networks, or other networks. Some of these reasons have to do with the benefits of cooperation over networks in terms of improved performance and improved robustness and resilience to failure. Other reasons deal with privacy and secrecy considerations where agents may not be comfortable sharing their data with remote fusion centers. In other situations, the data may already be available in dispersed locations, as happens with cloud computing. One may also be interested in learning and extracting information through data mining from large data sets. Decentralized learning procedures offer an attractive approach to dealing with such data sets. Decentralized mechanisms can also serve as important enablers for the design of robotic swarms, which can assist in the exploration of disaster areas.

1.1.1 ASYNCHRONOUS BEHAVIOR

The survey article [1] and monograph [5] focused on the case of synchronous networks where data arrive at all agents in a synchronous manner and updates by the agents are also performed in a synchronous manner. The network topology was assumed to remain largely static during the adaptation process. Under these conditions, the limits of performance and stability of these networks were identified in some detail for two main classes of distributed strategies: consensus and diffusion constructions. In this chapter, we extend the overview from [1] to cover *asynchronous* environments. In such environments, the operation of the network can suffer from the occurrence of various random events including randomly changing topologies, random link failures, random data arrival times, and agents turning on and off randomly. Agents may also stop updating their solutions or may stop sending or receiving information in a random manner and without coordination with other agents. Results in [2, 3, 4] examined the implications of such asynchronous events on network performance in some detail and under a fairly general model for the random events.

The purpose of this chapter is to summarize the key conclusions from these works in a manner that complements the presentation from [1] for the benefit of the reader. While the works [2, 3, 4] consider a broader formulation involving complex-valued variables, we limit the discussion here to real-valued variables in order not to overload the notation and to convey the key insights more directly. Proofs and derivations are often omitted and can be found in the above references; the emphasis is on presenting the results in a motivated manner and on commenting on the insights they provide into the operation of asynchronous networks.

We indicated in [2, 3, 4] that there already exist many useful studies in the literature on the performance of consensus strategies in the presence of asynchronous events (see, e.g., [11]–[21]. There are also some studies in the context of diffusion strategies [22, 23]. However, with the exception of the latter two works, the earlier references assumed conditions that are not generally favorable for applications involving continuous *adaptation* and learning. For example, some of the works assumed a decaying step-size, which turns off adaptation after sufficient iterations have passed. Some other works assumed noise free data, which is a hindrance when learning from data perturbed by interferences and distortions. A third class of works focused on studying pure averaging algorithms, which are not required to respond to continuous data streaming. In the works [2, 3, 4], we adopted a more general asynchronous model that removes these limitations by allowing for various sources of random events and, moreover, the events are allowed to occur simultaneously. We also examined learning algorithms that respond to streaming data to enable adaptation. The main conclusion from the analysis in these works, and which will be summarized in future sections, is that asynchronous networks can still behave in a mean-square-error stable manner for sufficiently small step-sizes and, interestingly, their steady-state performance level is only slightly affected in comparison to synchronous behavior. The iterates computed by the various agents are still able to converge and hover around an agreement state with a small mean-square-error. These are reassuring results that support the intrinsic robustness and resilience of network-based cooperative solutions.

1.1.2 ORGANIZATION OF THE CHAPTER

Readers will benefit more from this chapter if they review first the earlier article [1]. We continue to follow a similar structure here, as well as a similar notation, since the material in both this chapter and the earlier reference [1] are meant to complement each other. We organize the presentation into three main components. The first part (Sec. 1.2) reviews fundamental results on adaptation and learning by *single* stand-alone agents. The second part (Sec. 1.3) covers asynchronous centralized solutions. The objective is to explain the gain in performance that results from aggregating the data from the agents and processing it centrally at a fusion center. The centralized performance is used as a frame of reference for assessing various implementations. While centralized solutions can be powerful, they nevertheless suffer from a number of limitations. First, in real-time applications where agents collect data continu-

ously, the repeated exchange of information back and forth between the agents and the fusion center can be costly especially when these exchanges occur over wireless links or require nontrivial routing resources. Second, in some sensitive applications, agents may be reluctant to share their data with remote centers for various reasons including privacy and secrecy considerations. More importantly perhaps, centralized solutions have a critical point of failure: if the central processor fails, then this solution method collapses altogether.

For these reasons, we cover in the remaining sections of the chapter (Secs. 1.4 and 1.5) distributed asynchronous strategies of the consensus and diffusion types, and examine their dynamics, stability, and performance metrics. In the distributed mode of operation, agents are connected by a topology and they are permitted to share information only with their immediate neighbors. The study of the behavior of such networked agents is more challenging than in the single-agent and centralized modes of operation due to the coupling among interacting agents and due to the fact that the networks are generally sparsely connected.

1.2 SINGLE-AGENT ADAPTATION AND LEARNING

We begin our treatment by reviewing stochastic gradient algorithms, with emphasis on their application to the problems of adaptation and learning by stand-alone agents.

1.2.1 RISK AND LOSS FUNCTIONS

Thus, let $J(w) : \mathbb{R}^{M \times 1} \mapsto \mathbb{R}$ denote a twice-differentiable real-valued (cost or utility or risk) function of a real-valued vector argument, $w \in \mathbb{R}^{M \times 1}$. When the variable w is complex-valued, some important technical differences arise that are beyond the scope of this chapter; they are addressed in [2, 3, 4, 5]. Likewise, some adjustments to the arguments are needed when the risk function is non-smooth (non-differentiable); as explained in the works [24, 25]. It is sufficient for our purposes in this chapter to limit the presentation to real arguments and smooth risk functions without much loss in generality.

We denote the gradient vectors of $J(w)$ relative to w and w^{T} by the following row and column vectors, respectively:

$$\nabla_w J(w) \triangleq \left[\frac{\partial J(w)}{\partial w_1}, \frac{\partial J(w)}{\partial w_2}, \ldots, \frac{\partial J(w)}{\partial w_M} \right] \tag{1.1a}$$

$$\nabla_{w^{\mathsf{T}}} J(w) \triangleq [\nabla_w J(w)]^{\mathsf{T}} \tag{1.1b}$$

These definitions are in terms of the partial derivatives of $J(w)$ relative to the individual entries of $w = \text{col}\{w_1, w_2, \ldots, w_M\}$, where the notation $\text{col}\{\cdot\}$ refers to a column vector that is formed by stacking its arguments on top of each other. Likewise, the Hessian matrix of $J(w)$ with respect to w is defined as the following $M \times M$ symmet-

ric matrix:

$$\nabla_w^2 J(w) \triangleq \nabla_{w^\mathsf{T}}[\nabla_w J(w)] = \nabla_w[\nabla_{w^\mathsf{T}} J(w)] \tag{1.1c}$$

which is constructed from two successive gradient operations. It is common in adaptation and learning applications for the risk function $J(w)$ to be constructed as the expectation of some loss function, $Q(w; \boldsymbol{x})$, where the **boldface** variable $\boldsymbol{x}$ is used to denote some random data, say,

$$J(w) = \mathbb{E}\, Q(w; \boldsymbol{x}) \tag{1.2}$$

and the expectation is evaluated over the distribution of $\boldsymbol{x}$.

Example 1 (Mean-square-error (MSE) costs). Let $\boldsymbol{d}$ denote a zero-mean scalar random variable with variance $\sigma_d^2 = \mathbb{E}\,\boldsymbol{d}^2$, and let $\boldsymbol{u}$ denote a zero-mean $1 \times M$ random vector with covariance matrix $R_u = \mathbb{E}\,\boldsymbol{u}^\mathsf{T}\boldsymbol{u} > 0$. The combined quantities $\{\boldsymbol{d}, \boldsymbol{u}\}$ represent the random variable $\boldsymbol{x}$ referred to in (1.2). The cross-covariance vector is denoted by $r_{du} = \mathbb{E}\,\boldsymbol{d}\boldsymbol{u}^\mathsf{T}$. We formulate the problem of estimating $\boldsymbol{d}$ from $\boldsymbol{u}$ in the linear least-mean-squares-error sense or, equivalently, the problem of seeking the vector w^o that minimizes the quadratic cost function:

$$J(w) \triangleq \mathbb{E}\,(\boldsymbol{d} - \boldsymbol{u}w)^2 = \sigma_d^2 - r_{du}^\mathsf{T} w - w^\mathsf{T} r_{du} + w^\mathsf{T} R_u w \tag{1.3a}$$

This cost corresponds to the following choice for the loss function:

$$Q(w; \boldsymbol{x}) = (\boldsymbol{d} - \boldsymbol{u}w)^2 \tag{1.3b}$$

Such quadratic costs are widely used in estimation and adaptation problems [26]–[30]. They are also widely used as quadratic risk functions in machine learning applications [31, 32]. The gradient vector and Hessian matrix of $J(w)$ are easily seen to be:

$$\nabla_w J(w) = 2\,(R_u w - r_{du})^\mathsf{T}\,, \quad \nabla_w^2 J(w) = 2R_u \tag{1.3c}$$

∎

Example 2 (Logistic or log-loss risks). Let $\boldsymbol{\gamma}$ denote a binary random variable that assumes the values ± 1, and let $\boldsymbol{h}$ denote an $M \times 1$ random (feature) vector with $R_h = \mathbb{E}\,\boldsymbol{h}\boldsymbol{h}^\mathsf{T}$. The combined quantities $\{\boldsymbol{\gamma}, \boldsymbol{h}\}$ represent the random variable $\boldsymbol{x}$ referred to in (1.2). In the context of machine learning and pattern classification problems [31, 32, 33], the variable $\boldsymbol{\gamma}$ designates the class that feature vector $\boldsymbol{h}$ belongs to. In these problems, one seeks the vector w^o that minimizes the regularized logistic risk function:

$$J(w) \triangleq \frac{\rho}{2}\|w\|^2 + \mathbb{E}\left\{\ln\left[1 + e^{-\boldsymbol{\gamma}\boldsymbol{h}^\mathsf{T} w}\right]\right\} \tag{1.4a}$$

where $\rho > 0$ is some regularization parameter, $\ln(\cdot)$ is the natural logarithm function, and $\|w\|^2 = w^\mathsf{T} w$. The risk (1.4a) corresponds to the following choice for the loss function:

$$Q(w; \boldsymbol{x}) \triangleq \frac{\rho}{2}\|w\|^2 + \ln\left[1 + e^{-\boldsymbol{\gamma}\boldsymbol{h}^\mathsf{T} w}\right] \tag{1.4b}$$

Once w^o is recovered, its value can be used to classify new feature vectors, say, $\{\boldsymbol{h}_\ell\}$, into classes $+1$ or -1. This can be achieved by assigning feature vectors with $\boldsymbol{h}_\ell^\mathsf{T} w^o \geq 0$ to one class and feature vectors with

6   **CHAPTER 1** Asynchronous Adaptive Networks

$h_\ell^\mathsf{T} w^o < 0$ to another class. It can be easily verified that:

$$\nabla_w J(w) = \rho w^\mathsf{T} - \mathbb{E} \left\{ \gamma h^\mathsf{T} \cdot \frac{e^{-\gamma h^\mathsf{T} w}}{1 + e^{-\gamma h^\mathsf{T} w}} \right\} \qquad (1.4c)$$

$$\nabla_w^2 J(w) = \rho I_M + \mathbb{E} \left\{ h h^\mathsf{T} \cdot \frac{e^{-\gamma h^\mathsf{T} w}}{\left(1 + e^{-\gamma h^\mathsf{T} w} \right)^2} \right\} \qquad (1.4d)$$

where I_M denotes the identity matrix of size $M \times M$.

∎

1.2.2 CONDITIONS ON COST FUNCTION

Stochastic gradient algorithms are powerful iterative procedures for solving optimization problems of the form

$$\min_w \ J(w) \qquad (1.5)$$

While the analysis that follows can be pursued under more relaxed conditions (see, e.g., the treatments in [34]–[37]), it is sufficient for our purposes to require $J(w)$ to be strongly-convex and twice-differentiable with respect to w. The cost function $J(w)$ is said to be ν-strongly convex if, and only if, its Hessian matrix is sufficiently bounded away from zero [35, 38, 39, 40]:

$$J(w) \text{ is } \nu\text{-strongly convex} \iff \nabla_w^2 J(w) \ge \nu I_M > 0 \qquad (1.6)$$

for all w and for some scalar $\nu > 0$, where the notation $A > 0$ signifies that matrix A is positive-definite. Strong convexity is a useful condition in the context of adaptation and learning from streaming data because it helps guard against ill-conditioning in the algorithms; it also helps ensure that $J(w)$ has a *unique* global minimum, say, at location w^o; there will be no other minima, maxima, or saddle points. In addition, it is well-known that strong convexity helps endow stochastic-gradient algorithms with geometric convergence rates in the order of $O(\alpha^i)$, for some $0 \le \alpha < 1$ and where i is the iteration index [35, 36].

In many problems of interest in adaptation and learning, the cost function $J(w)$ is either already strongly-convex or can be made strongly-convex by means of regularization. For example, it is common in machine learning problems [31, 32] and in adaptation and estimation problems [28, 30] to incorporate regularization factors into the cost functions; these factors help ensure strong convexity. For instance, the mean-square-error cost (1.3a) is strongly convex whenever $R_u > 0$. If R_u happens to be singular, then the following regularized cost will be strongly convex:

$$J(w) \triangleq \frac{\rho}{2} \|w\|^2 + \mathbb{E}(d - uw)^2 \qquad (1.7)$$

where $\rho > 0$ is a regularization parameter similar to (1.4a).

Besides strong convexity, we shall also assume that the gradient vector of $J(w)$ is

δ-Lipschitz, namely, there exists $\delta > 0$ such that

$$\|\nabla_w J(w_2) - \nabla_w J(w_1)\| \leq \delta \|w_2 - w_1\| \tag{1.8}$$

for all w_1 and w_2. It can be verified that for twice-differentiable costs, conditions (1.6) and (1.8) combined are equivalent to

$$0 < \nu I_M \leq \nabla_w^2 J(w) \leq \delta I_M \tag{1.9}$$

For example, it is clear that the Hessian matrices in (1.3c) and (1.4d) satisfy this property since

$$2\lambda_{\min}(R_u) I_M \leq \nabla_w^2 J(w) \leq 2\lambda_{\max}(R_u) I_M \tag{1.10a}$$

in the first case and

$$\rho I_M \leq \nabla_w^2 J(w) \leq (\rho + \lambda_{\max}(R_h)) I_M \tag{1.10b}$$

in the second case, where the notation $\lambda_{\min}(R)$ and $\lambda_{\max}(R)$ refers to the smallest and largest eigenvalues of the symmetric matrix argument, R, respectively. In summary, we will be assuming the following conditions [5, 41, 42, 2].

Assumption 1 (**Conditions on cost function**). *The cost function $J(w)$ is twice-differentiable and satisfies* (1.9) *for some positive parameters $\nu \leq \delta$. Condition* (1.9) *is equivalent to requiring $J(w)$ to be ν-strongly convex and for its gradient vector to be δ-Lipschitz as in* (1.6) *and* (1.8)*, respectively.*

∎

1.2.3 **STOCHASTIC-GRADIENT APPROXIMATION**

The traditional gradient-descent algorithm for solving (1.5) takes the form:

$$w_i = w_{i-1} - \mu \nabla_{w^\top} J(w_{i-1}), \quad i \geq 0 \tag{1.11}$$

where $i \geq 0$ is an iteration index and $\mu > 0$ is a small step-size parameter. Starting from some initial condition, w_{-1}, the iterates $\{w_i\}$ correspond to successive estimates for the minimizer w^o. In order to run recursion (1.11), we need to have access to the true gradient vector. This information is generally unavailable in most instances involving learning from data. For example, when cost functions are defined as the expectations of certain loss functions as in (1.2), the statistical distribution of the data x may not be known beforehand. In that case, the exact form of $J(w)$ will not be known since the expectation of $Q(w; x)$ cannot be computed. In such situations, it is customary to replace the true gradient vector, $\nabla_{w^\top} J(w_{i-1})$, by an instantaneous approximation for it, and which we shall denote by $\widehat{\nabla_{w^\top} J}(w_{i-1})$. Doing so leads to the following *stochastic-gradient* recursion in lieu of (1.11):

$$\boxed{w_i = w_{i-1} - \mu \widehat{\nabla_{w^\top} J}(w_{i-1}), \quad i \geq 0} \tag{1.12}$$

8   **CHAPTER 1** Asynchronous Adaptive Networks

We use the **boldface** notation, $\boldsymbol{w}_i$, for the iterates in (1.12) to highlight the fact that these iterates are now randomly perturbed versions of the values $\{w_i\}$ generated by the original recursion (1.11). The random perturbations arise from the use of the approximate gradient vector. The boldface notation is therefore meant to emphasize the random nature of the iterates in (1.12). We refer to recursion (1.12) as a *synchronous* implementation since updates occur continuously over the iteration index i. This terminology is meant to distinguish the above recursion from its *asynchronous* counterpart, which is introduced later in Sec. 1.2.5.

We illustrate construction (1.12) by considering a scenario from classical adaptive filter theory [28, 26, 27], where the gradient vector is approximated directly from data realizations. The construction will reveal why stochastic-gradient implementations of the form (1.12), using approximate rather than exact gradient information, become naturally endowed with the ability to respond to *streaming* data.

Example 3 (**LMS adaptation**). Let $\boldsymbol{d}(i)$ denote a streaming sequence of zero-mean random variables with variance $\sigma_d^2 = \mathbb{E}\,\boldsymbol{d}^2(i)$. Let $\boldsymbol{u}_i$ denote a streaming sequence of $1 \times M$ independent zero-mean random vectors with covariance matrix $R_u = \mathbb{E}\,\boldsymbol{u}_i^{\mathsf{T}}\boldsymbol{u}_i > 0$. Both processes $\{\boldsymbol{d}(i), \boldsymbol{u}_i\}$ are assumed to be jointly wide-sense stationary. The cross-covariance vector between $\boldsymbol{d}(i)$ and $\boldsymbol{u}_i$ is denoted by $r_{du} = \mathbb{E}\,\boldsymbol{d}(i)\boldsymbol{u}_i^{\mathsf{T}}$. The data $\{\boldsymbol{d}(i), \boldsymbol{u}_i\}$ are assumed to be related via a linear regression model of the form:

$$\boldsymbol{d}(i) = \boldsymbol{u}_i w^o + \boldsymbol{v}(i) \tag{1.13a}$$

for some unknown parameter vector w^o, and where $\boldsymbol{v}(i)$ is a zero-mean white-noise process with power $\sigma_v^2 = \mathbb{E}\,\boldsymbol{v}^2(i)$ and assumed independent of $\boldsymbol{u}_j$ for all i, j. Observe that we are using parentheses to represent the time-dependency of a scalar variable, such as writing $\boldsymbol{d}(i)$, and subscripts to represent the time-dependency of a vector variable, such as writing $\boldsymbol{u}_i$. This convention will be used throughout the chapter. In a manner similar to Example 1, we again pose the problem of estimating w^o by minimizing the mean-square error cost

$$J(w) = \mathbb{E}\,(\boldsymbol{d}(i) - \boldsymbol{u}_i w)^2 \equiv \mathbb{E}\,Q(w; \boldsymbol{x}_i) \tag{1.13b}$$

where now the quantities $\{\boldsymbol{d}(i), \boldsymbol{u}_i\}$ represent the random data $\boldsymbol{x}_i$ in the definition of the loss function, $Q(w; \boldsymbol{x}_i)$. Using (1.11), the gradient-descent recursion in this case will take the form:

$$w_i = w_{i-1} - 2\mu\,[R_u w_{i-1} - r_{du}], \quad i \geq 0 \tag{1.13c}$$

The main difficulty in running this recursion is that it requires knowledge of the moments $\{r_{du}, R_u\}$. This information is rarely available beforehand; the adaptive agent senses instead realizations $\{\boldsymbol{d}(i), \boldsymbol{u}_i\}$ whose statistical distributions have moments $\{r_{du}, R_u\}$. The agent can use these realizations to approximate the moments and the true gradient vector. There are many constructions that can be used for this purpose, with different constructions leading to different adaptive algorithms [29, 28, 26, 27]. It is sufficient to focus on one of the most popular adaptive algorithms, which results from using the data $\{\boldsymbol{d}(i), \boldsymbol{u}_i\}$ to compute *instantaneous* approximations for the unavailable moments as follows:

$$r_{du} \approx \boldsymbol{d}(i)\boldsymbol{u}_i^{\mathsf{T}}, \qquad R_u \approx \boldsymbol{u}_i^{\mathsf{T}}\boldsymbol{u}_i \tag{1.13d}$$

By doing so, the true gradient vector is approximated by:

$$\widehat{\nabla_{w^{\mathsf{T}}}}\,J(w) = 2\left[\boldsymbol{u}_i^{\mathsf{T}}\boldsymbol{u}_i w - \boldsymbol{u}_i^{\mathsf{T}}\boldsymbol{d}(i)\right] = \nabla_{w^{\mathsf{T}}}\,Q(w; \boldsymbol{x}_i) \tag{1.13e}$$

Observe that this construction amounts to replacing the true gradient vector, $\nabla_{w^{\mathsf{T}}}\,J(w)$, by the gradient vector of the loss function itself (which, equivalently, amounts to dropping the expectation operator). Substituting (1.13e) into (1.13c) leads to the well-known (synchronous) least-mean-squares (LMS, for

short) algorithm [26, 27, 28, 43]:

$$\boxed{\boldsymbol{w}_i = \boldsymbol{w}_{i-1} + 2\mu\boldsymbol{u}_i^\top \left[\boldsymbol{d}(i) - \boldsymbol{u}_i\boldsymbol{w}_{i-1}\right], \quad i \geq 0} \tag{1.13f}$$

The LMS algorithm is therefore a stochastic-gradient algorithm. By relying directly on the instantaneous data $\{\boldsymbol{d}(i), \boldsymbol{u}_i\}$, the algorithm is infused with useful tracking abilities. This is because drifts in the model w^o from (1.13a) will be reflected in the data $\{\boldsymbol{d}(i), \boldsymbol{u}_i\}$, which are used directly in the update (1.13f).

∎

If desired, it is also possible to employ iteration-dependent step-size sequences, $\mu(i)$, in (1.12) instead of the constant step-size μ, and to require $\mu(i)$ to satisfy

$$\sum_{i=0}^{\infty} \mu^2(i) < \infty, \qquad \sum_{i=0}^{\infty} \mu(i) = \infty \tag{1.14}$$

Under some technical conditions, it is well-known that such step-size sequences ensure the convergence of $\boldsymbol{w}_i$ towards w^o almost surely as $i \to \infty$ [5, 35, 36, 37]. However, conditions (1.14) force the step-size sequence to decay to zero, which is problematic for applications requiring continuous adaptation and learning from streaming data. This is because, in such applications, it is not unusual for the location of the minimizer, w^o, to drift with time. With $\mu(i)$ decaying towards zero, the stochastic-gradient algorithm (1.12) will stop updating and will not be able to track drifts in the solution. For this reason, we shall focus on constant step-sizes from this point onwards since we are interested in solutions with tracking abilities.

Now, the use of an approximate gradient vector in (1.12) introduces perturbations relative to the operation of the original recursion (1.11). We refer to the perturbation as gradient noise and define it as the difference:

$$\boxed{\boldsymbol{s}_i(\boldsymbol{w}_{i-1}) \overset{\Delta}{=} \widehat{\nabla_{w^\top} J}(\boldsymbol{w}_{i-1}) - \nabla_{w^\top} J(\boldsymbol{w}_{i-1})} \tag{1.15}$$

The presence of this perturbation prevents the stochastic iterate, $\boldsymbol{w}_i$, from converging almost-surely to the minimizer w^o when constant step-sizes are used. Some deterioration in performance will occur and the iterate $\boldsymbol{w}_i$ will instead fluctuate close to w^o. We will assess the size of these fluctuations by measuring their steady-state mean-square value (also called mean-square-deviation or MSD). It will turn out that the MSD is small and in the order of $O(\mu)$ — see (1.28c) further ahead. It will also turn out that stochastic-gradient algorithms converge towards their MSD levels at a geometric rate. In this way, we will be able to conclude that adaptation with small constant step-sizes can still lead to reliable performance in the presence of gradient noise, which is a reassuring result. We will also be able to conclude that adaptation with constant step-sizes is useful even for stationary environments. This is because it is generally sufficient in practice to reach an iterate $\boldsymbol{w}_i$ within some fidelity level from w^o in a *finite* number of iterations. As long as the MSD level is satisfactory, a stochastic-gradient algorithm will be able to attain satisfactory fidelity

within a reasonable time frame. In comparison, although diminishing step-sizes ensure almost-sure convergence of w_i to w^o, they nevertheless disable tracking and can only guarantee slower than geometric rates of convergence (see, e.g., [5, 35, 36, 5]). The next example from [41] illustrates the nature of the gradient noise process (1.15) in the context of mean-square-error adaptation.

Example 4 (**Gradient noise**). It is clear from the expressions in Example 3 that the corresponding gradient noise process is

$$s_i(w_{i-1}) = 2\left(R_u - u_i^\mathsf{T} u_i\right)\widetilde{w}_{i-1} - 2u_i^\mathsf{T} v(i) \tag{1.16a}$$

where we introduced the error vector:

$$\widetilde{w}_i \triangleq w^o - w_i \tag{1.16b}$$

Let the symbol $\mathcal{F}_{i-1}$ represent the collection of all possible random events generated by the past iterates $\{w_j\}$ up to time $i-1$ (more formally, $\mathcal{F}_{i-1}$ is the *filtration* generated by the random process w_j for $j \leq i-1$):

$$\mathcal{F}_{i-1} \triangleq \text{filtration}\{w_{-1}, w_0, w_1, \ldots, w_{i-1}\} \tag{1.16c}$$

It follows from the conditions on the random processes $\{u_i, v(i)\}$ in Example 3 that

$$\mathbb{E}\left[s_i(w_{i-1})|\mathcal{F}_{i-1}\right] = 0 \tag{1.16d}$$

$$\mathbb{E}\left[\|s_i(w_{i-1})\|^2|\mathcal{F}_{i-1}\right] \leq 4c\,\|\widetilde{w}_{i-1}\|^2 + 4\sigma_v^2\,\text{Tr}(R_u) \tag{1.16e}$$

for some constant $c \geq 0$. If we take expectations of both sides of (1.16e), we further conclude that the variance of the gradient noise, $\mathbb{E}\|s_i(w_{i-1})\|^2$, is bounded by the combination of two factors. The first factor depends on the quality of the iterate, $\mathbb{E}\|\widetilde{w}_{i-1}\|^2$, while the second factor depends on σ_v^2. Therefore, even if the adaptive agent is able to approach w^o with great fidelity so that $\mathbb{E}\|\widetilde{w}_{i-1}\|^2$ is small, the size of the gradient noise will still depend on σ_v^2.   ∎

1.2.4 **CONDITIONS ON GRADIENT NOISE PROCESS**

In order to examine the convergence and performance properties of the stochastic-gradient recursion (1.12), it is necessary to introduce some assumptions on the stochastic nature of the gradient noise process, $s_i(\cdot)$. The conditions that we introduce in the sequel are similar to conditions used earlier in the optimization literature, e.g., in [35, pp. 95–102] and [44, p. 635]; they are also motivated by the conditions we observed in the mean-square-error case in Example 4. Following the developments in [41, 2, 42], we let

$$R_{s,i}(w_{i-1}) \triangleq \mathbb{E}\left[s_i(w_{i-1})s_i^\mathsf{T}(w_{i-1})|\mathcal{F}_{i-1}\right] \tag{1.17a}$$

denote the conditional second-order moment of the gradient noise process, which generally depends on i. We assume that, in the limit, the covariance matrix tends to

a constant value when evaluated at w^o and is denoted by

$$R_s \triangleq \lim_{i\to\infty} \mathbb{E}\left[s_i(w^o)s_i^{\mathsf{T}}(w^o)|\mathcal{F}_{i-1}\right] \qquad (1.17b)$$

For example, comparing with expression (1.16a) for mean-square-error costs, we have

$$s_i(w^o) = -2u_i^{\mathsf{T}}v(i) \qquad (1.18a)$$
$$R_s = 4\sigma_v^2 R_u \qquad (1.18b)$$

Assumption 2 (**Conditions on gradient noise**). *It is assumed that the first and second-order conditional moments of the gradient noise process satisfy* (1.17b) *and*

$$\mathbb{E}\left[s_i(w_{i-1})|\mathcal{F}_{i-1}\right] = 0 \qquad (1.19a)$$
$$\mathbb{E}\left[\|s_i(w_{i-1})\|^2|\mathcal{F}_{i-1}\right] \le \beta^2 \|\widetilde{w}_{i-1}\|^2 + \sigma_s^2 \qquad (1.19b)$$

almost surely, for some nonnegative scalars β^2 and σ_s^2.

∎

Condition (1.19a) ensures that the approximate gradient vector is unbiased. It follows from conditions (1.19a)–(1.19b) that the gradient noise process itself satisfies:

$$\mathbb{E}\, s_i(w_{i-1}) = 0 \qquad (1.20a)$$
$$\mathbb{E}\, \|s_i(w_{i-1})\|^2 \le \beta^2 \mathbb{E}\|\widetilde{w}_{i-1}\|^2 + \sigma_s^2 \qquad (1.20b)$$

It is straightforward to verify that the gradient noise process (1.16a) in the mean-square-error case satisfies conditions (1.19a)–(1.19b). Note in particular from (1.16e) that we can make the identifications $\sigma_s^2 \to 4\sigma_v^2\mathrm{Tr}(R_u)$ and $\beta^2 \to 4c$.

1.2.5 **RANDOM UPDATES**

We examined the performance of synchronous updates of the form (1.12) in some detail in [1, 5]. As indicated earlier, the focus of the current chapter is on extending the treatment from [1] to asynchronous implementations. Accordingly, the first main digression in the exposition relative to [1] occurs at this stage.

Thus, note that the stochastic-gradient recursion (1.12) employs a constant step-size parameter, $\mu > 0$. This means that this implementation expects the approximate gradient vector, $\widehat{\nabla_{w^{\mathsf{T}}} J}(w_{i-1})$, to be available at every iteration. Nevertheless, there are situations where data may arrive at the agent at random times, in which case the updates will also be occurring at random times. One way to capture this behavior is to model the step-size parameter as a *random* process, which we shall denote by the boldface notation $\mu(i)$ (since boldface letters in our notation refer to random quantities). Doing so, allows us to replace the synchronous implementation (1.12)

12   **CHAPTER 1** Asynchronous Adaptive Networks

by the following asynchronous recursion:

$$w_i = w_{i-1} - \mu(i)\widehat{\nabla_{w^\top} J}(w_{i-1}), \quad i \geq 0 \tag{1.21}$$

Observe that we are attaching a time index to the step-size parameter in order to highlight that its value will now be changing randomly from one iteration to another.

For example, one particular instance discussed further ahead in Example 5 is when $\mu(i)$ is a Bernoulli random variable assuming one of two possible values, say, $\mu(i) = \mu > 0$ with probability p_μ and $\mu(i) = 0$ with probability $1 - p_\mu$. In this case, recursion (1.21) will be updating p_μ−fraction of the time. We will not limit our presentation to the Bernoulli model but will allow the random step-size to assume broader probability distributions; we denote its first and second-order moments as follows.

Assumption 3 (**Conditions on step-size process**). *It is assumed that the stochastic process $\{\mu(i), i \geq 0\}$ consists of a sequence of independent and bounded random variables, $\mu(i) \in [0, \mu_{\text{ub}}]$, where $\mu_{\text{ub}} > 0$ is a constant upper bound. The mean and variance of $\mu(i)$ are fixed over i, and they are denoted by:*

$$\bar{\mu} \stackrel{\Delta}{=} \mathbb{E}\,\mu(i) \tag{1.22a}$$

$$\sigma_\mu^2 \stackrel{\Delta}{=} \mathbb{E}\,(\mu(i) - \bar{\mu})^2 \tag{1.22b}$$

with $\bar{\mu} > 0$ and $\sigma_\mu^2 \geq 0$. Moreover, it is assumed that, for any i, the random variable $\mu(i)$ is independent of any other random variable in the learning algorithm.

■

The following variable will play an important role in characterizing the mean-square-error performance of asynchronous updates and, hence, we introduce a unique symbol for it:

$$\mu_x \stackrel{\Delta}{=} \bar{\mu} + \frac{\sigma_\mu^2}{\bar{\mu}} \tag{1.23}$$

where the subscript "x" is meant to refer to the "asynchronous" mode of operation. This expression captures the first and second-order moments of the random variations in the step-size parameter into a single variable, μ_x. While the constant step-size μ determines the performance of the synchronous implementation (1.12), it turns out that the constant variable μ_x defined above will play an equivalent role for the asynchronous implementation (1.21). Note further that the synchronous stochastic-gradient iteration (1.12) can be viewed as a special case of recursion (1.21) when the variance of $\mu(i)$ is set to zero, i.e., $\sigma_\mu^2 = 0$, and the mean value $\bar{\mu}$ is set to μ. Therefore, by using these substitutions, we will able to deduce performance metrics for (1.12) from the performance metrics that we shall present for (1.21). The following

two examples illustrate situations involving random updates.

Example 5 (**Random updates under a Bernoulli model**). Assume that at every iteration i, the agent adopts a random "on-off" policy to reduce energy consumption. It tosses a coin to decide whether to enter an active learning mode or a sleeping mode. Let $0 < p_\mu < 1$ denote the probability of entering the active mode. During the active mode, the agent employs a step-size value μ. This model is useful for situations in which data arrives randomly at the agent: at every iteration i, new data is available with probability p_μ. The random step-size process is therefore of the following type:

$$\mu(i) = \begin{cases} \mu, & \text{with probability } p_\mu \\ 0, & \text{with probability } 1 - p_\mu \end{cases} \tag{1.24a}$$

In this case, the mean and variance of $\mu(i)$ are given by:

$$\bar{\mu} = p_\mu \mu, \qquad \sigma_\mu^2 = p_\mu(1 - p_\mu)\mu^2, \qquad \mu_x = \mu \tag{1.24b}$$

∎

Example 6 (**Random updates under a Beta model**). Since the random step-size $\mu(i)$ is limited to a finite-length interval $[0, \mu_{ub}]$, we may extend the Bernoulli model from the previous example by adopting a more general continuous Beta distribution for $\mu(i)$. The Beta distribution is an extension of the Bernoulli distribution. While the Bernoulli distribution assumes two discrete possibilities for the random variable, say, $\{0, \mu\}$, the Beta distribution allows for any value in the continuum $[0, \mu]$.

Thus, let x denote a generic scalar random variable that assumes values in the interval $[0, 1]$ according to a Beta distribution. Then, according to this distribution, the pdf of x, denoted by $f_x(x; \xi, \zeta)$, is determined by two shape parameters $\{\xi, \zeta\}$ as follows [45, 46]:

$$f_x(x; \xi, \zeta) = \begin{cases} \dfrac{\Gamma(\xi + \zeta)}{\Gamma(\xi)\Gamma(\zeta)} x^{\xi-1}(1 - x)^{\zeta-1}, & 0 \leq x \leq 1 \\ 0, & \text{otherwise} \end{cases} \tag{1.25a}$$

where $\Gamma(\cdot)$ denotes the Gamma function [47, 48]. Figure 1.1 plots $f_x(x; \xi, \zeta)$ for two values of ζ. The mean and variance of the Beta distribution (1.25a) are given by:

$$\bar{x} = \frac{\xi}{\xi + \zeta}, \qquad \sigma_x^2 = \frac{\xi\zeta}{(\xi + \zeta)^2(\xi + \zeta + 1)} \tag{1.25b}$$

We note that the classical uniform distribution over the interval $[0, 1]$ is a special case of the Beta distribution for $\xi = \zeta = 1$ — see Fig. 1.1. Likewise, the Bernoulli distribution with $p_\mu = 1/2$ is recovered from the Beta distribution by letting $\xi = \zeta \to 0$.

In the Beta model for asynchronous adaptation, we assume that the ratio $\mu(i)/\mu_{ub}$ follows a Beta distribution with parameters $\{\xi, \zeta\}$. Under this model, the mean and variance of the random step-size become:

$$\bar{\mu} = \left(\frac{\xi}{\xi + \zeta}\right)\mu_{ub}, \qquad \sigma_\mu^2 = \left(\frac{\xi\zeta}{(\xi + \zeta)^2(\xi + \zeta + 1)}\right)\mu_{ub}^2 \tag{1.26}$$

∎

1.2.6 **MEAN-SQUARE-ERROR STABILITY**

We now examine the convergence of the asynchronous stochastic-gradient recursion (1.21). In the statement below, the notation $a = O(\mu)$ means $a \leq b\mu$ for some constant b that is independent of μ.

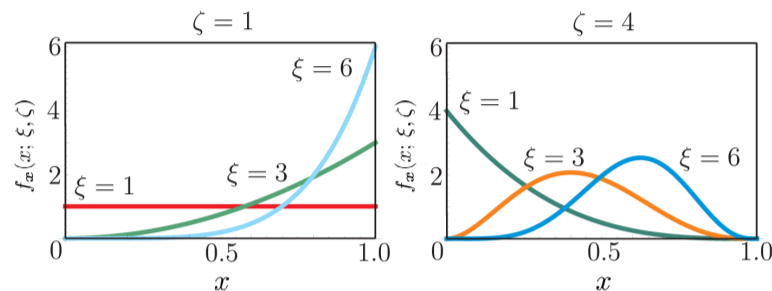**14   CHAPTER 1** Asynchronous Adaptive Networks

FIGURE 1.1   Beta distribution.

The pdf of the Beta distribution, $f_x(x; \xi, \zeta)$, defined by (1.25a) for different values of the shape parameters ξ and ζ. *Source: Figure extracted with permission from [2].*

Lemma 1 (Mean-square-error stability). *Assume the conditions under Assumptions 1, 2, and 3 on the cost function, the gradient noise process, and the random step-size process hold. Let $\mu_o = 2\nu/(\delta^2 + \beta^2)$. For any μ_x satisfying*

$$\mu_x < \mu_o \tag{1.27}$$

it holds that $\mathbb{E}\|\widetilde{w}_i\|^2$ converges exponentially (i.e., at a geometric rate) according to the recursion

$$\mathbb{E}\|\widetilde{w}_i\|^2 \;\leq\; \alpha\,\mathbb{E}\|\widetilde{w}_{i-1}\|^2 + (\bar{\mu}^2 + \sigma_\mu^2)\sigma_s^2 \tag{1.28a}$$

where the scalar α satisfies $0 \leq \alpha < 1$ and is given by

$$
\begin{aligned}
\alpha \;&\overset{\Delta}{=}\; 1 - 2\nu\bar{\mu} + (\delta^2 + \beta^2)(\bar{\mu}^2 + \sigma_\mu^2) \\
&=\; 1 - 2\nu\bar{\mu} + O(\mu_x^2)
\end{aligned}
\tag{1.28b}
$$

It follows from (1.28a) that, for sufficiently small step-sizes:

$$\limsup_{i \to \infty}\; \mathbb{E}\|\widetilde{w}_i\|^2 = O(\mu_x) \tag{1.28c}$$

Proof. We subtract w^o from both sides of (1.21) to get

$$\widetilde{w}_i = \widetilde{w}_{i-1} + \mu(i)\,\nabla_{w^{\mathsf{T}}} J(w_{i-1}) + \mu(i)\,s_i(w_{i-1}) \tag{1.29a}$$

We now appeal to the mean-value theorem [5, 35, 49] to write:

$$\nabla_{w^{\mathsf{T}}} J(w_{i-1}) = -\left[\int_0^1 \nabla_w^2 J(w_o - t\widetilde{w}_{i-1})dt\right]\widetilde{w}_{i-1} \;\overset{\Delta}{=}\; -H_{i-1}\widetilde{w}_{i-1} \tag{1.29b}$$

where we are introducing the *symmetric* and *random* time-variant matrix H_{i-1} to represent the integral

expression. Substituting into (1.29a), we get

$$\widetilde{\boldsymbol{w}}_i = [I_M - \boldsymbol{\mu}(i)\boldsymbol{H}_{i-1}]\,\widetilde{\boldsymbol{w}}_{i-1} + \boldsymbol{\mu}(i)\,\boldsymbol{s}_i(\boldsymbol{w}_{i-1}) \tag{1.29c}$$

so that from Assumption 2:

$$\mathbb{E}\left[\|\widetilde{\boldsymbol{w}}_i\|^2|\mathcal{F}_{i-1}\right] \;\leq\; \left(\mathbb{E}\left[\|I_M - \boldsymbol{\mu}(i)\,\boldsymbol{H}_{i-1}\|^2|\mathcal{F}_{i-1}\right]\right)\|\widetilde{\boldsymbol{w}}_{i-1}\|^2 + \left(\mathbb{E}\,\boldsymbol{\mu}^2(i)\right)\left(\mathbb{E}\left[\|\boldsymbol{s}_i(\boldsymbol{w}_{i-1})\|^2|\mathcal{F}_{i-1}\right]\right) \tag{1.30}$$

It follows from (1.9) that

$$\|I_M - \boldsymbol{\mu}(i)\,\boldsymbol{H}_{i-1}\|^2 = [\rho\,(I_M - \boldsymbol{\mu}(i)\,\boldsymbol{H}_{i-1})]^2$$
$$\leq \max\left\{[1 - \boldsymbol{\mu}(i)\,\delta]^2,\; [1 - \boldsymbol{\mu}(i)\,\nu]^2\right\}$$
$$\leq 1 - 2\boldsymbol{\mu}(i)\,\nu + \boldsymbol{\mu}^2(i)\delta^2 \tag{1.31a}$$

since $\nu \leq \delta$. In the first line above, the notation $\rho(A)$ denotes the spectral radius of its matrix argument (i.e., $\rho(A) = \max_k |\lambda_k(A)|$ in terms of the largest magnitude eigenvalue of A). From (1.31a) we obtain

$$\mathbb{E}\left(\|I_M - \boldsymbol{\mu}(i)\,\boldsymbol{H}_{i-1}\|^2|\mathcal{F}_{i-1}\right) \leq 1 - 2\bar{\mu}\nu + (\bar{\mu}^2 + \sigma_\mu^2)\delta^2 \tag{1.31b}$$

Taking expectations of both sides of (1.30), we arrive at (1.28a) from (1.20b) and (1.31b) with α given by (1.28b). The bound in (1.27) on the moments of the random step-size ensures that $0 \leq \alpha < 1$. For the $O(\mu_x^2)$ approximation in expression (1.28b) note from (1.23) that

$$(\bar{\mu}^2 + \sigma_\mu^2) \;=\; \bar{\mu}\mu_x \;\leq\; \mu_x^2 \tag{1.32}$$

Iterating recursion (1.28a) gives

$$\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_i\|^2 \leq \alpha^{i+1}\,\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_{-1}\|^2 + \frac{(\bar{\mu}^2 + \sigma_\mu^2)\sigma_s^2}{1 - \alpha} \tag{1.33a}$$

Since $0 \leq \alpha < 1$, there exists an iteration value I_o large enough such that

$$\alpha^{i+1}\,\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_{-1}\|^2 \;\leq\; \frac{(\bar{\mu}^2 + \sigma_\mu^2)\sigma_s^2}{1 - \alpha}, \quad i > I_o \tag{1.33b}$$

It follows that the variance $\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_i\|^2$ converges exponentially to a region that is upper bounded by $2(\bar{\mu}^2 + \sigma_\mu^2)\sigma_s^2/(1-\alpha)$. It can be verified that this bound does not exceed $2\mu_x\sigma_s^2/\nu$, which is $O(\mu_x)$, for any $\mu_x < \mu_o/2$.

<div align="right">□</div>

1.2.7 **MEAN-SQUARE-ERROR PERFORMANCE**

We conclude from (1.28c) that the mean-square error (MSE) can be made as small as desired by using small step-sizes, μ_x. In this section we derive a closed-form expression for the asymptotic mean-square error, which is more frequently called the *mean-square deviation* (MSD) and is defined as:

$$\boxed{\mathrm{MSD} \;\stackrel{\Delta}{=}\; \lim_{i \to \infty} \mathbb{E}\,\|\widetilde{\boldsymbol{w}}_i\|^2} \tag{1.34a}$$

16   **CHAPTER 1** Asynchronous Adaptive Networks

Strictly speaking, the limit on the right-hand side of the above expression may not exist. A more accurate definition for the MSD appears in Eq. (4.86) of [5], namely,

$$\text{MSD} \overset{\Delta}{=} \mu_x \cdot \left(\lim_{\mu_x \to 0} \limsup_{i \to \infty} \frac{1}{\mu_x} \mathbb{E} \|\widetilde{w}_i\|^2 \right) \qquad (1.34b)$$

However, it was explained in [5][Sec. 4.5] that derivations that assume the validity of (1.34a) still lead to the same expression for the MSD to first-order in μ_x as derivations that rely on the more formal definition (1.34b). We therefore continue with (1.34a) for simplicity of presentation. We explain below how an expression for the MSD can be obtained by following the energy conservation technique of [5, 29, 28, 50, 51]. For that purpose, we need to introduce two smoothness conditions.

Assumption 4 (**Smoothness conditions**). *In addition to Assumptions 1 and 2, we assume that the Hessian matrix of the cost function and the noise covariance matrix defined by* (1.17a) *are locally Lipschitz continuous in a small neighborhood around* $w = w^o$:

$$\left\| \nabla_w^2 J(w^o + \delta w) - \nabla_w^2 J(w^o) \right\| \leq \tau \|\delta w\| \qquad (1.35a)$$

$$\left\| R_{s,i}(w^o + \delta w) - R_{s,i}(w^o) \right\| \leq \tau_2 \|\delta w\|^\kappa \qquad (1.35b)$$

for small perturbations $\|\delta w\| \leq r$ *and for some* $\tau, \tau_2 \geq 0$ *and* $1 \leq \kappa \leq 2$.

∎

The range of values for κ can be enlarged, e.g., to $\kappa \in (0, 4]$. The only change in allowing a wider range for κ is that the exponent of the higher-order term, $O(\mu_x^{3/2})$, that will appear in several performance expressions, as is the case with (1.41a)–(1.41b), will need to be adjusted from $\frac{3}{2}$ to $\min\{\frac{3}{2}, 1 + \frac{\kappa}{2}\}$, without affecting the first-order term that determines the MSD [3, 5, 52]. Therefore, it is sufficient to continue with $\kappa \in [1, 2]$ to illustrate the key concepts though the MSD expressions will still be valid to first-order in μ_x.

Using (1.9), it can be verified that condition (1.35a) translates into a global Lipschitz property relative to the minimizer w^o, i.e., it will also hold that [5, 3]:

$$\|\nabla_w^2 J(w) - \nabla_w^2 J(w^o)\| \leq \tau' \|w - w^o\| \qquad (1.35c)$$

for all w and for some $\tau' \geq 0$. For example, both conditions (1.35a)–(1.35b) are readily satisfied by mean-square-error costs. Using property (1.35c), we can now motivate a useful long-term model for the evolution of the error vector $\widetilde{w}_i$ after sufficient iterations, i.e., for $i \gg 1$. Indeed, let us reconsider recursion (1.29c) and introduce the deviation matrix:

$$\widetilde{H}_{i-1} \overset{\Delta}{=} H - H_{i-1} \qquad (1.36a)$$

where the constant (symmetric and positive-definite) matrix H is defined as:

$$H \triangleq \nabla_w^2 J(w^o) \tag{1.36b}$$

Substituting (1.36a) into (1.29c) gives

$$\widetilde{w}_i = (I_M - \mu(i) H) \widetilde{w}_{i-1} + \mu(i) s_i(w_{i-1}) + \mu(i) c_{i-1} \tag{1.37a}$$

where

$$c_{i-1} \triangleq \widetilde{H}_{i-1} \widetilde{w}_{i-1} \tag{1.37b}$$

Using (1.35c) and the fact that $(\mathbb{E} a)^2 \leq \mathbb{E} a^2$ for any real-valued random variable, we can bound the conditional expectation of the norm of the perturbation term as follows:

$$\mathbb{E}\left[\|\mu(i) c_{i-1}\| \,|\, \mathcal{F}_{i-1} \right] = (\mathbb{E} \mu(i)) \, (\mathbb{E}\left[\|c_{i-1}\| \,|\, \mathcal{F}_{i-1} \right])$$

$$\leq \sqrt{(\mathbb{E} \mu^2(i))} \, \mathbb{E}\left[\|c_{i-1}\| \,|\, \mathcal{F}_{i-1} \right]$$

$$\overset{(1.35c)}{\leq} \sqrt{\bar{\mu}^2 + \sigma_\mu^2} \cdot \frac{\tau'}{2} \|\widetilde{w}_{i-1}\|^2$$

$$\leq \frac{\bar{\mu}^2 + \sigma_\mu^2}{\bar{\mu}} \cdot \frac{\tau'}{2} \|\widetilde{w}_{i-1}\|^2$$

$$= \mu_x \cdot \frac{\tau'}{2} \|\widetilde{w}_{i-1}\|^2 \tag{1.38a}$$

so that using (1.28c), we conclude that:

$$\limsup_{i \to \infty} \mathbb{E} \|\mu(i) c_{i-1}\| = O(\mu_x^2) \tag{1.38b}$$

We can deduce from this result that $\|\mu(i) c_{i-1}\| = O(\mu_x^2)$ asymptotically with *high probability* [5, 3]. To see this, let $r_c = m\mu_x^2$, for any constant integer $m \geq 1$. Now, calling upon Markov's inequality [53, 54, 55], we conclude from (1.38b) that for $i \gg 1$:

$$\text{Pr}\left(\|\mu(i) c_{i-1}\| < r_c\right) = 1 - \text{Pr}\left(\|\mu(i) c_{i-1}\| \geq r_c\right)$$

$$\geq 1 - \frac{\mathbb{E} \|\mu(i) c_{i-1}\|}{r_c}$$

$$\overset{(1.38b)}{\geq} 1 - O(1/m) \tag{1.38c}$$

This result shows that the probability of having $\|\mu(i) c_{i-1}\|$ bounded by r_c can be made arbitrarily close to one by selecting a large enough value for m. Once the value for m has been fixed to meet a desired confidence level, then $r_c = O(\mu_x^2)$. This analysis, along with recursion (1.37a), motivate us to assess the mean-square performance of the error recursion (1.29c) by considering instead the following long-term model,

18   **CHAPTER 1** Asynchronous Adaptive Networks

which holds with high probability after sufficient iterations $i \gg 1$:

$$\widetilde{\boldsymbol{w}}_i = (I_M - \boldsymbol{\mu}(i)\,H)\widetilde{\boldsymbol{w}}_{i-1} + \boldsymbol{\mu}(i)\,\boldsymbol{s}_i(\boldsymbol{w}_{i-1}) + O(\mu_x^2) \tag{1.39}$$

Working with iteration (1.39) is helpful because its dynamics is driven by the constant matrix H as opposed to the random matrix $\boldsymbol{H}_{i-1}$ in the original error recursion (1.29c). If desired, it can be shown that, under some technical conditions on the fourth-order moment of the gradient noise process, the MSD expression that will result from using (1.39) is within $O(\mu_x^{3/2})$ of the actual MSD expression for the original recursion (1.29c) — see [5, 52, 3] for a formal proof of this fact. Therefore, it is sufficient to rely on the long-term model (1.39) to obtain performance expressions that are accurate to first-order in μ_x. Figure 1.2 provides a block-diagram representation for (1.39).
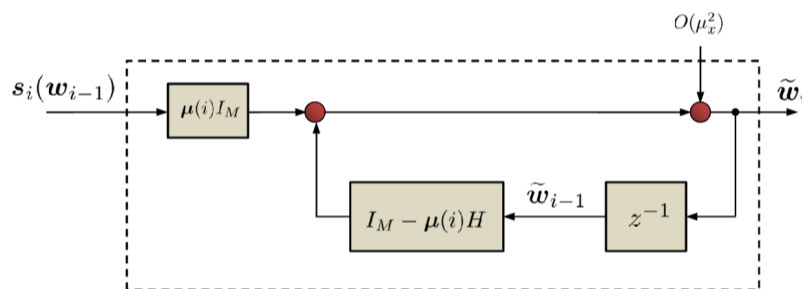

FIGURE 1.2   Long-term dynamics.
A block-diagram representation of the long-term recursion (1.39) for single-agent adaptation and learning. *Source: Figure extracted with permission from [1].*

Before explaining how model (1.39) can be used to assess the MSD, we remark that there is a second useful metric for evaluating the performance of stochastic gradient algorithms. This metric relates to the mean excess-cost; which is also called the *excess-risk* (ER) in the machine learning literature [31, 32] and the excess-mean-square-error (EMSE) in the adaptive filtering literature [26, 27, 28]. We denote it by the letters ER and define it as the average fluctuation of the cost function around its minimum value:

$$\text{ER} \triangleq \lim_{i \to \infty} \mathbb{E}\,\{J(\boldsymbol{w}_{i-1}) - J(w^o)\} \tag{1.40a}$$

Using the smoothness condition (1.35a), and the mean-value theorem [35, 49] again,

it can be verified that [3, 5, 52]:

$$\text{ER} \overset{\Delta}{=} \lim_{i \to \infty} \mathbb{E} \, \|\widetilde{\boldsymbol{w}}_{i-1}\|^2_{\frac{1}{2}H} + O(\mu_x^{3/2}) \tag{1.40b}$$

Lemma 2 (**Mean-square-error performance**). *Assume the conditions under Assumptions 1, 2, 3, and 4 on the cost function, the gradient noise process, and the random step-size process hold. Assume further that the asynchronous step-size parameter μ_x is sufficiently small to ensure mean-square stability as required by* (1.27). *Then, the* MSD *and* ER *metrics for the asynchronous stochastic-gradient algorithm* (1.21) *are well-approximated to first-order in μ_x by the expressions:*

$$\text{MSD}^{\text{asyn}} = \frac{\mu_x}{2} \text{Tr} \left(H^{-1} R_s \right) + O(\mu_x^{3/2}) \tag{1.41a}$$

$$\text{ER}^{\text{asyn}} = \frac{\mu_x}{4} \text{Tr} \left(R_s \right) + O(\mu_x^{3/2}) \tag{1.41b}$$

where R_s and H are defined by (1.17b) *and* (1.36b), *and where we are adding the superscript "asyn" for clarity in order to distinguish these measures from the corresponding measures in the synchronous case (mentioned below in (1.43a)–(1.43c)). Moreover, we derived earlier in* (1.28b) *the following expression for the convergence rate:*

$$\alpha^{\text{asyn}} = 1 - 2\nu\bar{\mu} + O(\mu_x^2) \tag{1.41c}$$

Proof. We introduce the eigen-decomposition $H = U\Lambda U^{\mathsf{T}}$, where U is orthonormal and Λ is diagonal with positive entries, and rewrite (1.39) in terms of transformed quantities:

$$\overline{\boldsymbol{w}}_i = (I_M - \boldsymbol{\mu}(i)\,\Lambda)\,\overline{\boldsymbol{w}}_{i-1} + \boldsymbol{\mu}(i)\,\overline{\boldsymbol{s}}_i(\boldsymbol{w}_{i-1}) + O(\mu_x^2) \tag{1.42a}$$

where $\overline{\boldsymbol{w}}_i = U^{\mathsf{T}}\widetilde{\boldsymbol{w}}_i$ and $\overline{\boldsymbol{s}}_i(\boldsymbol{w}_{i-1}) = U^{\mathsf{T}}\boldsymbol{s}_i(\boldsymbol{w}_{i-1})$. Let Σ denote an arbitrary $M \times M$ diagonal matrix with positive entries that we are free to choose. Then, equating the weighted squared norms of both sides of (1.42a) and taking expectations gives for $i \gg 1$:

$$\mathbb{E} \, \|\overline{\boldsymbol{w}}_i\|^2_{\Sigma} = \mathbb{E} \, \|\overline{\boldsymbol{w}}_{i-1}\|^2_{\Sigma'} + (\bar{\mu}^2 + \sigma_\mu^2)\,\mathbb{E} \, \|\overline{\boldsymbol{s}}_i(\boldsymbol{w}_{i-1})\|^2_{\Sigma} + O(\mu_x^{5/2}) \tag{1.42b}$$

where

$$\Sigma' \overset{\Delta}{=} \mathbb{E}\,(I_M - \boldsymbol{\mu}(i)\,\Lambda)\Sigma(I_M - \boldsymbol{\mu}(i)\,\Lambda) = \Sigma - 2\bar{\mu}\Lambda\Sigma + O(\mu_x^2) \tag{1.42c}$$

From (1.17b), (1.19b), (1.28c), and (1.35b) we obtain:

$$\lim_{i \to \infty} \mathbb{E} \, \|\overline{\boldsymbol{s}}_i(\boldsymbol{w}_{i-1})\|^2_{\Sigma} = \text{Tr}(U\Sigma U^{\mathsf{T}} R_s) + O(\mu_x^{\kappa/2}) \tag{1.42d}$$

Therefore, substituting into (1.42b) gives for $i \to \infty$:

$$\lim_{i \to \infty} \mathbb{E} \, \|\overline{\boldsymbol{w}}_i\|^2_{2\Lambda\Sigma} = \mu_x \text{Tr}(U\Sigma U^{\mathsf{T}} R_s) + O(\mu_x^{3/2}) \tag{1.42e}$$

Since we are free to choose Σ, we let $\Sigma = \frac{1}{2}\Lambda^{-1}$ and arrive at (1.41a) since $\|\overline{\boldsymbol{w}}_i\|^2 = \|\widetilde{\boldsymbol{w}}_i\|^2$ and $U\Sigma U^{\mathsf{T}} = \frac{1}{2}H^{-1}$. On the other hand, selecting $\Sigma = \frac{1}{4}I_M$ leads to (1.41b). $\square$

We recall our earlier remark that the synchronous stochastic-gradient recursion

20   **CHAPTER 1** Asynchronous Adaptive Networks

(1.12) can be viewed as a special case of the asynchronous update (1.21) by setting $\sigma_\mu^2 = 0$ and $\mu_x = \bar{\mu} \equiv \mu$. Substituting these values into (1.41a)–(1.41c), we obtain for the synchronous implementation (1.12):

$$\text{MSD}^{\text{sync}} = \frac{\mu}{2} \, \text{Tr} \left(H^{-1} R_s \right) + O(\mu^{3/2}) \tag{1.43a}$$

$$\text{ER}^{\text{sync}} = \frac{\mu}{2} \, \text{Tr} \left(R_s \right) + O(\mu^{3/2}) \tag{1.43b}$$

$$\alpha^{\text{sync}} = 1 - 2\nu\mu + O(\mu^2) \tag{1.43c}$$

which are the same expressions presented in [1] and which agree with classical results for LMS adaptation [56]–[61]. The matrices R_s that appear in these expressions, and in (1.41a)–(1.41c), were defined earlier in (1.17b) and they correspond to the co-variance matrices of the gradient noise processes in their respective (synchronous or asynchronous) implementations. The examples that follow show how expressions (1.41a) and (1.41b) can be used to recover performance metrics for mean-square-error adaptation and learning under random updates.

Example 7 (**Performance of asynchronous LMS adaptation**). We reconsider the LMS recursion (1.13f) albeit in an asynchronous mode of operation, namely,

$$\boxed{w_i = w_{i-1} + 2\mu(i) u_i^\mathsf{T} \left[d(i) - u_i w_{i-1} \right], \quad i \geq 0} \tag{1.44}$$

We know from Example 3 and (1.18a) that this situation corresponds to $H = 2R_u$ and $R_s = 4\sigma_v^2 R_u$. Substituting into (1.41a) and (1.41b) leads to the following expressions for the MSD and EMSE of the asynchronous LMS filter:

$$\text{MSD}_{\text{LMS}}^{\text{asyn}} \approx \mu_x M \sigma_v^2 = O(\mu_x) \tag{1.45a}$$

$$\text{EMSE}_{\text{LMS}}^{\text{asyn}} \approx \mu_x \sigma_v^2 \text{Tr}(R_u) = O(\mu_x) \tag{1.45b}$$

where here, and elsewhere, we will be using the notation $\approx$ to indicate that we are ignoring higher-order terms in μ_x. For example, let us assume a Bernoulli update model for $\mu(i)$ where the filter updates with probability p_μ using a step-size value μ or stays inactive otherwise. In this case, we conclude from (1.24b) that $\mu_x = \mu$ so that the above performance expressions for the MSD and EMSE metrics will coincide with the values obtained in the synchronous case as well. In other words, the steady-state performance levels are not affected whether the algorithm learns in a synchronous or asynchronous manner. However, the convergence rate is affected since $\bar{\mu} = \mu p_\mu$ and, therefore,

$$\alpha_{\text{LMS}}^{\text{sync}} \approx 1 - 2\nu\mu \tag{1.46a}$$

$$\alpha_{\text{LMS}}^{\text{asyn}} \approx 1 - 2\nu\mu p_\mu > \alpha_{\text{LMS}}^{\text{sync}} \tag{1.46b}$$

It follows that asynchronous LMS adaptation attains the same performance levels as synchronous LMS adaptation albeit at a slower convergence rate.

We may alternatively compare the performance of the synchronous and asynchronous implementations by fixing their convergence rates to the same value. Thus, consider now a second random update scheme with mean $\bar{\mu}$ and let us set $\mu = \bar{\mu}$. That is, the step-size used by the synchronous implementation is set equal to the mean step-size used by the asynchronous implementation. Then, in this case, we will get $\alpha_{\text{LMS}}^{\text{sync}} = \alpha_{\text{LMS}}^{\text{asyn}}$ so that the convergence rates coincide to first-order. However, it now holds that $\text{MSD}_{\text{LMS}}^{\text{asyn}} > \text{MSD}_{\text{LMS}}^{\text{sync}}$ since $\bar{\mu}_x > \mu$ so that some deterioration in MSD performance occurs.

∎

Example 8 (**Performance of asynchronous online learners**). Consider a stand-alone learner receiving a streaming sequence of independent data vectors $\{x_i, i \geq 0\}$ that arise from some fixed probability distribution $\mathcal{X}$. The goal is to learn the vector w^o that optimizes some v-strongly convex risk function $J(w)$ defined in terms of a loss function [62, 63]:

$$w^o \overset{\Delta}{=} \operatorname{argmin}_w J(w) = \operatorname{argmin}_w \mathbb{E}\, Q(w; x_i) \tag{1.47a}$$

In an asynchronous environment, the learner seeks w^o by running the stochastic-gradient algorithm with random step-sizes:

$$\boxed{\boldsymbol{w}_i = \boldsymbol{w}_{i-1} - \boldsymbol{\mu}(i)\, \nabla_{w^\mathsf{T}} Q(\boldsymbol{w}_{i-1}; x_i), \quad i \geq 0} \tag{1.47b}$$

The gradient noise vector is still given by

$$\boldsymbol{s}_i(\boldsymbol{w}_{i-1}) = \nabla_{w^\mathsf{T}}\, Q(\boldsymbol{w}_{i-1}; x_i) - \nabla_{w^\mathsf{T}}\, J(\boldsymbol{w}_{i-1}) \tag{1.47c}$$

Since $\nabla_w J(w^o) = 0$, and since the distribution of x_i is stationary, it follows that the covariance matrix of $\boldsymbol{s}_i(w^o)$ is constant and given by

$$R_s = \mathbb{E}\, \nabla_{w^\mathsf{T}}\, Q(w^o; x_i) \nabla_w\, Q(w^o; x_i) \tag{1.47d}$$

The excess-risk measure that will result from this stochastic implementation is then given by (1.41b) so that $\mathrm{ER} = O(\mu_x)$.

∎

1.3 **CENTRALIZED ADAPTATION AND LEARNING**

The discussion in the previous section establishes the mean-square stability of *stand-alone* adaptive agents for small step-sizes (Lemma 1), and provides expressions for their MSD and ER metrics (Lemma 2). We now examine two situations involving a multitude of similar agents. In the first scenario, each agent senses data and analyzes it independently of the other agents. We refer to this mode of operation as non-cooperative processing. In the second scenario, the agents transmit the collected data for processing at a fusion center. We refer to this mode of operation as centralized or batch processing. We motivate the discussion by considering first the case of mean-square-error costs. Subsequently, we extend the results to more general costs.

1.3.1 **NON-COOPERATIVE MSE PROCESSING**

Thus, consider *separate* agents, labeled $k = 1, 2, \ldots, N$. Each agent, k, receives streaming data

$$\{\boldsymbol{d}_k(i), \boldsymbol{u}_{k,i}; i \geq 0\} \tag{1.48a}$$

where we are using the subscript k to index the data at agent k. We assume that the data at each agent satisfies the same statistical properties as in Example 3, and the same linear regression model (1.13a) with a common w^o albeit with noise $\boldsymbol{v}_k(i)$. We

22 **CHAPTER 1** Asynchronous Adaptive Networks

denote the statistical moments of the data at agent k by

$$R_{u,k} = \mathbb{E}\,\boldsymbol{u}_{k,i}^{\mathsf{T}}\boldsymbol{u}_{k,i} > 0, \quad \sigma_{v,k}^2 = \mathbb{E}\,\boldsymbol{v}_k^2(i) \tag{1.48b}$$

We further assume in this motivating example that the $R_{u,k}$ are uniform across the agents so that

$$R_{u,k} \equiv R_u, \quad k = 1, 2, \ldots, N \tag{1.48c}$$

In this way, the cost $J_k(w) = \mathbb{E}\,(\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}w)^2$, which is associated with agent k, will satisfy a condition similar to (1.9) with the corresponding parameters $\{v, \delta\}$ given by (cf. (1.10a)):

$$\nu = 2\lambda_{\min}(R_u), \quad \delta = 2\lambda_{\max}(R_u) \tag{1.49a}$$

Now, if each agent runs the asynchronous LMS learning rule (1.44) to estimate w^o on its own then, according to (1.45a), each agent k will attain an individual MSD level that is given by

$$\boxed{\mathrm{MSD}_{\mathrm{ncop},k}^{\mathrm{asyn}} \approx \mu_x M \sigma_{v,k}^2, \quad k = 1, 2, \ldots, N} \tag{1.49b}$$

where we are further assuming that the parameter μ_x is uniform across the agents to enable a meaningful comparison. Moreover, according to (1.28b), agent k will converge towards this level at a rate dictated by:

$$\boxed{\alpha_{\mathrm{ncop},k}^{\mathrm{asyn}} \approx 1 - 4\mu_x\lambda_{\min}(R_u)} \tag{1.49c}$$

The subscript "ncop" is used in (1.49b) and (1.49c) to indicate that these expressions are for the non-cooperative mode of operation. It is seen from (1.49b) that agents with noisier data (i.e., larger $\sigma_{v,k}^2$) will perform worse and have larger MSD levels than agents with cleaner data. We are going to show in later sections that cooperation among the agents, whereby agents share information with their neighbors, can help enhance their individual performance levels.

1.3.2 CENTRALIZED MSE PROCESSING

Let us now contrast the above non-cooperative solution with a centralized implementation whereby, at every iteration i, the N agents transmit their raw data $\{\boldsymbol{d}_k(i), \boldsymbol{u}_{k,i}\}$ to a fusion center for processing. In a *synchronous* environment, once the fusion center receives the raw data, it can run a standard stochastic-gradient update of the form:

$$\boldsymbol{w}_i = \boldsymbol{w}_{i-1} + \mu\left[\frac{1}{N}\sum_{k=1}^{N}2\boldsymbol{u}_{k,i}^{\mathsf{T}}(\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}\boldsymbol{w}_{i-1})\right] \tag{1.50a}$$

where μ is the constant step-size, and the term multiplying μ can be seen to correspond to a sample average of several approximate gradient vectors. The analysis in

[1, 5] showed that the MSD performance that results from this implementation is given by (using expression (1.64a) with $H_k = 2R_u$ and $R_{s,k} = 4\sigma_{v,k}^2 R_u$):

$$\text{MSD}_{\text{cent}}^{\text{sync}} \approx \frac{\mu M}{N} \left(\frac{1}{N} \sum_{k=1}^{N} \sigma_{v,k}^2 \right) \qquad (1.50b)$$

Moreover, using expression (1.64b) given further ahead, this centralized solution will converge towards the above MSD level at the same rate as the non-cooperative solution:

$$\alpha_{\text{cent}}^{\text{sync}} \approx 1 - 4\mu\lambda_{\min}(R_u) \qquad (1.50c)$$

In an asynchronous environment, there are now *several* random events that can interfere with the operation of the fusion center. Let us consider initially one particular random event that corresponds to the situation in which the fusion center may or may not update at any particular iteration (e.g., due to some power-saving strategy). In a manner similar to (1.21), we may represent this scenario by writing:

$$w_i = w_{i-1} + \mu(i) \left[\frac{1}{N} \sum_{k=1}^{N} 2u_{k,i}^\mathsf{T} (d_k(i) - u_{k,i} w_{i-1}) \right] \qquad (1.51a)$$

with a random step-size process, $\mu(i)$; this process is again assumed to satisfy the conditions under Assumption 3. The analysis in the sequel will show that the MSD performance that results from this implementation is given by:

$$\text{MSD}_{\text{cent}}^{\text{asyn,1}} \approx \frac{\mu_x M}{N} \left(\frac{1}{N} \sum_{k=1}^{N} \sigma_{v,k}^2 \right) \qquad (1.51b)$$

where we are adding the superscript "1" to indicate that this is a preliminary result pertaining to the particular asynchronous implementation (1.51a). We will be generalizing this result soon, at which point we will drop the superscript "1". Likewise, using expression (1.64b) given further ahead, this version of the asynchronous centralized solution converges towards the above MSD level at the same rate as the non-cooperative solution (1.49c) and the synchronous version (1.50c):

$$\alpha_{\text{cent}}^{\text{asyn,1}} \approx 1 - 4\mu_x \lambda_{\min}(R_u) \qquad (1.51c)$$

Observe from (1.50b) and (1.51b) that the MSD level attained by the centralized solution is proportional to $1/N$ times the *average* noise power across all agents. This scaled average noise power can be larger than some of the individual noise variances and smaller than the remaining noise variances. This example shows that it does not generally hold that centralized stochastic-gradient implementations outperform all individual non-cooperative agents [64].

Now, more generally, observe that in the synchronous batch processing case

24   **CHAPTER 1** Asynchronous Adaptive Networks

(1.50a), as well as in the asynchronous implementation (1.51a), the data collected from the various agents are equally aggregated with a weighting factor equal to $1/N$. We can incorporate a second type of random events besides the random variations in the step-size parameter. This second source of uncertainty involves the possibility of failure (or weakening) in the links connecting the individual agents to the fusion center (e.g., due to fading or outage, or perhaps due to the agents themselves deciding to enter into a sleep mode according to some power-saving policy). We can capture these possibilities by extending formulation (1.51a) in the following manner:

$$
\boldsymbol{w}_i = \boldsymbol{w}_{i-1} + \boldsymbol{\mu}(i)\left[\sum_{k=1}^{N} 2\boldsymbol{\pi}_k(i)\boldsymbol{u}_{k,i}^{\mathsf{T}}(\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}\boldsymbol{w}_{i-1})\right] \tag{1.52a}
$$

where $\boldsymbol{\mu}(i)$ continues to be a random step-size process, but now the coefficients $\{\boldsymbol{\pi}_k(i); k = 1, 2, \dots, N\}$ are new random fusion coefficients that satisfy:

$$
\sum_{k=1}^{N} \boldsymbol{\pi}_k(i) = 1, \quad \boldsymbol{\pi}_k(i) \geq 0 \tag{1.52b}
$$

for every $i \geq 0$.

Assumption 5 (Conditions on random fusion coefficients). *It is assumed that $\boldsymbol{\pi}_k(i)$ is independent of $\boldsymbol{\mu}(i)$ and of other random variables in the learning algorithm. The fusion coefficients are also independent over time, namely, $\boldsymbol{\pi}_k(i)$ and $\boldsymbol{\pi}_\ell(j)$ are independent for any $i \neq j$. For the same time i, the coefficients $\{\boldsymbol{\pi}_k(i)\}$ are correlated over space in view of the first requirement in (1.52b). The mean and co-variance(s) of each $\boldsymbol{\pi}_k(i)$ are denoted by:*

$$
\bar{\pi}_k \overset{\Delta}{=} \mathbb{E}\,\boldsymbol{\pi}_k(i) \tag{1.53a}
$$

$$
c_{\pi,k\ell} \overset{\Delta}{=} \mathbb{E}\,(\boldsymbol{\pi}_k(i) - \bar{\pi}_k)(\boldsymbol{\pi}_\ell(i) - \bar{\pi}_\ell) \tag{1.53b}
$$

for all $k, \ell = 1, 2, \dots, N$ and all $i \geq 0$. When $k = \ell$, the scalar $c_{\pi,kk}$ corresponds to the variance of $\boldsymbol{\pi}_k(i)$ and, therefore, we shall also use the alternative notation $\sigma_{\pi,k}^2$ for this case:

$$
\sigma_{\pi,k}^2 \overset{\Delta}{=} c_{\pi,kk} \geq 0 \tag{1.53c}
$$

∎

It is straightforward to verify that the first and second-order moments of the coeffi-

cients $\{\pi_k(i)\}$ satisfy:

$$\bar{\pi}_k \geq 0, \quad \sum_{k=1}^{N} \bar{\pi}_k = 1 \tag{1.54a}$$

$$\sum_{k=1}^{N} c_{\pi,k\ell} = 0, \quad \text{for any } \ell \tag{1.54b}$$

$$\sum_{\ell=1}^{N} c_{\pi,k\ell} = 0, \quad \text{for any } k \tag{1.54c}$$

Note that the earlier asynchronous implementation (1.51a) is a special case of the more general formulation (1.52a) when the mean of the random fusion coefficients are chosen as $\bar{\pi}_k = 1/N$ and their variances are set to zero, $\sigma_{\pi,k}^2 = 0$. In order to enable a fair and meaningful comparison among the three centralized implementations (1.50a), (1.51a), and (1.52a), we shall assume that the means of the fusion coefficients in the latest implementation are set to $\bar{\pi}_k = 1/N$ (results for arbitrary mean values are listed in Example 9 further ahead and appear in [4]). We will show in the sequel that for this choice of $\bar{\pi}_k$, the MSD performance of implementation (1.52a) is given by:

$$\text{MSD}_{\text{cent}}^{\text{asyn}} \approx \frac{\mu_x M}{N} \left[\frac{1}{N} \sum_{k=1}^{N} \left(1 + N^2 \sigma_{\pi,k}^2\right) \sigma_{v,k}^2 \right] \tag{1.55a}$$

Moreover, using expression (1.62b) given further ahead, this asynchronous centralized solution will converge towards the above MSD level at the same rate as the non-cooperative solution (1.49c) and the synchronous version (1.50c):

$$\alpha_{\text{cent}}^{\text{asyn}} \approx 1 - 4\mu_x \lambda_{\min}(R_u) \tag{1.55b}$$

1.3.3 **STOCHASTIC-GRADIENT CENTRALIZED SOLUTION**

The previous two sections focused on mean-square-error costs. We now extend the conclusions to more general costs. Thus, consider a collection of N agents, each with an individual convex cost function, $J_k(w)$. The objective is to determine the unique minimizer w^o of the aggregate cost:

$$J^{\text{glob}}(w) \stackrel{\Delta}{=} \sum_{k=1}^{N} J_k(w) \tag{1.56}$$

It is now the above aggregate cost, $J^{\text{glob}}(w)$, that will be required to satisfy the conditions of Assumptions 1 and 4 relative to some parameters $\{\nu_c, \delta_c, \tau_c\}$, with the subscript "c" used to indicate that these factors correspond to the centralized implemen-

tation. Under these conditions, the cost $J^{\text{glob}}(w)$ will have a unique minimizer, which we continue to denote by w^o. We will not be requiring each individual cost, $J_k(w)$, to be strongly convex. It is sufficient for at least one of these costs to be strongly convex while the remaining costs can be convex; this condition ensures the strong convexity of $J^{\text{glob}}(w)$. Moreover, although some individual costs may not have a unique minimizer, we require in this exposition that w^o is one of their minima so that all individual costs share a minimum at location w^o; the treatment in [1, 5, 65] considers the more general case in which the minimizers of the individual costs $\{J_k(w)\}$ can be different and need not contain a common location, w^o.

There are many centralized solutions that can be used to determine the unique minimizer w^o of (1.56), with some solution techniques being more powerful than other techniques. Nevertheless, we shall focus on centralized implementations of the stochastic-gradient type. The reason we consider the *same* class of stochastic gradient algorithms for non-cooperative, centralized, and distributed solutions in this chapter is to enable a *meaningful* comparison among the various implementations. Thus, we first consider a *synchronous* centralized strategy of the following form:

$$\boldsymbol{w}_i = \boldsymbol{w}_{i-1} - \mu \left[\frac{1}{N} \sum_{k=1}^{N} \widehat{\nabla_{w^{\mathsf{T}}} J_k}(\boldsymbol{w}_{i-1}) \right], \quad i \geq 0 \tag{1.57a}$$

with a constant step-size, μ. When the fusion center employs random step-sizes, the above solution is replaced by:

$$\boldsymbol{w}_i = \boldsymbol{w}_{i-1} - \boldsymbol{\mu}(i) \left[\frac{1}{N} \sum_{k=1}^{N} \widehat{\nabla_{w^{\mathsf{T}}} J_k}(\boldsymbol{w}_{i-1}) \right], \quad i \geq 0 \tag{1.57b}$$

where the process $\boldsymbol{\mu}(i)$ now satisfies Assumption 3. More generally, the asynchronous implementation can employ random fusion coefficients as well such as:

$$\boldsymbol{w}_i = \boldsymbol{w}_{i-1} - \boldsymbol{\mu}(i) \sum_{k=1}^{N} \boldsymbol{\pi}_k(i) \widehat{\nabla_{w^{\mathsf{T}}} J_k}(\boldsymbol{w}_{i-1}), \quad i \geq 0 \tag{1.57c}$$

where the coefficients $\{\boldsymbol{\pi}_k(i)\}$ satisfy Assumption 5 with means

$$\bar{\pi}_k = 1/N \tag{1.57d}$$

1.3.4 PERFORMANCE OF CENTRALIZED SOLUTION

To examine the performance of the asynchronous implementation (1.57c)–(1.57d), we proceed in two steps. First, we identify the gradient noise that is present in the recursion; it is equal to the difference between the true gradient vector for the global cost, $J^{\text{glob}}(w)$, defined by (1.56) and its approximation. Second, we argue that (1.57c) has a form similar to the single-agent stochastic-gradient algorithm (1.21)

and, therefore, invoke earlier results to write down performance metrics for the centralized solution (1.57c)–(1.57d).

We start by introducing the individual gradient noise processes:

$$s_{k,i}(w_{i-1}) \triangleq \widehat{\nabla_{w^\top} J_k}(w_{i-1}) - \nabla_{w^\top} J_k(w_{i-1})$$ (1.58a)

for $k = 1, 2, \ldots, N$. We assume that these noises satisfy conditions similar to Assumption 2 with parameters $\{\beta_k^2, \sigma_{s,k}^2, R_{s,k}\}$, i.e.,

$$R_{s,k} \triangleq \lim_{i \to \infty} \mathbb{E}\left[s_{k,i}(w^o)s_{k,i}^\top(w^o)|\mathcal{F}_{i-1}\right]$$ (1.58b)

and

$$\mathbb{E}\left[s_{k,i}(w_{i-1})|\mathcal{F}_{i-1}\right] = 0$$ (1.58c)

$$\mathbb{E}\left[\|s_{k,i}(w_{i-1})\|^2|\mathcal{F}_{i-1}\right] \leq \beta_k^2 \|\widetilde{w}_{i-1}\|^2 + \sigma_{s,k}^2$$ (1.58d)

Additionally, we assume that the gradient noise components across the agents are uncorrelated with each other:

$$\mathbb{E}\left[s_{k,i}(w_{i-1})s_{\ell,i}^\top(w_{i-1})|\mathcal{F}_{i-1}\right] = 0, \quad \text{all } k \neq \ell$$ (1.58e)

Using these gradient noise terms, it is straightforward to verify that recursion (1.57c) can be rewritten as:

$$w_i = w_{i-1} - \frac{\mu(i)}{N}\left[s_i(w_{i-1}) + \sum_{k=1}^{N} \nabla_{w^\top} J_k(w_{i-1})\right]$$ (1.59)

where $s_i(w_{i-1})$ denotes the overall gradient noise; its expression is given by

$$s_i(w_{i-1}) = \sum_{k=1}^{N}\left[N\pi_k(i)\widehat{\nabla_{w^\top} J_k}(w_{i-1}) - \nabla_{w^\top} J_k(w_{i-1})\right]$$ (1.60)

Since iteration (1.59) has the form of a stochastic gradient recursion with random update similar to (1.21), we can infer its mean-square-error behavior from Lemmas 1 and 2 if the noise process $s_i(w_{i-1})$ can be shown to satisfy conditions similar to Assumption 2 with some parameters $\{\beta_c^2, \sigma_s^2\}$. Indeed, starting from (1.60), some algebra will show that

$$\mathbb{E}\left[s_i(w_{i-1})|\mathcal{F}_{i-1}\right] = 0$$ (1.61a)

$$\mathbb{E}\left[\|s_i(w_{i-1})\|^2|\mathcal{F}_{i-1}\right] \leq \beta_c^2\|\widetilde{w}_{i-1}\|^2 + \sigma_s^2$$ (1.61b)

28   CHAPTER 1 Asynchronous Adaptive Networks

where

$$\beta_c^2 \quad \triangleq \quad \sum_{k=1}^{N} \left[\beta_k^2 + N^2 \sigma_{\pi,k}^2 (\beta_k^2 + \delta_c^2) \right] \tag{1.61c}$$

$$\sigma_s^2 \quad \triangleq \quad \sum_{k=1}^{N} (1 + N^2 \sigma_{\pi,k}^2) \sigma_{s,k}^2 \tag{1.61d}$$

The following result now follows from Lemmas 1 and 2 [4, 5].

Lemma 3 (**Convergence of centralized solution**). *Assume the aggregate cost* (1.56) *satisfies the conditions under Assumption 1 for some parameters* $0 < \nu_c \leq \delta_c$. *Assume further that the individual gradient noise processes defined by* (1.58a) *satisfy the conditions under Assumption 2 for some parameters* $\{\beta_k^2, \sigma_{s,k}^2, R_{s,k}\}$, *in addition to the orthogonality condition* (1.58e). *Let* $\mu_o = 2\nu_c/(\delta_c^2 + \beta_c^2)$. *For any* $\mu_x/N < \mu_o$, *the iterates generated by the asynchronous centralized solution* (1.57c)–(1.57d) *satisfy:*

$$\mathbb{E} \|\widetilde{w}_i\|^2 \ \leq \ \alpha \, \mathbb{E} \|\widetilde{w}_{i-1}\|^2 + \sigma_s^2 (\bar{\mu}^2 + \sigma_\mu^2)/N^2 \tag{1.62a}$$

where the scalar α *satisfies* $0 \leq \alpha < 1$ *and is given by*

$$\alpha_{\text{cent}}^{\text{asyn}} = 1 - 2\nu_c \, (\mu_x/N) + (\delta_c^2 + \beta_c^2)(\bar{\mu}^2 + \sigma_\mu^2)/N^2 \tag{1.62b}$$

It follows from (1.62a) *that for sufficiently small step-size parameter* $\mu_x \ll 1$:

$$\limsup_{i \to \infty} \mathbb{E} \|\widetilde{w}_i\|^2 = O(\mu_x) \tag{1.62c}$$

Moreover, under smoothness conditions similar to (1.35a) *for* $J^{glob}(w)$ *for some parameter* $\tau_c \geq 0$, *and similar to* (1.35b) *for the individual gradient noise covariance matrices, it holds for small* μ_x *that:*

$$\text{MSD}_{\text{cent}}^{\text{asyn}} = \frac{\mu_x}{2N} \, \text{Tr} \left[\left(\sum_{k=1}^{N} H_k \right)^{-1} \left(\sum_{k=1}^{N} (1 + N^2 \sigma_{\pi,k}^2) R_{s,k} \right) \right] + O\left(\mu_x^{3/2} \right) \tag{1.63}$$

where $H_k = \nabla_w^2 J_k(w^o)$. ∎

We can recover from the expressions in the lemma, performance results for the particular asynchronous implementation described earlier by (1.57b) by setting $\sigma_{\pi,k}^2 = 0$ so that

$$\text{MSD}_{\text{cent}}^{\text{asyn},1} \quad \approx \quad \frac{\mu_x}{2N} \, \text{Tr} \left[\left(\sum_{k=1}^{N} H_k \right)^{-1} \left(\sum_{k=1}^{N} R_{s,k} \right) \right] \tag{1.64a}$$

$$\alpha_{\text{cent}}^{\text{asyn},1} \quad \approx \quad 1 - 2\nu_c \mu_x/N \tag{1.64b}$$

It is seen from (1.63) and (1.64a) that when the fusion center operates under the more general asynchronous policy (1.57c), the additional randomness in the fusion coeffi-

cients $\{\pi_k(i)\}$ degrades the MSD performance relative to (1.64a) due to the presence of the factor $1 + N^2\sigma_{\pi,k}^2 > 1$, i.e., to first-order in μ_x we have:

$$\text{MSD}^{\text{asyn,1}} < \text{MSD}^{\text{asyn}} \tag{1.65a}$$

In comparison, the added randomness due to the $\{\pi_k(i)\}$ does not have significant impact on the convergence rate since it is straightforward to see that, to first order in the step-size parameter μ_x:

$$\alpha_{\text{cent}}^{\text{asyn,1}} \approx \alpha_{\text{cent}}^{\text{asyn}} \tag{1.65b}$$

Likewise, setting $\bar{\mu}_k = \mu$ and $\sigma_{\pi,k}^2 = 0$, the performance of the synchronous centralized solution (1.57a) is obtained as a special case of the results in the lemma:

$$\text{MSD}_{\text{cent}}^{\text{sync}} \approx \frac{\mu}{2N} \text{Tr}\left[\left(\sum_{k=1}^{N} H_k\right)^{-1}\left(\sum_{k=1}^{N} R_{s,k}\right)\right] \tag{1.66a}$$

$$\alpha_{\text{cent}}^{\text{sync}} \approx 1 - 2\nu_c\mu/N \tag{1.66b}$$

These expressions agree with the performance results presented in [1, 5].

Example 9 (**Case of general mean-values**). Although not used in this chapter, we remark in passing that if the mean values $\{\bar{\pi}_k\}$ are not fixed at $1/N$, as was required by (1.57d), then the MSD performance of the asynchronous centralized solution (1.57c) will instead be given by

$$\text{MSD}_{\text{cent}}^{\text{asyn}} = \frac{\mu_x N}{2} \text{Tr}\left[\left(\sum_{k=1}^{N} H_k\right)^{-1}\left(\sum_{k=1}^{N} (\bar{\pi}_k^2 + \sigma_{\pi,k}^2)R_{s,k}\right)\right] + O\left(\mu_x^{3/2}\right) \tag{1.67}$$

∎

1.3.5 **COMPARISON WITH NON-COOPERATIVE PROCESSING**

We can now compare the performance of the asynchronous centralized solution (1.57c) against the performance of non-cooperative processing when agents act independently of each other and run the recursion:

$$\boxed{w_{k,i} = w_{k,i-1} - \mu(i)\,\widehat{\nabla_{w^{\mathsf{T}}}J_k}(w_{k,i-1}), \quad i \geq 0} \tag{1.68a}$$

This comparison is *meaningful* when all agents share the same unique minimizer so that we can compare how well the individual agents are able to recover the same w^o as the centralized solution. For this reason, we re-introduce the requirement that all individual costs $\{J_k(w)\}$ are ν-strongly convex with a uniform parameter ν. Since $J^{\text{glob}}(w)$ is the aggregate sum of the individual costs, then we can set the lower bound ν_c for the Hessian of $J^{\text{glob}}(w)$ at $\nu_c = N\nu$. From expressions (1.28b) and (1.64b) we then conclude that, for a sufficiently small μ_x, the convergence rates of the asynchronous non-cooperative solution (1.68a) and the asynchronous centralized solution

with random update (1.57b) will be similar:

$$\alpha_{\text{cent}}^{\text{asyn}} \quad \approx \quad 1 - 2\nu_c\,(\mu_x/N) \;=\; 1 - 2\nu\mu_x \;\approx\; \alpha_{\text{ncop},k}^{\text{asyn}} \tag{1.68b}$$

Moreover, we observe from (1.41a) that the average MSD level across N non-cooperative asynchronous agents is given by

$$\text{MSD}_{\text{ncop,av}}^{\text{asyn}} \approx \frac{\mu_x}{2N}\,\text{Tr}\!\left[\sum_{k=1}^{N} H_k^{-1} R_{s,k}\right] \tag{1.68c}$$

so that comparing with (1.64a), some simple algebra allows us to conclude that, for small step-sizes and to first order in μ_x:

$$\text{MSD}_{\text{cent}}^{\text{asyn,1}} \;\leq\; \text{MSD}_{\text{ncop,av}}^{\text{asyn}} \tag{1.68d}$$

That is, while the asynchronous centralized solution (1.57b) with random updates need not outperform every individual non-cooperative agent in general, its performance outperforms the average performance across all non-cooperative agents.

The next example illustrates this result by considering the scenario where all agents have the same Hessian matrices at $w = w^o$, namely, $H_k \equiv H$ for $k = 1, 2, \ldots, N$. This situation occurs, for example, when the individual costs are identical across the agents, say, $J_k(w) \equiv J(w)$, as is common in machine learning applications. This situation also occurs for the mean-square-error costs we considered earlier in this section when the regression covariance matrices, $\{R_{u,k}\}$, are uniform across all agents, i.e., $R_{u,k} \equiv R_u$ for $k = 1, 2, \ldots, N$. In these cases with uniform Hessian matrices H_k, the example below establishes that the asynchronous centralized solution (1.57b) with random updates improves over the average MSD performance of the non-cooperative solution (1.68a) by a factor of N.

Example 10 (*N***-fold improvement in MSD performance**). Consider a collection of N agents whose individual cost functions, $J_k(w)$, are ν-strongly convex and are minimized at the same location $w = w^o$. The costs are also assumed to have identical Hessian matrices at $w = w^o$, i.e., $H_k \equiv H$. Then, using (1.64a), the MSD of the asynchronous centralized implementation (1.57b) with random updates is given by:

$$\text{MSD}_{\text{cent}}^{\text{asyn,1}} \approx \frac{1}{N}\left(\frac{\mu_x}{2N}\sum_{k=1}^{N}\text{Tr}(H^{-1} R_{s,k})\right) \approx \frac{1}{N}\,\text{MSD}_{\text{ncop,av}}^{\text{asyn}} \tag{1.69}$$

∎

Example 11 (**Random fusion can degrade MSD performance**). Although the convergence rates of the asynchronous centralized solution (1.57c) and the non-cooperative solution (1.68a) agree to first-order in μ_x, the relation between their MSD values is indefinite (contrary to (1.68d)), as illustrated by the following example.

Consider the same setting of Example 10 and assume further that the variances of the random fusion coefficients are uniform, i.e., $\sigma_{\pi,k}^2 \equiv \sigma_\pi^2$. Then, using (1.63), the MSD of the asynchronous centralized implementation (1.57c) is given by

$$\text{MSD}_{\text{cent}}^{\text{asyn}} \quad \approx \quad \frac{1 + N^2\sigma_\pi^2}{N}\left(\frac{\mu_x}{2N}\sum_{k=1}^{N}\text{Tr}(H^{-1} R_{s,k})\right) \;\approx\; \left(\frac{1 + N^2\sigma_\pi^2}{N}\right)\text{MSD}_{\text{ncop,av}}^{\text{asyn}} \tag{1.70}$$

Therefore, to first-order in μ_x, we find that

$$
\begin{cases}
\text{MSD}^{\text{asyn}}_{\text{cent}} \leq \text{MSD}^{\text{asyn}}_{\text{ncop,av}}, & \text{if } \sigma^2_\pi \leq \frac{N-1}{N^2} \\
\text{MSD}^{\text{asyn}}_{\text{cent}} > \text{MSD}^{\text{asyn}}_{\text{ncop,av}}, & \text{if } \sigma^2_\pi > \frac{N-1}{N^2}
\end{cases}
\tag{1.71}
$$

In other words, if the variance of the random fusion coefficients is large enough, then the centralized solution will generally have degraded performance relative to the non-cooperative solution (which is an expected result).

∎

Example 12 (Fully-connected networks). In preparation for the discussion on networked agents, it is useful to describe one extreme situation where a collection of N agents are fully connected to each other — see Fig. 1.3. In this case, each agent is able to access the data from all other agents and, therefore, each individual agent can run a synchronous or asynchronous centralized implementation, say, one of the same form as (1.57b):

$$
\boldsymbol{w}_{k,i} = \boldsymbol{w}_{k,i-1} - \boldsymbol{\mu}(i) \left[\frac{1}{N} \sum_{\ell=1}^{N} \widehat{\nabla_{w^\top} J_\ell}(\boldsymbol{w}_{k,i-1}) \right], \quad i \geq 0
\tag{1.72}
$$

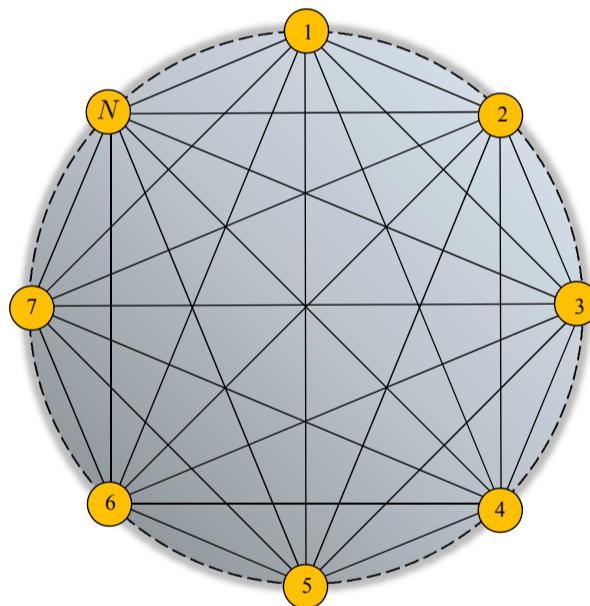**FIGURE 1.3   Fully-connected network.**

Example of a fully-connected network, where each agent can access information from all other agents. *Source: Figure extracted with permission from [1].*

When this happens, each agent will attain the same performance level as that of the asynchronous centralized solution (1.57b). Two observations are in place. First, note from (1.72) that the information

that agent k is receiving from all other agents is their gradient vector approximations. Obviously, other pieces of information could be shared among the agents, such as their iterates $\{w_{\ell,i-1}\}$. Second, note that the right-most term multiplying $\mu(i)$ in (1.72) corresponds to a convex combination of the approximate gradients from the various agents, with the combination coefficients being uniform and all equal to $1/N$. In general, there is no need for these combination weights to be identical. Even more importantly, agents do not need to have access to information from all other agents in the network. We are going to see in the sequel that interactions with a limited number of neighbors is sufficient for the agents to attain performance that is comparable to that of the centralized solution.

Figure 1.4 shows a simple selection of connected topologies for five agents. The leftmost panel corresponds to the non-cooperative case and the rightmost panel corresponds to the fully-connected case. The panels in between illustrate some other topologies. In the coming sections, we are going to present results that allow us to answer useful questions about such networked agents such as: (a) Which topology has best performance in terms of mean-square error and convergence rate? (b) Given a connected topology, can it be made to approach the performance of the centralized solution? (c) Which aspects of the topology influence performance? (d) Which aspects of the combination weights (policy) influence performance? (e) Can different topologies deliver similar performance levels? (f) Is cooperation always beneficial? and (g) If the individual agents are able to solve the inference task individually in a stable manner, does it follow that the connected network will remain stable regardless of the topology and regardless of the cooperation strategy?

non-cooperative agents   examples of connected networks   fully-connected network
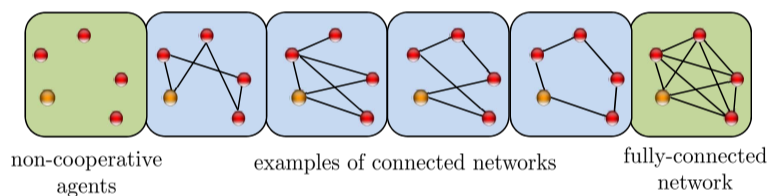
FIGURE 1.4   Connected networks.

Examples of connected networks, with the left-most panel representing a collection of non-cooperative agents. *Source: Figure extracted with permission from [1].*

■

1.4 SYNCHRONOUS MULTI-AGENT ADAPTATION AND LEARNING

In this section, we describe distributed strategies of the consensus (e.g., [7, 11, 15] and [68]–[77]) and diffusion (e.g., [1, 5, 6, 7, 41, 78, 79, 80]) types. These strategies rely solely on localized interactions among neighboring agents, and they can be used to seek the minimizer of (1.56). We first describe the network model.

1.4.1 **STRONGLY-CONNECTED NETWORKS**

Figure 1.5 shows an example of a network consisting of N connected agents, labeled $k = 1, 2, \ldots, N$. Following the presentation from [5, 7], the network is represented by a graph consisting of N vertices (representing the agents) and a set of edges connecting the agents to each other. An edge that connects an agent to itself is called a self-loop. The neighborhood of an agent k is denoted by $\mathcal{N}_k$ and it consists of all agents that are connected to k by an edge. Any two neighboring agents k and ℓ have the ability to share information over the edge connecting them.
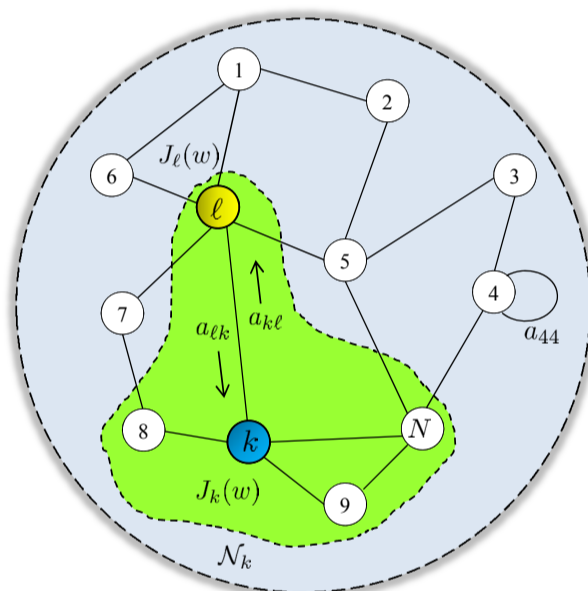
FIGURE 1.5   Neighborhoods and interactions.

Agents that are linked by edges can share information. The neighborhood of agent k is marked by the highlighted area. *Source: Figure extracted with permission from [1].*

We assume an undirected graph so that if agent k is a neighbor of agent ℓ, then agent ℓ is also a neighbor of agent k. We assign a pair of nonnegative scaling weights, $\{a_{k\ell}, a_{\ell k}\}$, to the edge connecting k and ℓ. The scalar $a_{\ell k}$ is used by agent k to scale the data it receives from agent ℓ; this scaling can be interpreted as a measure of the confidence level that agent k assigns to its interaction with agent ℓ. Likewise, $a_{k\ell}$ is used by agent ℓ to scale the data it receives from agent k. The weights $\{a_{k\ell}, a_{\ell k}\}$ can be different so that the exchange of information between the neighboring agents $\{k, \ell\}$ need not be symmetrical. One or both weights can also be zero.

A network is said to be connected if paths with nonzero scaling weights can be

found linking any two distinct agents in *both* directions, either directly when they are neighbors or by passing through intermediate agents when they are not neighbors. In this way, information can flow in both directions between any two agents in the network, although the forward path from an agent k to some other agent ℓ need not be the same as the backward path from ℓ to k. A strongly-connected network is a connected network with at least one non-trivial self-loop, meaning that $a_{kk} > 0$ for some agent k.

The strong connectivity of a network translates into a useful property on the combination weights. Assume we collect the coefficients $\{a_{\ell k}\}$ into an $N \times N$ matrix $A = [a_{\ell k}]$, such that the entries on the k-th column of A contain the coefficients used by agent k to scale data arriving from its neighbors $\ell \in \mathcal{N}_k$; we set $a_{\ell k} = 0$ if $\ell \notin \mathcal{N}_k$. We refer to A as the *combination* matrix or policy. It turns out that combination matrices that correspond to strongly-connected networks are *primitive* — an $N \times N$ matrix A with nonnegative entries is said to be primitive if there exists some finite integer $n_o > 0$ such that all entries of A^{n_o} are strictly positive [5, 7, 81].

1.4.2 DISTRIBUTED OPTIMIZATION

Network cooperation can be exploited to solve adaptation, learning, and optimization problems in a decentralized manner in response to streaming data. To explain how cooperation can be achieved, we start by associating with each agent k a twice-differentiable cost function $J_k(w) : \mathbb{R}^{M \times 1} \mapsto \mathbb{R}$. The objective of the network of agents is to seek the unique minimizer of the aggregate cost function, $J^{\mathrm{glob}}(w)$, defined by (1.56). Now, however, we seek a *distributed* (as opposed to a centralized) solution. In a distributed implementation, each agent k can only rely on its own data and on data from its neighbors.

We continue to assume that $J^{\mathrm{glob}}(w)$ satisfies the conditions of Assumptions 1 and 4 with parameters $\{\nu_d, \delta_d, \tau_d\}$, with the subscript "d" now used to indicate that these parameters are related to the distributed implementation. Under these conditions, the cost $J^{\mathrm{glob}}(w)$ will have a unique minimizer, which we continue to denote by w^o. For simplicity of presentation, we will also assume in the remainder of this chapter that the individual costs $J_k(w)$ are strongly convex as well. These costs can be distinct across the agents or they can all be identical, i.e., $J_k(w) \equiv J(w)$ for $k = 1, 2, \ldots, N$; in the latter situation, the problem of minimizing (1.56) would correspond to the case in which the agents work together to optimize the same cost function. If we let w_k^o denote the minimizer of $J_k(w)$, we continue to assume for this exposition that each $J_k(w)$ is also minimized at w^o:

$$w_k^o \equiv w^o, \quad k = 1, 2, \ldots, N \tag{1.73}$$

The case where the individual costs are only convex and need not be strongly convex is discussed in [5, 1, 42]; most of the results and conclusions continue to hold but the derivations become more technical. Likewise, the case in which the individual costs need not share minimizers is discussed in these references. In that case, as was already shown in [65], the iterates $\boldsymbol{w}_{k,i}$ by the individual agents will not approach

the minimizer w^o of (1.56), but rather the minimizer $w^\star$ of a weighted aggregate function with some positive scaling weights $\{q_k\}$. This function was defined in Eq. (57b) of [1]. It was also explained in [1] how this convergence property adds a useful degree of freedom to the operation of the network, and how it can be exploited advantageously to steer the network to converge to desirable limit points, including to w^o, through the selection of the combination policy A (which determines the scaling weights $\{q_k\}$ in the weighted aggregate cost).

It is, nevertheless, sufficient for the purposes of this chapter to continue with the case (1.73). There are many important situations in practice where the minimizers of individual costs coincide with each other. For instance, examples abound where agents need to work cooperatively to attain a common objective such as tracking a target, locating a food source, or evading a predator (see, e.g., [82, 83, 6]). This scenario is also common in machine learning problems [31, 32, 84, 85, 86, 87] when data samples at the various agents are generated by a common distribution parameterized by some vector, w^o. One such situation is illustrated in the next example.

Example 13 (Mean-square-error (MSE) networks). Consider the same setting of Example 3 except that we now have N agents observing streaming data $\{d_k(i), u_{k,i}\}$ that satisfy the regression model (1.13a) with regression covariance matrices $R_{u,k} = \mathbb{E}\, u_{k,i}^{\mathsf{T}} u_{k,i} > 0$ and with the same unknown w^o, i.e.,

$$d_k(i) = u_{k,i} w^o + v_k(i) \tag{1.74a}$$

The individual mean-square-error costs are defined by

$$J_k(w) = \mathbb{E}\,(d_k(i) - u_{k,i} w)^2 \tag{1.74b}$$

and are strongly convex in this case, with the minimizer of each $J_k(w)$ occurring at

$$w_k^o \triangleq R_{u,k}^{-1} r_{du,k}, \quad k = 1, 2, \ldots, N \tag{1.74c}$$

If we multiply both sides of (1.74a) by $u_{k,i}^{\mathsf{T}}$ from the left, and take expectations, we find that w^o satisfies

$$r_{du,k} = R_{u,k} w^o \tag{1.75}$$

This relation shows that the unknown w^o from (1.74a) satisfies the same expression as w_k^o in (1.74c), for any $k = 1, 2, \ldots, N$, so that we must have $w^o = w_k^o$. Therefore, this example amounts to a situation where all costs $\{J_k(w)\}$ attain their minima at the same location, w^o.

We shall use the network model of this example to illustrate other results in the chapter. For ease of reference, we shall refer to strongly-connected networks with agents receiving data according to model (1.74a) and seeking to estimate w^o by adopting the mean-square-error costs $J_k(w)$ defined above, as *mean-square-error* (MSE) *networks*. We assume for these networks that the measurement noise process $v_k(i)$ is temporally white and independent over space so that

$$\mathbb{E}\, v_k(i) v_\ell(j) = \sigma_{v,k}^2 \delta_{k,\ell} \delta_{i,j} \tag{1.76}$$

in terms of the Kronecker delta $\delta_{k,\ell}$. Likewise, we assume that the regression data $u_{k,i}$ is temporally white and independent over space so that

$$\mathbb{E}\, u_{k,i}^{\mathsf{T}} u_{\ell,j} = R_{u,k} \delta_{k,\ell} \delta_{i,j} \tag{1.77}$$

Moreover, the measurement noise $v_k(i)$ and the regression data $u_{\ell,j}$ are independent of each other for all k, ℓ, i, j. These statistical conditions help facilitate the analysis of such networks.

∎

36   **CHAPTER 1** Asynchronous Adaptive Networks

In the next subsections we list *synchronous* distributed algorithms of the consensus and diffusion types for the optimization of $J^{\mathrm{glob}}(w)$. We only list the algorithms here; for motivation and justifications, the reader may refer to the treatments in [1, 5]. Moreover, Sec. V.D of [1] provides commentary on several other related works in the literature, in addition to the history and evolution of the consensus and diffusion strategies.

1.4.3 SYNCHRONOUS CONSENSUS STRATEGY

Let $w_{k,i}$ denote the iterate that is available at agent k at iteration i; this iterate serves as the estimate for w^o. The consensus iteration at each agent k is described by the following construction (see, e.g., [11, 15, 36, 69, 71, 77]):

$$w_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k}\, w_{\ell,i-1} - \mu_k\, \widehat{\nabla_{w^\mathsf{T}} J_k}(w_{k,i-1}) \qquad (1.78)$$

where the $\{\mu_k\}$ are individual step-size parameters, and where the combination coefficients $\{a_{\ell k}\}$ that appear in (1.78) are nonnegative scalars that are required to satisfy the following conditions for each agent $k = 1, 2, \ldots, N$:

$$a_{\ell k} \geq 0, \quad \sum_{\ell=1}^{N} a_{\ell k} = 1, \quad a_{\ell k} = 0 \text{ if } \ell \notin \mathcal{N}_k \qquad (1.79a)$$

Condition (1.79a) implies that the combination matrix $A = [a_{\ell k}]$ satisfies

$$A^\mathsf{T} \mathbb{1} = \mathbb{1} \qquad (1.79b)$$

where $\mathbb{1}$ denotes the vector with all entries equal to one. We say that A is left-stochastic. One useful property of left-stochastic matrices is that their spectral radius is equal to one [7, 81, 88, 89, 90]:

$$\rho(A) = 1 \qquad (1.79c)$$

An equivalent representation that is useful for later analysis is to rewrite the consensus iteration (1.78) as shown in the following listing, where the intermediate iterate that results from the neighborhood combination is denoted by $\psi_{k,i-1}$. Observe that the gradient vector in the consensus implementation (1.80) is evaluated at $w_{k,i-1}$ and not $\psi_{k,i-1}$.

Consensus strategy for distributed adaptation

for each time instant $i \geq 0$:

each agent $k = 1, 2, \ldots, N$ performs the update:

$$\begin{cases} \boldsymbol{\psi}_{k,i-1} = \displaystyle\sum_{\ell \in \mathcal{N}_k} a_{\ell k} \, \boldsymbol{w}_{\ell,i-1} \\[2mm] \boldsymbol{w}_{k,i} = \boldsymbol{\psi}_{k,i-1} - \mu_k \, \widehat{\nabla_{w^{\mathsf{T}}} J}_k \left(\boldsymbol{w}_{k,i-1} \right) \end{cases} \qquad (1.80)$$

end

We remark that one way to motivate the consensus update (1.78) is to start from the non-cooperative step (1.68a) and replace the first iterate $\boldsymbol{w}_{k,i-1}$ by the convex combination used in (1.78).

Example 14 (**Consensus LMS networks**). For the MSE network of Example 13, the consensus strategy reduces to:

$$\begin{cases} \boldsymbol{\psi}_{k,i-1} = \displaystyle\sum_{\ell \in \mathcal{N}_k} a_{\ell k} \, \boldsymbol{w}_{\ell,i-1} \\[2mm] \boldsymbol{w}_{k,i} = \boldsymbol{\psi}_{k,i-1} + 2\mu_k \boldsymbol{u}_{k,i}^{\mathsf{T}} [\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i} \boldsymbol{w}_{k,i-1}] \end{cases} \qquad (1.81)$$

∎

1.4.4 SYNCHRONOUS DIFFUSION STRATEGIES

There is an inherent asymmetry in the consensus construction. Observe from the computation of $\boldsymbol{w}_{k,i}$ in (1.80) that the update starts from $\boldsymbol{\psi}_{k,i-1}$ and corrects it by the approximate gradient vector evaluated at $\boldsymbol{w}_{k,i-1}$ (and not at $\boldsymbol{\psi}_{k,i-1}$). This *asymmetry* will be shown later, e.g., in Example 25, to be problematic when the consensus strategy is used for adaptation and learning over networks. This is because the asymmetry can cause an unstable growth in the state of the network [80] — see also the explanations in [1] and [5][Sec. 10.6]. Diffusion strategies remove the asymmetry problem.

Combine-then-Adapt (CTA) Diffusion. In the CTA formulation of the diffusion strategy, the *same* iterate $\boldsymbol{\psi}_{k,i-1}$ is used to compute $\boldsymbol{w}_{k,i}$, thus leading to description (1.82a) where the gradient vector is evaluated at $\boldsymbol{\psi}_{k,i-1}$ as well. The reason for the name "Combine-then-Adapt" is that the first step in (1.82a) involves a combination step, while the second step involves an adaptation step. The reason for the qualification "diffusion" is that the use of $\boldsymbol{\psi}_{k,i-1}$ to evaluate the gradient vector allows information to diffuse more thoroughly through the network. This is because information is not only being diffused through the aggregation of the neighborhood iterates, but also through the evaluation of the gradient vector at the aggregate state value.

38   CHAPTER 1 Asynchronous Adaptive Networks

Diffusion strategy for distributed adaptation (CTA)

for each time instant $i \geq 0$:

each agent $k = 1, 2, \ldots, N$ performs the update:

$$\begin{cases} \boldsymbol{\psi}_{k,i-1} = \displaystyle\sum_{\ell \in \mathcal{N}_k} a_{\ell k}\, \boldsymbol{w}_{\ell,i-1} \\[2mm] \boldsymbol{w}_{k,i} = \boldsymbol{\psi}_{k,i-1} - \mu_k\, \widehat{\nabla_{w^\top} J_k}\left(\boldsymbol{\psi}_{k,i-1}\right) \end{cases} \tag{1.82a}$$

end

Adapt-then-Combine (ATC) Diffusion. A similar implementation can be obtained by switching the order of the combination and adaptation steps in (1.82a), as shown in the listing (1.82b). The structure of the CTA and ATC strategies are fundamentally identical: the difference lies in which variable we choose to correspond to the updated iterate $\boldsymbol{w}_{k,i}$. In ATC, we choose the result of the *combination* step to be $\boldsymbol{w}_{k,i}$, whereas in CTA we choose the result of the *adaptation* step to be $\boldsymbol{w}_{k,i}$.

Diffusion strategy for distributed adaptation (ATC)

for each time instant $i \geq 0$:

each agent $k = 1, 2, \ldots, N$ performs the update:

$$\begin{cases} \boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} - \mu_k\, \widehat{\nabla_{w^\top} J_k}(\boldsymbol{w}_{k,i-1}) \\[2mm] \boldsymbol{w}_{k,i} = \displaystyle\sum_{\ell \in \mathcal{N}_k} a_{\ell k}\, \boldsymbol{\psi}_{\ell,i} \end{cases} \tag{1.82b}$$

end

One main motivation for the introduction of the diffusion strategies (1.82a) and (1.82b) is the fact that they enable *single* time-scale distributed learning from *streaming* data under *constant* step-size adaptation and in a stable manner [2, 65, 78, 79] and [91]–[95] — see also [5][Chs. 9–11]; the diffusion strategies further allow A to be left-stochastic, which permit larger modes of cooperation than doubly-stochastic policies. The CTA diffusion strategy (1.83a) was first introduced for mean-square-error estimation problems in [78, 91, 92, 93]. The ATC diffusion structure (1.83b), with adaptation preceding combination, appeared in the work [96] on adaptive distributed least-squares schemes and also in the works [79, 95, 97, 98] on distributed mean-square-error and state-space estimation methods. The CTA structure (1.82a) with an iteration dependent step-size that decays to zero, $\mu(i) \to 0$, was employed in [13, 99, 100] to solve distributed optimization problems that require all agents to reach agreement. The ATC form (1.82b), also with an iteration dependent sequence $\mu(i)$ that decays to zero, was employed in [101, 102] to ensure almost-sure convergence and agreement among agents.

Example 15 (Diffusion LMS networks). For the MSE network of Example 13, the ATC and CTA diffu-

sion strategies reduce to:

$$
\begin{cases}
\boldsymbol{\psi}_{k,i-1} = \displaystyle\sum_{\ell \in \mathcal{N}_k} a_{\ell k}\, \boldsymbol{w}_{\ell,i-1} \qquad \text{(CTA diffusion)} \\[2mm]
\boldsymbol{w}_{k,i} = \boldsymbol{\psi}_{k,i-1} + 2\mu_k \boldsymbol{u}_{k,i}^{\mathsf{T}} \left[\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}\boldsymbol{\psi}_{k,i-1} \right]
\end{cases}
\tag{1.83a}
$$

and

$$
\begin{cases}
\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} + 2\mu_k \boldsymbol{u}_{k,i}^{\mathsf{T}} \left[\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}\boldsymbol{w}_{k,i-1} \right] \\[2mm]
\boldsymbol{w}_{k,i} = \displaystyle\sum_{\ell \in \mathcal{N}_k} a_{\ell k}\, \boldsymbol{\psi}_{\ell,i} \qquad \text{(ATC diffusion)}
\end{cases}
\tag{1.83b}
$$

∎

Example 16 (Diffusion logistic regression). We revisit the pattern classification problem from Example 2, where we consider a collection of N networked agents cooperating with each other to solve the logistic regression problem. Each agent receives streaming data $\{\boldsymbol{\gamma}_k(i), \boldsymbol{h}_{k,i}\}$, where the variable $\boldsymbol{\gamma}_k(i)$ assumes the values ± 1 and designates the class that the feature vector $\boldsymbol{h}_{k,i}$ belongs to. The objective is to use the training data to determine the vector w^o that minimizes the cost

$$
J_k(w) \;\triangleq\; \frac{\rho}{2}\|w\|^2 + \mathbb{E}\left\{ \ln\left[1 + e^{-\boldsymbol{\gamma}_k(i)\boldsymbol{h}_{k,i}^{\mathsf{T}} w} \right] \right\}
\tag{1.84}
$$

under the assumption of joint wide-sense stationarity over the random data. It is straightforward to verify that the ATC diffusion strategy (1.82b) reduces to the following form in this case:

$$
\begin{cases}
\boldsymbol{\psi}_{k,i} = (1 - \rho\mu_k)\boldsymbol{w}_{k,i-1} + \mu_k \left(\dfrac{\boldsymbol{\gamma}_k(i)}{1 + e^{\boldsymbol{\gamma}_k(i)\boldsymbol{h}_{k,i}^{\mathsf{T}}\boldsymbol{w}_{k,i-1}}} \right) \boldsymbol{h}_{k,i} \\[4mm]
\boldsymbol{w}_{k,i} = \displaystyle\sum_{\ell \in \mathcal{N}_k} a_{\ell k}\, \boldsymbol{\psi}_{\ell,i}
\end{cases}
\tag{1.85}
$$

∎

1.5 **ASYNCHRONOUS MULTI-AGENT ADAPTATION AND LEARNING**

There are various ways by which asynchronous events can be introduced into the operation of a distributed strategy. Without loss in generality, we illustrate the model for asynchronous operation by describing it for the ATC diffusion strategy (1.82b); similar constructions apply to CTA diffusion (1.82a) and consensus (1.80).

1.5.1 **ASYNCHRONOUS MODEL**

In a first instance, we model the step-size parameters as random variables and replace (1.82b) by:

$$
\begin{cases}
\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} - \boldsymbol{\mu}_k(i)\, \widehat{\nabla_{w^{\mathsf{T}}}\, J_k}(\boldsymbol{w}_{k,i-1}) \\[2mm]
\boldsymbol{w}_{k,i} = \displaystyle\sum_{\ell \in \mathcal{N}_k} a_{\ell k}\, \boldsymbol{\psi}_{\ell,i}
\end{cases}
\tag{1.86}
$$

40   **CHAPTER 1** Asynchronous Adaptive Networks

In this model, the neighborhoods and the network topology remain fixed and only the $\boldsymbol{\mu}_k(i)$ assume random values. The step-sizes can vary across the agents and, therefore, their means and variances become agent-dependent. Moreover, the step-sizes across agents can be correlated with each other. We therefore denote the first and second-order moments of the step-size parameters by:

$$\bar{\mu}_k \quad \overset{\Delta}{=} \quad \mathbb{E}\,\boldsymbol{\mu}_k(i) \tag{1.87a}$$

$$\sigma^2_{\mu,k} \quad \overset{\Delta}{=} \quad \mathbb{E}\,(\boldsymbol{\mu}_k(i) - \bar{\mu}_k)^2 \tag{1.87b}$$

$$c_{\mu,k\ell} \quad \overset{\Delta}{=} \quad \mathbb{E}\,(\boldsymbol{\mu}_k(i) - \bar{\mu}_k)(\boldsymbol{\mu}_\ell(i) - \bar{\mu}_\ell) \tag{1.87c}$$

When $\ell = k$, the scalar $c_{\mu,kk}$ coincides with the variance of $\boldsymbol{\mu}_k(i)$, i.e., $c_{\mu,kk} = \sigma^2_{\mu,k} \geq 0$. On the other hand, if the step-sizes across the agents happen to be uncorrelated, then $c_{\mu,k\ell} = 0$ for $k \neq \ell$.

More broadly, we can allow for random variations in the neighborhoods (and, hence, in the network structure), and random variations in the combination coefficients as well. We capture this more general asynchronous implementation by writing:

$$\begin{cases} \boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} - \boldsymbol{\mu}_k(i)\widehat{\nabla_{w^\mathsf{T}} J_k}(\boldsymbol{w}_{k,i-1}) \\ \boldsymbol{w}_{k,i} = \displaystyle\sum_{\ell \in \boldsymbol{\mathcal{N}}_{k,i}} \boldsymbol{a}_{\ell k}(i)\,\boldsymbol{\psi}_{\ell,i} \end{cases} \tag{1.88}$$

where the combination coefficients $\{\boldsymbol{a}_{\ell k}(i)\}$ are now *random* and, moreover, the symbol $\boldsymbol{\mathcal{N}}_{k,i}$ denotes the *randomly*-changing neighborhood of agent k at time i. These neighborhoods become random because the random variations in the combination coefficients can turn links on and off depending on the values of the $\{\boldsymbol{a}_{\ell,k}(i)\}$. We continue to require the combination coefficients $\{\boldsymbol{a}_{\ell k}(i)\}$ to satisfy the same structural constraint as given before by (1.79a), i.e.,

$$\sum_{\ell \in \boldsymbol{\mathcal{N}}_{k,i}} \boldsymbol{a}_{\ell k}(i) = 1, \quad \text{and} \quad \begin{cases} \boldsymbol{a}_{\ell k}(i) > 0, & \text{if } \ell \in \boldsymbol{\mathcal{N}}_{k,i} \\ \boldsymbol{a}_{\ell k}(i) = 0, & \text{otherwise} \end{cases} \tag{1.89}$$

Since these coefficients are now random, we denote their first and second-order moments by:

$$\bar{a}_{\ell k} \quad \overset{\Delta}{=} \quad \mathbb{E}\,\boldsymbol{a}_{\ell k}(i) \tag{1.90a}$$

$$\sigma^2_{a,\ell k} \quad \overset{\Delta}{=} \quad \mathbb{E}\,(\boldsymbol{a}_{\ell k}(i) - \bar{a}_{\ell k})^2 \tag{1.90b}$$

$$c_{a,\ell k,nm} \quad \overset{\Delta}{=} \quad \mathbb{E}\,(\boldsymbol{a}_{\ell k}(i) - \bar{a}_{\ell k})(\boldsymbol{a}_{nm}(i) - \bar{a}_{nm}) \tag{1.90c}$$

When $\ell = n$ and $k = m$, the scalar $c_{a,\ell k,nm}$ coincides with the variance of $\boldsymbol{a}_{\ell k}(i)$, i.e., $c_{a,\ell k,nm} = \sigma^2_{a,\ell k} \geq 0$.

Example 17 (Asynchronous diffusion LMS networks). For the MSE network of Example 13, the ATC diffusion strategy (1.86) with random update reduces to

$$
\begin{cases}
\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} + 2\boldsymbol{\mu}_k(i)\boldsymbol{u}_{k,i}^{\mathsf{T}}\,[\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}\boldsymbol{w}_{k,i-1}] \\
\boldsymbol{w}_{k,i} = \displaystyle\sum_{\ell \in \mathcal{N}_k} a_{\ell k}\,\boldsymbol{\psi}_{\ell,i} \quad \text{(diffusion with random updates)}
\end{cases}
\tag{1.91a}
$$

whereas the *asynchronous* ATC diffusion strategy (1.88) reduces to:

$$
\begin{cases}
\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} + 2\boldsymbol{\mu}_k(i)\boldsymbol{u}_{k,i}^{\mathsf{T}}\,[\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}\boldsymbol{w}_{k,i-1}] \\
\boldsymbol{w}_{k,i} = \displaystyle\sum_{\ell \in \boldsymbol{\mathcal{N}}_{k,i}} \boldsymbol{a}_{\ell k}(i)\,\boldsymbol{\psi}_{\ell,i} \quad \text{(asynchronous diffusion)}
\end{cases}
\tag{1.91b}
$$

We can view implementation (1.91a) as a special case of the asynchronous update (1.91b) when the variances of the random combination coefficients $\{\boldsymbol{a}_{\ell k}(i)\}$ are set to zero.

∎

Example 18 (Asynchronous consensus LMS networks). Similarly, for the same MSE network of Example 13, the asynchronous consensus strategy is given by

$$
\begin{cases}
\boldsymbol{\psi}_{k,i-1} = \displaystyle\sum_{\ell \in \boldsymbol{\mathcal{N}}_{k,i}} \boldsymbol{a}_{\ell k}(i)\,\boldsymbol{w}_{\ell,i-1} \\
\boldsymbol{w}_{k,i} = \boldsymbol{\psi}_{k,i-1} + 2\boldsymbol{\mu}_k(i)\boldsymbol{u}_{k,i}^{\mathsf{T}}[\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}\boldsymbol{w}_{k,i-1}]
\end{cases}
\tag{1.92}
$$

∎

1.5.2 MEAN GRAPH

We refer to the topology that corresponds to the average combination coefficients $\{\bar{a}_{\ell k}\}$ as the *mean graph*, which is fixed over time. For each agent k, the neighborhood defined by the mean graph is denoted by $\mathcal{N}_k$. It is straightforward to verify that the mean combination coefficients $\bar{a}_{\ell k}$ satisfy the following constraints over the mean graph (compare with (1.79a) and (1.89)):

$$
\boxed{\sum_{\ell \in \mathcal{N}_k} \bar{a}_{\ell k} = 1, \quad \text{and} \quad
\begin{cases}
\bar{a}_{\ell k} > 0, & \text{if } \ell \in \mathcal{N}_k \\
\bar{a}_{\ell k} = 0, & \text{otherwise}
\end{cases}}
\tag{1.93}
$$

One example of a random network with two equally probable realizations and its mean graph is shown in Fig. 1.6 [2]. The letter ω is used to index the sample space of the random matrix $\boldsymbol{A}_i$.

There is one useful result that relates the random neighborhoods $\{\boldsymbol{\mathcal{N}}_{k,i}\}$ from (1.88) to the neighborhoods $\{\mathcal{N}_k\}$ from the mean graph. It is not difficult to verify that $\mathcal{N}_k$ is equal to the *union* of all possible realizations for the random neighborhoods $\boldsymbol{\mathcal{N}}_{k,i}$; this property is already illustrated by the example of Fig. 1.6:

$$
\mathcal{N}_k = \bigcup_{\omega \in \Omega} \boldsymbol{\mathcal{N}}_{k,i}(\omega)
\tag{1.94}
$$

42   **CHAPTER 1** Asynchronous Adaptive Networks

$$\boldsymbol{A}_i(\omega_1) =$$

$$\begin{bmatrix} 0.4 & 0 & 0 & 0.2 \\ 0 & 0.3 & 0.5 & 0 \\ 0 & 0.7 & 0.5 & 0 \\ 0.6 & 0 & 0 & 0.8 \end{bmatrix}$$

$$\mathcal{N}_{1,i}(\omega_1) = \{1,4\}$$
$$\mathcal{N}_{2,i}(\omega_1) = \{2,3\}$$
$$\mathcal{N}_{3,i}(\omega_1) = \{2,3\}$$
$$\mathcal{N}_{4,i}(\omega_1) = \{1,4\}$$

$$\boldsymbol{A}_i(\omega_2) =$$

$$\begin{bmatrix} 0.4 & 0.3 & 0 & 0 \\ 0.6 & 0.5 & 0.2 & 0 \\ 0 & 0.2 & 0.5 & 0.3 \\ 0 & 0 & 0.3 & 0.7 \end{bmatrix}$$

$$\mathcal{N}_{1,i}(\omega_2) = \{1,2\}$$
$$\mathcal{N}_{2,i}(\omega_2) = \{1,2,3\}$$
$$\mathcal{N}_{3,i}(\omega_2) = \{2,3,4\}$$
$$\mathcal{N}_{4,i}(\omega_2) = \{3,4\}$$

$$\bar{A} = 0.5[\boldsymbol{A}_i(\omega_1) + \boldsymbol{A}_i(\omega_2)] =$$

$$\begin{bmatrix} 0.4 & 0.15 & 0 & 0.1 \\ 0.3 & 0.4 & 0.35 & 0 \\ 0 & 0.45 & 0.5 & 0.15 \\ 0.3 & 0 & 0.15 & 0.75 \end{bmatrix}$$

$$\mathcal{N}_1 = \{1,2,4\}$$
$$\mathcal{N}_2 = \{1,2,3\}$$
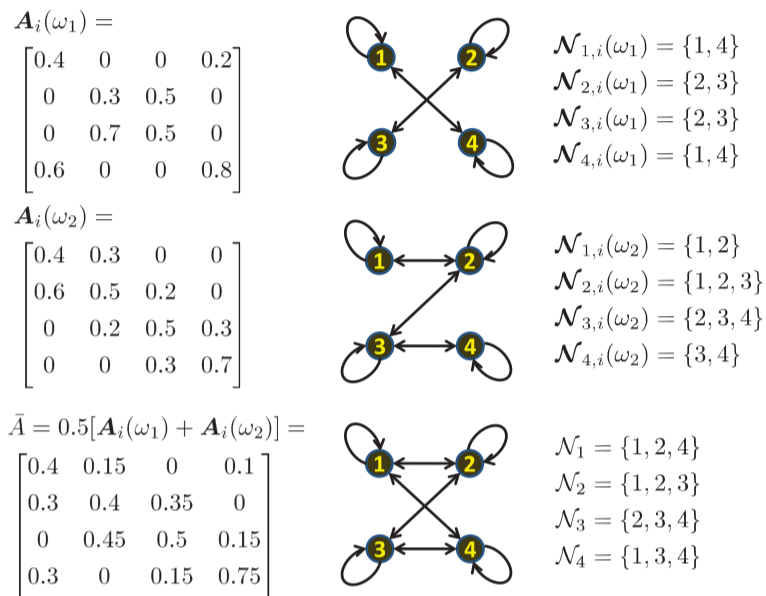$$\mathcal{N}_3 = \{2,3,4\}$$
$$\mathcal{N}_4 = \{1,3,4\}$$

FIGURE 1.6   Mean graph.
The first two rows show two equally probable realizations with the respective neighborhoods. The last row shows the resulting mean graph. *Source: This is a modified version of a Figure extracted with permission from [2].*

for any k, where Ω denotes the sample space for $\mathcal{N}_{k,i}$.

1.5.3   RANDOM COMBINATION POLICY

The first and second-order moments of the combination coefficients will play an important role in characterizing the stability and mean-square-error performance of the asynchronous network (1.88). We collect these moments in matrix form as follows. We first group the combination coefficients into a matrix:

$$\boldsymbol{A}_i \ \triangleq \ [\,\boldsymbol{a}_{\ell k}(i)\,]_{\ell,k=1}^N \qquad (N \times N) \qquad (1.95a)$$

The sequence $\{\boldsymbol{A}_i, i \geq 0\}$ represents a stochastic process consisting of left-stochastic random matrices whose entries satisfy the conditions in (1.89) at every time i. We subsequently introduce the mean and Kronecker-covariance matrix of $\boldsymbol{A}_i$ and assume these quantities are constant over time; we denote them by the $N \times N$ matrix $\bar{A}$ and

the $N^2 \times N^2$ matrix C_A, respectively:

$$\bar{A} \quad \overset{\Delta}{=} \quad \mathbb{E}\, A_i = [\, \bar{a}_{\ell k}\,]_{\ell,k=1}^{N} \tag{1.96a}$$

$$C_A \quad \overset{\Delta}{=} \quad \mathbb{E}[(A_i - \bar{A}) \otimes (A_i - \bar{A})] \tag{1.96b}$$

The matrix C_A is *not* a conventional covariance matrix and is not necessarily Hermitian. The reason for its introduction is because it captures the correlations of each entry of A_i with all other entries in A_i. For example, for a network with $N = 2$ agents, the entries of $\bar{A}$ and C_A will be given by:

$$\bar{A} \quad = \quad \begin{bmatrix} \bar{a}_{11} & \bar{a}_{12} \\ \bar{a}_{21} & \bar{a}_{22} \end{bmatrix} \tag{1.97a}$$

$$C_A \quad = \quad \left[\begin{array}{cc|cc} c_{a,11,11} & c_{a,11,12} & c_{a,12,11} & c_{a,12,12} \\ c_{a,11,21} & c_{a,11,22} & c_{a,12,21} & c_{a,12,22} \\ \hline c_{a,21,11} & c_{a,21,12} & c_{a,22,11} & c_{a,22,12} \\ c_{a,21,21} & c_{a,21,22} & c_{a,22,21} & c_{a,22,22} \end{array}\right] \tag{1.97b}$$

We thus see that the (ℓ, k)−th block of C_A contains the covariance coefficients of $\boldsymbol{a}_{\ell k}$ with all other entries of A_i. One useful property of the matrices $\{\bar{A}, C_A\}$ so defined is that their elements are nonnegative and the following matrices are left-stochastic:

$$\boxed{\left(\bar{A}\right)^{\mathsf{T}} \mathbb{1}_N = \mathbb{1}_N, \quad (\bar{A} \otimes \bar{A} + C_A)^{\mathsf{T}} \mathbb{1}_{N^2} = \mathbb{1}_{N^2}} \tag{1.98}$$

Assumption 6 (**Asynchronous network model**). *It is assumed that the random processes $\{\boldsymbol{\mu}_k(i), \boldsymbol{a}_{\ell m}(j)\}$ are independent of each other for all $k, \ell, m, i,$ and j. They are also independent of any other random variable in the learning algorithm.*

∎

The asynchronous network model described in this section covers many situations of practical interest. For example, we can choose the sample space for each step-size $\boldsymbol{\mu}_k(i)$ to be the binary choice $\{0, \mu\}$ to model random "on-off" behavior at each agent k for the purpose of saving power, waiting for data, or even due to random agent failures. Similarly, we can choose the sample space for each combination coefficient $\boldsymbol{a}_{\ell k}(i)$, $\ell \in \mathcal{N}_k \backslash \{k\}$, to be $\{0, a_{\ell k}\}$ to model a random "on-off" status for the link from agent ℓ to agent k at time i for the purpose of either saving communication cost or due to random link failures. Note that the convex constraint (1.89) can always be satisfied by adjusting the value of $\boldsymbol{a}_{kk}(i)$ according to the realizations of $\{\boldsymbol{a}_{\ell k}(i); \ell \in \mathcal{N}_{k,i} \backslash \{k\}\}$.

Example 19 (**The spatially-uncorrelated model**). A useful special case of the asynchronous network model of this section is the spatially-uncorrelated model. In this case, at each iteration i, the random step-sizes $\{\boldsymbol{\mu}_k(i); k = 1, 2, \ldots, N\}$ are uncorrelated with each other across the network, and the random combination coefficients $\{\boldsymbol{a}_{\ell k}(i); \ell \neq k, k = 1, 2, \ldots, N\}$ are also uncorrelated with each other across the

44   **CHAPTER 1** Asynchronous Adaptive Networks

network. Then, it can be verified that the covariances $\{c_{\mu,k\ell}\}$ in (1.87c) and $\{c_{a,\ell k,nm}\}$ in (1.90c) will be fully determined by the variances $\{\sigma_\mu^2, \sigma_{a,\ell k}^2\}$:

$$c_{\mu,kk} = \sigma_{\mu,k}^2, \quad c_{\mu,k\ell} = 0, \; k \neq \ell \tag{1.99a}$$

and

$$c_{a,\ell k,nm} = \begin{cases} \sigma_{a,\ell k}^2, & \text{if } k = m, \ell = n, \ell \in \mathcal{N}_k \backslash \{k\} \\ -\sigma_{a,\ell k}^2, & \text{if } k = m = n, \ell \in \mathcal{N}_k \backslash \{k\} \\ -\sigma_{a,nk}^2, & \text{if } k = m = \ell, n \in \mathcal{N}_k \backslash \{k\} \\ \displaystyle\sum_{j \in \mathcal{N}_k \backslash \{k\}} \sigma_{a,jk}^2, & \text{if } k = m = \ell = n \\ 0, & \text{otherwise} \end{cases} \tag{1.99b}$$

∎

1.5.4 **PERRON VECTORS**

Now that we introduced the network model, we can move on to examine the effect of network cooperation on performance. Some interesting patterns of behavior arise when agents cooperate to solve a global optimization problem in a distributed manner from streaming data [1, 5]. For example, one interesting result established in [2, 3, 42, 65, 103] is that the effect of the network topology on performance is captured by the Perron vector of the combination policy. This vector turns out to summarize the influence of the topology on performance so much so that different topologies with similar Perron vectors will end up delivering similar performance. We explained the role of Perron vectors in the context of synchronous adaptation and learning in [1, 5]. Here we focus on asynchronous networks. In this case, two Perron vectors will be needed since the randomness in the combination policy is now represented by two moment matrices, $\bar{A}$ and C_A. In the synchronous case, only one Perron vector was necessary since the combination policy was fixed and described by a matrix A. Although we are focusing on the asynchronous case in the sequel, we will be able to recover results for synchronous networks as special cases.

Let us first recall the definition of Perron vectors for synchronous networks, say, of the form described by (1.82b) with combination policy A. We assume the network is strongly-connected. In this case, the left-stochastic matrix A will be primitive. For such primitive matrices, it follows from the Perron-Frobenius Theorem [81] that: (a) the matrix A will have a *single* eigenvalue at one; (b) all other eigenvalues of A will be strictly inside the unit circle so that $\rho(A) = 1$; and (c) with proper sign scaling, all entries of the right-eigenvector of A corresponding to the single eigenvalue at one will be *positive*. Let p denote this right-eigenvector with its entries $\{p_k\}$ normalized to add up to one, i.e.,

$$\boxed{Ap = p, \quad \mathbb{1}^\mathsf{T} p = 1, \quad p_k > 0, \; k = 1, 2, \ldots, N} \tag{1.100}$$

We refer to p as the *Perron* eigenvector of A. It was explained in [1, 5] how the entries of this vector determine the mean-square-error performance and convergence

rate of the network; these results will be revisited further ahead when we recover them as special cases of the asynchronous results.

On the other hand, for an asynchronous implementation, the individual realizations of the random combination matrix A_i in (1.88) need not be primitive. In this context, we will require a form of primitiveness to hold on average as follows.

Definition 1 (**Strongly-connected asynchronous model**). *We say that an asynchronous model with random combination coefficients* $\{a_{\ell k}(i)\}$ *is strongly-connected if the Kronecker-covariance matrix given by* $\bar{A} \otimes \bar{A} + C_A$ *is primitive.*

∎

Observe that if we set $\bar{A} = A$ and $C_A = 0$, then we recover the condition for strong-connectedness in the synchronous case, namely, that A should be primitive.

Definition 1 means that the directed graph (digraph) associated with the matrix $\bar{A} \otimes \bar{A} + C_A$ is strongly-connected (e.g., [89, pp. 30,34] and [5]). It is straightforward to check from the definition of C_A in (1.96b) that

$$\bar{A} \otimes \bar{A} + C_A \;=\; \mathbb{E}\,(A_i \otimes A_i) \tag{1.101}$$

so that the digraph associated with $\bar{A} \otimes \bar{A} + C_A$ is the union of all possible digraphs associated with the realizations of $A_i \otimes A_i$ [104, p. 29]. Therefore, as explained in [2, 3, 4], definition 1 amounts to an assumption that the *union* of all possible digraphs associated with the realizations of $A_i \otimes A_i$ is strongly-connected. As illustrated in Fig. 1.7, this condition still allows individual digraphs associated with realizations of A_i to be weakly-connected with or without self-loops or even to be disconnected [3].

It follows from property (1.98) and Definition 1, the matrix $\bar{A} \otimes \bar{A} + C_A$ is left-stochastic and primitive. It can be verified that mean matrix $\bar{A}$ is also primitive if $\bar{A} \otimes \bar{A} + C_A$ is primitive (although the converse is not true). Therefore, the matrix $\bar{A}$ is both left-stochastic and primitive. We denote the Perron eigenvector of $\bar{A} \otimes \bar{A} + C_A$ by $p_c \in \mathbb{R}^{N^2 \times 1}$, which satisfies:

$$\boxed{(\bar{A} \otimes \bar{A} + C_A)p_c = p_c, \qquad p_c^{\mathsf{T}}\mathbb{1} = 1} \tag{1.102a}$$

Likewise, we denote the Perron eigenvector of $\bar{A}$ by $\bar{p} \in \mathbb{R}^{N \times 1}$, which satisfies:

$$\boxed{\bar{A}\bar{p} = \bar{p}, \qquad \bar{p}^{\mathsf{T}}\mathbb{1} = 1} \tag{1.102b}$$

Observe that if we set $C_A = 0$ and $\bar{A} = A$, then we recover the synchronous case information, namely, $\bar{p} = p$ and $p_c = p \otimes p$.

The entries of the Perron vectors $\{p_c, \bar{p}\}$ are related to each other, as illustrated by the following explanation (proofs appear in [3][App. VII]). Since the vector p_c is of

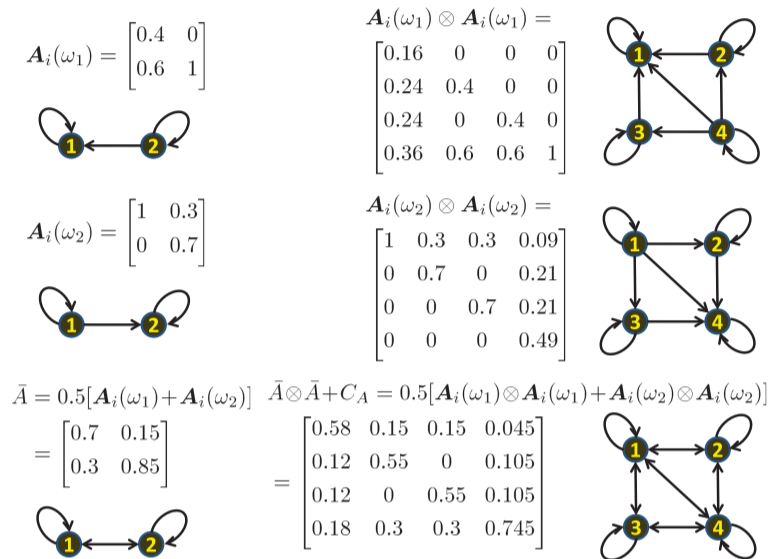**46**   **CHAPTER 1** Asynchronous Adaptive Networks

FIGURE 1.7   Digraphs.

The combination policy A_i has two equally probable realizations in this example, denoted by $\{A_i(\omega_1), A_i(\omega_2)\}$. Observe that neither of the digraphs $A_i(\omega_1) \otimes A_i(\omega_1)$ or $A_i(\omega_2) \otimes A_i(\omega_2)$ is strongly-connected due to the existence of the source and sink nodes. However, the digraph associated with $\mathbb{E}(A_i \otimes A_i)$, which is the union of the first two digraphs, is strongly-connected, where information can flow in any direction through the network. *Source: This is a modified version of a Figure extracted with permission from [3].*

dimension $N^2 \times 1$, we partition it into N sub-vectors of dimension $N \times 1$ each:

$$p_c \triangleq \mathrm{col}\{p_1, p_2, \ldots, p_N\} \tag{1.103a}$$

where p_k denotes the k-th sub-vector. We construct an $N \times N$ matrix P_c from these sub-vectors:

$$P_c \triangleq [p_1 \ \ p_2 \ \ \ldots \ \ p_N] = [p_{c,\ell k}]_{k,\ell=1}^N \tag{1.103b}$$

We use $p_{\ell k}$ to denote the (ℓ, k)-th element of matrix P_c, which is equal to the ℓ-th element of p_k. It can be verified that the matrix P_c in (1.103b) is symmetric positive semi-definite and it satisfies

$$\boxed{P_c \mathbb{1} = \bar{p}, \quad P_c^\mathsf{T} = P_c, \quad P_c \geq 0} \tag{1.103c}$$

where $\bar{p}$ is the Perron eigenvector in (1.102b). We can further establish the following

useful relations:

$$p_{c,\ell k} = p_{c,k\ell}, \quad \sum_{k=1}^{N} p_{c,\ell k} = \bar{p}_\ell, \quad \sum_{\ell=1}^{N} p_{c,\ell k} = \bar{p}_k \tag{1.103d}$$

We can also verify that the matrix difference

$$C_c \overset{\Delta}{=} P_c - \bar{p}\bar{p}^\mathsf{T}, \quad C_c^\mathsf{T} = C_c, \quad C_c \geq 0 \tag{1.103e}$$

is symmetric, positive semi-definite, and satisfies $C_c \mathbb{1} = 0$. Moreover, it is straight-forward to verify that

$$\boxed{c_{c,kk} = p_{c,kk} - \bar{p}_k^2 \geq 0} \tag{1.104}$$

where $c_{c,kk}$ denotes the (k,k)-th entry in C_c.

1.6 ASYNCHRONOUS NETWORK PERFORMANCE

We now comment on the performance of asynchronous networks and compare their metrics against both non-cooperative and centralized strategies.

1.6.1 MSD PERFORMANCE

We denote the MSD performance of the individual agents and the average MSD performance across the network by:

$$\text{MSD}_{\text{dist},k} \overset{\Delta}{=} \lim_{i \to \infty} \mathbb{E} \|\widetilde{\boldsymbol{w}}_{k,i}\|^2 \tag{1.105a}$$

$$\text{MSD}_{\text{dist,av}} \overset{\Delta}{=} \frac{1}{N} \sum_{k=1}^{N} \text{MSD}_{\text{dist},k} \tag{1.105b}$$

where the error vectors are measured relative to the global optimizer, w^o. We further denote the gradient noise process at the individual agents by

$$\boldsymbol{s}_{k,i}(w) \overset{\Delta}{=} \widehat{\nabla_{w^\mathsf{T}} J_k}(w) - \nabla_{w^\mathsf{T}} J_k(w) \tag{1.106a}$$

and define

$$H_k \overset{\Delta}{=} \nabla_w^2 J_k(w^o) \tag{1.106b}$$

$$R_{s,k} \overset{\Delta}{=} \lim_{i \to \infty} \mathbb{E}\left[\boldsymbol{s}_{k,i}(w^o)\boldsymbol{s}_{k,i}^\mathsf{T}(w^o) \,|\, \mathcal{F}_{i-1}\right] \tag{1.106c}$$

where $\mathcal{F}_{i-1}$ now represents the collection of all random events generated by the iterates from across all agents, $\{\boldsymbol{w}_{k,j},\ k = 1, 2, \ldots, N\}$, up to time $i-1$:

$$\mathcal{F}_{i-1} \overset{\Delta}{=} \text{filtration}\{\boldsymbol{w}_{k,-1}, \boldsymbol{w}_{k,0}, \boldsymbol{w}_{k,1}, \ldots, \boldsymbol{w}_{k,i-1},\ \text{all } k\} \tag{1.106d}$$

48   **CHAPTER 1** Asynchronous Adaptive Networks

It is stated later in (1.149a), under some assumptions on the gradient noise processes (1.106a), that for strongly-connected asynchronous diffusion networks of the form (1.86), and for sufficiently small step-sizes μ_x [3, 5]:

$$\mathrm{MSD}_{\mathrm{dist},k}^{\mathrm{asyn}} \approx \mathrm{MSD}_{\mathrm{dist,av}}^{\mathrm{asyn}} \approx \frac{1}{2}\mathrm{Tr}\left[\left(\sum_{k=1}^{N}\bar{\mu}_k\bar{p}_k H_k\right)^{-1}\left(\sum_{k=1}^{N}(\bar{\mu}_k^2+\sigma_{\mu,k}^2)p_{c,kk}R_{s,k}\right)\right] \quad (1.107)$$

The same result holds for asynchronous consensus and CTA diffusion strategies. Observe from (1.107) the interesting conclusion that the distributed strategy is able to *equalize* the MSD performance across all agents for sufficiently small step-sizes. It is also instructive to compare expression (1.149a) with (1.63) and (1.67) in the centralized case. Observe how cooperation among agents leads to the appearance of the scaling coefficients $\{\bar{p}_k, p_{c,kk}\}$; these factors are determined by $\bar{A}$ and C_A.

Note further that if we set $\bar{\mu}_k = \mu$, $\sigma_{\mu,k}^2 = 0$, $\bar{p}_k = p_k$ and $p_{c,kk} = p_k^2$, then we recover the MSD expression for synchronous distributed strategies:

$$\mathrm{MSD}_{\mathrm{dist},k}^{\mathrm{sync}} \approx \mathrm{MSD}_{\mathrm{dist,av}}^{\mathrm{sync}} \approx \frac{1}{2}\mathrm{Tr}\left[\left(\sum_{k=1}^{N}\mu_k p_k H_k\right)^{-1}\left(\sum_{k=1}^{N}\mu_k^2 p_k^2 R_{s,k}\right)\right] \quad (1.108)$$

This result agrees with expression (62) from [1].

Example 20 (**MSE networks with random updates**). We continue with the setting of Example 13, which deals with MSE networks. We assume the first and second-order moments of the random step-sizes are uniform, i.e., $\bar{\mu}_k \equiv \bar{\mu}$ and $c_{\mu,kk} \equiv \sigma_{\mu}^2$, and also assume uniform regression covariance matrices, i.e., $R_{u,k} \equiv R_u$ for $k = 1, 2, \ldots, N$. It follows that $H_k = 2R_u \equiv H$ and $R_{s,k} = 4\sigma_{v,k}^2 R_u$. Substituting into (1.149a), and assuming a fixed topology with fixed combination coefficients set to $a_{\ell k}$, we conclude that the MSD performance of the diffusion strategy (1.91a) with random updates is well approximated by:

$$\mathrm{MSD}_{\mathrm{dist},k}^{\mathrm{asyn},1} \approx \mathrm{MSD}_{\mathrm{dist,av}}^{\mathrm{asyn},1} \approx \mu_x M\left(\sum_{k=1}^{N}p_k^2\sigma_{v,k}^2\right) \quad (1.109a)$$

where

$$\mu_x = \bar{\mu} + \frac{\sigma_{\mu}^2}{\bar{\mu}} \quad (1.109b)$$

and p_k is the k−th entry of the Perron vector p defined by (1.100).

If the combination matrix A happens to be doubly stochastic, then its Perron eigenvector becomes $p = \mathbb{1}/N$. Substituting $p_k = 1/N$ into (1.109a) gives

$$\mathrm{MSD}_{\mathrm{dist},k}^{\mathrm{asyn},1} \approx \mathrm{MSD}_{\mathrm{dist,av}}^{\mathrm{asyn},1} \approx \frac{\mu_x M}{N}\left(\frac{1}{N}\sum_{k=1}^{N}\sigma_{v,k}^2\right) \quad (1.109c)$$

which agrees with the centralized performance (1.51b). In other words, the asynchronous diffusion strategy is able to match the performance of the centralized solution for doubly stochastic combination policies, when both implementations employ random updates. Since the centralized solution can improve the average MSD performance over non-cooperative networks, we further conclude that the diffusion strategy can also exceed the average performance of non-cooperative networks.

∎

Example 21 (Asynchronous MSE networks). We continue with the setting of Example 20 except that we now employ the asynchronous LMS diffusion network (1.91b). Its MSD performance is well approximated by:

$$\text{MSD}^{\text{asyn}}_{\text{dist},k} \approx \text{MSD}^{\text{asyn}}_{\text{dist,av}} \approx \mu_x M \left(\sum_{k=1}^{N} p_{c,kk}\sigma^2_{v,k} \right) \tag{1.110a}$$

If the mean combination matrix $\bar{A}$ happens to be doubly stochastic, then its Perron eigenvector becomes $\bar{p} = \mathbb{1}/N$. Substituting $\bar{p}_k = 1/N$ into (1.109a), and using $p_{c,kk} = \bar{p}^2_k + c_{c,kk}$, where $c_{c,kk}$ is from (1.103e), gives

$$\text{MSD}^{\text{asyn}}_{\text{dist},k} \approx \text{MSD}^{\text{asyn}}_{\text{dist,av}} \approx \frac{\mu_x M}{N} \left[\frac{1}{N} \sum_{k=1}^{N} (1 + N^2 c_{c,kk})\sigma^2_{v,k} \right] \tag{1.110b}$$

It is clear that if $c_{c,kk} = \sigma^2_{\pi,k}$, then the MSD performance in (1.110b) will agree with the centralized performance (1.55a). In other words, the distributed diffusion strategy is able to match the performance of the centralized solution.

∎

1.7 **NETWORK STABILITY AND PERFORMANCE**

In this section, we examine more closely the performance and stability results that were alluded to in the earlier sections. We first examine the consensus and diffusion strategies in a unified manner, and subsequently focus on diffusion strategies due to their enhanced stability properties, as the ensuing discussion will reveal.

1.7.1 **MSE NETWORKS**

We motivate the discussion by presenting first some illustrative examples with MSE networks, which involve quadratic costs. Following the examples, we extend the framework to more general costs.

Example 22 (Error dynamics over MSE networks). We consider the MSE network of Example 13, which involves quadratic costs with a common minimizer, w^o. The update equations for the non-cooperative, consensus, and diffusion strategies are given by (1.68a), (1.81), and (1.83a)–(1.83b). We can group these strategies into a single unifying description by considering the following structure in terms of three sets of combination coefficients $\{a_{o,\ell k}(i), a_{1,\ell k}(i), a_{2,\ell k}(i)\}$:

$$\begin{cases} \boldsymbol{\phi}_{k,i-1} = \sum_{\ell \in \boldsymbol{N}_{k,i}} a_{1,\ell k}(i)\boldsymbol{w}_{\ell,i-1} \\ \boldsymbol{\psi}_{k,i} = \sum_{\ell \in \boldsymbol{N}_{k,i}} a_{o,\ell k}(i)\boldsymbol{\phi}_{\ell,i-1} + 2\mu_k(i)\boldsymbol{u}^{\mathsf{T}}_{k,i}\left[\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}\boldsymbol{\phi}_{k,i-1}\right] \\ \boldsymbol{w}_{k,i} = \sum_{\ell \in \boldsymbol{N}_{k,i}} a_{2,\ell k}(i)\boldsymbol{\psi}_{\ell,i} \end{cases} \tag{1.111}$$

In (1.111), the quantities $\{\boldsymbol{\phi}_{k,i-1}, \boldsymbol{\psi}_{k,i}\}$ denote $M \times 1$ intermediate variables, while the nonnegative entries of the $N \times N$ matrices $A_{o,i} = [a_{o,\ell k}(i)]$, $A_{1,i} = [a_{1,\ell k}(i)]$, and $A_{2,i} = [a_{2,\ell k}(i)]$ are assumed to satisfy the same conditions (1.89). Any of the combination weights $\{a_{o,\ell k}(i), a_{1,\ell k}(i), a_{2,\ell k}(i)\}$ is zero whenever $\ell \notin$

50   **CHAPTER 1** Asynchronous Adaptive Networks

$\mathcal{N}_{k,i}$. Different choices for $\{A_{o,i}, A_{1,i}, A_{2,i}\}$, including random and deterministic choices, correspond to different strategies, as the following examples reveal:

$$\text{non-cooperative:}\quad A_{1,i} = A_{o,i} = A_{2,i} = I_N \tag{1.112a}$$

$$\text{consensus:}\quad A_{o,i} = A_i,\ A_{1,i} = I_N = A_{2,i} \tag{1.112b}$$

$$\text{CTA diffusion:}\quad A_{1,i} = A_i,\ A_{2,i} = I_N = A_{o,i} \tag{1.112c}$$

$$\text{ATC diffusion:}\quad A_{2,i} = A_i,\ A_{1,i} = I_N = A_{o,i} \tag{1.112d}$$

where A_i denotes some generic combination policy satisfying (1.89). We associate with each agent k the following three errors:

$$\widetilde{w}_{k,i} \overset{\Delta}{=} w^o - w_{k,i} \tag{1.113a}$$

$$\widetilde{\psi}_{k,i} \overset{\Delta}{=} w^o - \psi_{k,i} \tag{1.113b}$$

$$\widetilde{\phi}_{k,i-1} \overset{\Delta}{=} w^o - \phi_{k,i-1} \tag{1.113c}$$

which measure the deviations from the global minimizer, w^o. Subtracting w^o from both sides of the equations in (1.111) we get

$$\begin{cases} \widetilde{\phi}_{k,i-1} = \displaystyle\sum_{\ell\in\mathcal{N}_{k,i}} a_{1,\ell k}(i)\,\widetilde{w}_{\ell,i-1} \\[2mm] \widetilde{\psi}_{k,i} = \displaystyle\sum_{\ell\in\mathcal{N}_{k,i}} a_{o,\ell k}(i)\widetilde{\phi}_{\ell,i-1} - 2\mu_k(i)u_{k,i}^{\mathsf{T}}u_{k,i}\widetilde{\phi}_{k,i-1} - 2\mu_k(i)u_{k,i}^{\mathsf{T}}v_k(i) \\[2mm] \widetilde{w}_{k,i} = \displaystyle\sum_{\ell\in\mathcal{N}_{k,i}} a_{2,\ell k}(i)\,\widetilde{\psi}_{\ell,i} \end{cases} \tag{1.114a}$$

In a manner similar to (1.16a), the gradient noise process at each agent k is given by

$$s_{k,i}(\phi_{k,i-1}) = 2\left(R_{u,k} - u_{k,i}^{\mathsf{T}}u_{k,i}\right)\widetilde{\phi}_{k,i-1} - 2u_{k,i}^{\mathsf{T}}v_k(i) \tag{1.114b}$$

In order to examine the evolution of the error dynamics across the network, we collect the error vectors from all agents into $N \times 1$ block error vectors (whose individual entries are of size $M \times 1$ each):

$$\widetilde{w}_i \overset{\Delta}{=} \begin{bmatrix} \widetilde{w}_{1,i} \\ \widetilde{w}_{2,i} \\ \vdots \\ \widetilde{w}_{N,i} \end{bmatrix},\ \widetilde{\psi}_i \overset{\Delta}{=} \begin{bmatrix} \widetilde{\psi}_{1,i} \\ \widetilde{\psi}_{2,i} \\ \vdots \\ \widetilde{\psi}_{N,i} \end{bmatrix},\ \widetilde{\phi}_{i-1} \overset{\Delta}{=} \begin{bmatrix} \widetilde{\phi}_{1,i-1} \\ \widetilde{\phi}_{2,i-1} \\ \vdots \\ \widetilde{\phi}_{N,i-1} \end{bmatrix} \tag{1.115a}$$

Motivated by the last term in the second equation in (1.114a), and by the gradient noise terms (1.114b), we also introduce the following $N \times 1$ column vectors whose entries are of size $M \times 1$ each:

$$z_i \overset{\Delta}{=} \begin{bmatrix} 2u_{1,i}^{\mathsf{T}}v_1(i) \\ 2u_{2,i}^{\mathsf{T}}v_2(i) \\ \vdots \\ 2u_{N,i}^{\mathsf{T}}v_N(i) \end{bmatrix},\quad s_i \overset{\Delta}{=} \begin{bmatrix} s_{1,i}(\phi_{1,i-1}) \\ s_{2,i}(\phi_{2,i-1}) \\ \vdots \\ s_{N,i}(\phi_{N,i-1}) \end{bmatrix} \tag{1.115b}$$

We further introduce the Kronecker products

$$\mathcal{A}_{o,i} \overset{\Delta}{=} A_{o,i} \otimes I_M,\quad \mathcal{A}_{1,i} \overset{\Delta}{=} A_{1,i} \otimes I_M,\quad \mathcal{A}_{2,i} \overset{\Delta}{=} A_{2,i} \otimes I_M \tag{1.116a}$$

and the following $N \times N$ *block* diagonal matrices, whose individual entries are of size $M \times M$ each:

$$\boldsymbol{\mathcal{M}}_i \triangleq \text{diag}\{ \, \mu_1(i)I_M, \, \mu_2(i)I_M, \, \ldots, \, \mu_N(i)I_M \, \} \tag{1.116b}$$

$$\boldsymbol{\mathcal{R}}_i \triangleq \text{diag}\left\{ \, 2\boldsymbol{u}_{1,i}^\mathsf{T}\boldsymbol{u}_{1,i}, \, 2\boldsymbol{u}_{2,i}^\mathsf{T}\boldsymbol{u}_{2,i}, \, \ldots, \, 2\boldsymbol{u}_{N,i}^\mathsf{T}\boldsymbol{u}_{N,i} \, \right\} \tag{1.116c}$$

From (1.114a) we can then easily conclude that the block network variables satisfy the recursions:

$$\begin{cases} \widetilde{\boldsymbol{\phi}}_{i-1} &=& \boldsymbol{\mathcal{A}}_{1,i}^\mathsf{T}\widetilde{\boldsymbol{w}}_{i-1} \\ \widetilde{\boldsymbol{\psi}}_i &=& \left(\boldsymbol{\mathcal{A}}_{o,i}^\mathsf{T} - \boldsymbol{\mathcal{M}}_i\boldsymbol{\mathcal{R}}_i \right)\widetilde{\boldsymbol{\phi}}_{i-1} - \boldsymbol{\mathcal{M}}_i\boldsymbol{z}_i \\ \widetilde{\boldsymbol{w}}_i &=& \boldsymbol{\mathcal{A}}_{2,i}^\mathsf{T}\widetilde{\boldsymbol{\psi}}_i \end{cases} \tag{1.117a}$$

so that the network weight error vector, $\widetilde{\boldsymbol{w}}_i$, evolves according to:

$$\widetilde{\boldsymbol{w}}_i = \boldsymbol{\mathcal{A}}_{2,i}^\mathsf{T}\left(\boldsymbol{\mathcal{A}}_{o,i}^\mathsf{T} - \boldsymbol{\mathcal{M}}_i\boldsymbol{\mathcal{R}}_i \right)\boldsymbol{\mathcal{A}}_{1,i}^\mathsf{T}\widetilde{\boldsymbol{w}}_{i-1} - \boldsymbol{\mathcal{A}}_{2,i}^\mathsf{T}\boldsymbol{\mathcal{M}}_i\boldsymbol{z}_i \tag{1.117b}$$

For comparison purposes, if each agent operates individually and uses the non-cooperative strategy (1.68a), then the weight error vector would instead evolve according to the following recursion:

$$\widetilde{\boldsymbol{w}}_i = (I_{MN} - \boldsymbol{\mathcal{M}}_i\boldsymbol{\mathcal{R}}_i)\widetilde{\boldsymbol{w}}_{i-1} - \boldsymbol{\mathcal{M}}_i\boldsymbol{z}_i, \quad i \geq 0 \tag{1.118}$$

where the matrices $\{\boldsymbol{\mathcal{A}}_{o,i}, \boldsymbol{\mathcal{A}}_{1,i}, \boldsymbol{\mathcal{A}}_{2,i}\}$ do not appear any longer, and with a block diagonal coefficient matrix $(I_{MN} - \boldsymbol{\mathcal{M}}_i\boldsymbol{\mathcal{R}}_i)$. It is also straightforward to verify that recursion (1.117b) can be equivalently rewritten in the following form in terms of the gradient noise vector, $\boldsymbol{s}_i$, defined by (1.115b):

$$\widetilde{\boldsymbol{w}}_i = \boldsymbol{\mathcal{B}}_i\widetilde{\boldsymbol{w}}_{i-1} + \boldsymbol{\mathcal{A}}_{2,i}^\mathsf{T}\boldsymbol{\mathcal{M}}\boldsymbol{s}_i \tag{1.119a}$$

where

$$\boldsymbol{\mathcal{B}}_i \triangleq \boldsymbol{\mathcal{A}}_{2,i}^\mathsf{T}\left(\boldsymbol{\mathcal{A}}_{o,i}^\mathsf{T} - \boldsymbol{\mathcal{M}}_i\boldsymbol{\mathcal{R}} \right)\boldsymbol{\mathcal{A}}_{1,i}^\mathsf{T} \tag{1.119b}$$

$$\boldsymbol{\mathcal{R}} \triangleq \mathbb{E}\,\boldsymbol{\mathcal{R}}_i = \text{diag}\{2R_{u,1}, \, 2R_{u,2}, \, \ldots, \, 2R_{u,N}\} \tag{1.119c}$$

∎

Example 23 (Mean-error behavior). We continue with the setting of Example 22. In mean-square-error analysis, we are interested in examining how the quantities $\mathbb{E}\,\widetilde{\boldsymbol{w}}_i$ and $\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_i\|^2$ evolve over time. If we refer back to the data model described in Example 13, where the regression data $\{\boldsymbol{u}_{k,i}\}$ were assumed to be temporally white and independent over space, then the stochastic matrix $\boldsymbol{\mathcal{R}}_i$ appearing in (1.117b)–(1.118) is seen to be statistically independent of $\widetilde{\boldsymbol{w}}_{i-1}$. We further assume that, in the unified formulation, the entries of the combination policies $\{A_{o,i}, A_{1,i}, A_{2,i}\}$ are independent of each other (as well as over time) and of any other variable in the learning algorithm. Therefore, taking expectations of both sides of these recursions, and invoking the fact that $\boldsymbol{u}_{k,i}$ and $\boldsymbol{v}_k(i)$ are also independent of each other and have zero means (so that $\mathbb{E}\,\boldsymbol{z}_i = 0$), we conclude that the mean-error vectors evolve according to the following recursions:

$$\mathbb{E}\,\widetilde{\boldsymbol{w}}_i = \bar{\mathcal{B}}\,(\mathbb{E}\,\widetilde{\boldsymbol{w}}_{i-1}) \qquad \text{(distributed)} \tag{1.120a}$$

$$\mathbb{E}\,\widetilde{\boldsymbol{w}}_i = \left(I_{MN} - \bar{\mathcal{M}}\mathcal{R} \right)(\mathbb{E}\,\widetilde{\boldsymbol{w}}_{i-1}) \qquad \text{(non-cooperative)} \tag{1.120b}$$

52 **CHAPTER 1** Asynchronous Adaptive Networks

where

$$\bar{\mathcal{B}} \;\triangleq\; \mathbb{E}\,\mathcal{B}_i \;=\; \bar{\mathcal{A}}_2^\mathsf{T}\left(\bar{\mathcal{A}}_o^\mathsf{T} - \bar{\mathcal{M}}\mathcal{R}\right)\bar{\mathcal{A}}_1^\mathsf{T} \tag{1.121a}$$

$$\bar{\mathcal{M}} \;=\; \mathbb{E}\,\mathcal{M}_i = \mathrm{diag}\{\,\bar{\mu}_1 I_M,\,\ldots,\,\bar{\mu}_N I_M\,\} \tag{1.121b}$$

$$\bar{\mathcal{A}}_o \;=\; \mathbb{E}\,\mathcal{A}_{o,i} \tag{1.121c}$$

$$\bar{\mathcal{A}}_1 \;=\; \mathbb{E}\,\mathcal{A}_{1,i} \tag{1.121d}$$

$$\bar{\mathcal{A}}_2 \;=\; \mathbb{E}\,\mathcal{A}_{2,i} \tag{1.121e}$$

The matrix $\bar{\mathcal{B}}$ controls the dynamics of the mean weight-error vector for the distributed strategies. Observe, in particular, that $\bar{\mathcal{B}}$ reduces to the following forms for the various strategies (non-cooperative (1.68a), consensus (1.81), and diffusion (1.83a)–(1.83b)):

$$\bar{\mathcal{B}}_{\mathrm{ncop}} \;=\; I_{MN} - \bar{\mathcal{M}}\mathcal{R} \tag{1.122a}$$

$$\bar{\mathcal{B}}_{\mathrm{cons}} \;=\; \bar{\mathcal{A}}^\mathsf{T} - \bar{\mathcal{M}}\mathcal{R} \tag{1.122b}$$

$$\bar{\mathcal{B}}_{\mathrm{atc}} \;=\; \bar{\mathcal{A}}^\mathsf{T}\left(I_{MN} - \bar{\mathcal{M}}\mathcal{R}\right) \tag{1.122c}$$

$$\bar{\mathcal{B}}_{\mathrm{cta}} \;=\; \left(I_{MN} - \bar{\mathcal{M}}\mathcal{R}\right)\bar{\mathcal{A}}^\mathsf{T} \tag{1.122d}$$

where $\bar{\mathcal{A}} = \bar{A} \otimes I_M$ and $\bar{A} = \mathbb{E}\,A_i$. ∎

Example 24 (**MSE networks with uniform agents**). The results of Example 23 simplify when all agents employ step-sizes with the same mean value, $\bar{\mu}_k \equiv \bar{\mu}$, and observe regression data with the same covariance matrix, $R_{u,k} \equiv R_u$ [7, 80]. In this case, we can express $\bar{\mathcal{M}}$ and $\mathcal{R}$ from (1.152b) and (1.119c) in Kronecker product form as follows:

$$\bar{\mathcal{M}} = \bar{\mu} I_N \otimes I_M, \qquad \mathcal{R} = I_N \otimes 2R_u \tag{1.123}$$

so that expressions (1.122a)–(1.122d) reduce to

$$\bar{\mathcal{B}}_{\mathrm{ncop}} \;=\; I_N \otimes (I_M - 2\bar{\mu} R_u) \tag{1.124a}$$

$$\bar{\mathcal{B}}_{\mathrm{cons}} \;=\; \bar{A}^\mathsf{T} \otimes I_M - 2\bar{\mu}(I_M \otimes R_u) \tag{1.124b}$$

$$\bar{\mathcal{B}}_{\mathrm{atc}} \;=\; \bar{A}^\mathsf{T} \otimes (I_M - 2\bar{\mu} R_u) \tag{1.124c}$$

$$\bar{\mathcal{B}}_{\mathrm{cta}} \;=\; \bar{A}^\mathsf{T} \otimes (I_M - 2\bar{\mu} R_u) \tag{1.124d}$$

Observe that $\bar{\mathcal{B}}_{\mathrm{atc}} = \bar{\mathcal{B}}_{\mathrm{cta}}$, so we denote these matrices by $\bar{\mathcal{B}}_{\mathrm{diff}}$. Using properties of the eigenvalues of Kronecker products of matrices, it can be easily verified that the MN eigenvalues of the above $\bar{\mathcal{B}}$ matrices are given by the following expressions in terms of the eigenvalues of the component matrices $\{\bar{A}, R_u\}$ for $k = 1, 2, \ldots N$ and $m = 1, 2, \ldots, M$:

$$\lambda(\bar{\mathcal{B}}_{\mathrm{ncop}}) \;=\; 1 - 2\bar{\mu}\lambda_m(R_u) \tag{1.125a}$$

$$\lambda(\bar{\mathcal{B}}_{\mathrm{cons}}) \;=\; \lambda_k(\bar{A}) - 2\bar{\mu}\lambda_m(R_u) \tag{1.125b}$$

$$\lambda(\bar{\mathcal{B}}_{\mathrm{diff}}) \;=\; \lambda_k(\bar{A})\left[1 - 2\bar{\mu}\lambda_m(R_u)\right] \tag{1.125c}$$

∎

Example 25 (**Potential instability in consensus networks**). Consensus strategies can become unstable when used for adaptation purposes [80, 5]. This undesirable effect is already reflected in expressions (1.125a)–(1.125c). In particular, observe that the eigenvalues of $\bar{A}$ appear multiplying $(1 - 2\mu\lambda_m(R_u))$ in expression (1.125c) for diffusion. As such, and since $\rho(\bar{A}) = 1$ for any left-stochastic matrix, we conclude

for this case of uniform agents that $\rho(\bar{\mathcal{B}}_{\text{diff}}) = \rho(\bar{\mathcal{B}}_{\text{ncop}})$. It follows that, regardless of the choice of the mean combination policy $\bar{A}$, the diffusion strategies will be stable in the mean (i.e., $\mathbb{E}\,\widetilde{w}_i$ will converge asymptotically to zero) whenever the individual non-cooperative agents are stable in the mean:

$$\text{individual agents stable} \implies \text{diffusion networks stable} \tag{1.126a}$$

The same conclusion is not true for consensus networks; the individual agents can be stable and yet the consensus network can become unstable. This is because $\lambda_k(\bar{A})$ appears as an additive (rather than multiplicative) term in (1.125b) (see [6, 80, 5] for examples):

$$\text{individual agents stable} \;\not\Rightarrow\; \text{consensus networks stable} \tag{1.126b}$$

The fact that the combination matrix $\bar{\mathcal{A}}^{\mathsf{T}}$ appears in an additive form in (1.122b) is the result of the asymmetry that was mentioned earlier in the update equation for the consensus strategy. In contrast, the update equations for the diffusion strategies lead to $\bar{\mathcal{A}}^{\mathsf{T}}$ appearing in a multiplicative form in (1.122c)–(1.122d). ∎

Example 26 (**Useful stability result**). It is observed from expressions (1.122c)–(1.122d) in the asynchronous case, as well as from the corresponding expressions (81c)–(81d) in the synchronous case studied in [1], that the mean stability of diffusion strategies usually involves examining the stability of a matrix product of the form:

$$\mathcal{B} \stackrel{\Delta}{=} \mathcal{A}_2^{\mathsf{T}} \mathcal{D} \mathcal{A}_1^{\mathsf{T}} \tag{1.127}$$

where $\mathcal{D}$ is a block diagonal symmetric matrix with blocks of size $M \times M$, while $\mathcal{A}_1$ and $\mathcal{A}_2$ are Kronecker product matrices defined in terms of $N \times N$ left-stochastic matrices A_1 and A_2 as $\mathcal{A}_1 = A_1 \otimes I_M$ and $\mathcal{A}_2 = A_2 \otimes I_M$. For example, in (1.122c) we have $A_1 = I_N$, $A_2 = \bar{A}$, $\mathcal{A}_1 = I_{MN}$, $\mathcal{A}_2 = \bar{\mathcal{A}}$, and $\mathcal{D} = I_{MN} - \mathcal{MR}$.

Matrix products of the form (1.127) are induced by the cooperation mechanism that is inherent to diffusion learning. They have a useful property: it turns out that these matrix products are stable, regardless of A_1 and A_2, as long as $\mathcal{D}$ is stable (i.e., has all its eigenvalues strictly inside the unit disc). This useful result is easy to establish for *symmetric* left-stochastic matrices A_1 and A_2, as already noted in [78]. This is because for symmetric matrices, their spectral radii coincide with their 2−induced norms and, hence,

$$\rho(A_1) = \|A_1\|, \quad \rho(A_2) = \|A_2\| \tag{1.128}$$

Consequently, since we already know that $\rho(A_1) = \rho(A_2) = 1$, it follows that

$$\begin{aligned}
\rho(\mathcal{B}) &\leq & \|\mathcal{B}\| \\
&\leq & \|\mathcal{A}_2\| \cdot \|\mathcal{D}\| \cdot \|\mathcal{A}_1\| \\
&= & \rho(\mathcal{A}_2) \cdot \rho(\mathcal{D}) \cdot \rho(\mathcal{A}_1) \\
&= & \rho(A_2) \cdot \rho(\mathcal{D}) \cdot \rho(A_1) \\
&= & \rho(\mathcal{D})
\end{aligned} \tag{1.129}$$

which confirms that a stable $\mathcal{D}$ guarantees a stable $\mathcal{B}$, regardless of A_1 and A_2.

The conclusion that $\mathcal{B}$ in (1.127) is stable whenever $\mathcal{D}$ is stable continues to hold even when the matrices A_1 and A_2 are *not* necessarily symmetric. However, the argument leading to (1.129) will need to be adjusted because property (1.128) need not hold anymore. This more general result was established in [7, App. D] and also in [105, App. A, pp. 3471–3473], where it was shown that multiplication of a symmetric block diagonal matrix $\mathcal{D}$ by any (not necessarily symmetric) left-stochastic Kronecker-product transformations from left and right generally reduces the spectral radius, i.e.,

$$\rho\left(\mathcal{A}_2^{\mathsf{T}} \mathcal{D} \mathcal{A}_1^{\mathsf{T}}\right) \leq \rho(\mathcal{D}) \tag{1.130}$$

Accordingly, a stable $\mathcal{D}$ again ensures a stable $\mathcal{B}$. This conclusion was established in the above references

by replacing the 2−induced norm used to arrive at (1.129) by a more convenient block-maximum norm, denoted by $\| \cdot \|_{b,\infty}$ and defined as follows.

Let $x = \mathrm{col}\{x_1, x_2, \ldots, x_N\}$ denote an $N \times 1$ *block* column vector whose individual entries are themselves vectors of size $M \times 1$ each. Following [36, 106, 7], the block maximum norm of x is denoted by $\|x\|_{b,\infty}$ and is defined as

$$\|x\|_{b,\infty} \overset{\Delta}{=} \max_{1 \leq k \leq N} \|x_k\| \tag{1.131}$$

That is, $\|x\|_{b,\infty}$ is equal to the largest Euclidean norm of its block components (this definition extends the regular notion of the ∞−norm of a vector to block vectors). The vector norm (1.131) induces a block-maximum matrix norm. Let $\mathcal{A}$ denote an arbitrary $N \times N$ block matrix with individual block entries of size $M \times M$ each. Then, the block-maximum norm of $\mathcal{A}$ is defined as

$$\|\mathcal{A}\|_{b,\infty} \overset{\Delta}{=} \max_{x \neq 0} \frac{\|\mathcal{A}x\|_{b,\infty}}{\|x\|_{b,\infty}} \tag{1.132}$$

The block-maximum norm has several useful properties — see [7]. In particular, when A is $N \times N$ left-stochastic and $\mathcal{A} = A \otimes I_M$, then it can be verified that $\|\mathcal{A}^\mathsf{T}\|_{b,\infty} = 1$. Likewise, when $\mathcal{D}$ is block diagonal and symmetric, then $\|\mathcal{D}\|_{b,\infty} = \rho(\mathcal{D})$. Consequently, repeating the argument leading to (1.129) and replacing the 2−induced norm used there by the block-maximum norm we have

$$\rho(\mathcal{B}) \leq \|\mathcal{B}\|_{b,\infty} \leq \|\mathcal{A}_2^\mathsf{T}\|_{b,\infty} \cdot \|\mathcal{D}\|_{b,\infty} \cdot \|\mathcal{A}_1^\mathsf{T}\|_{b,\infty} = \rho(D) \tag{1.133}$$

and we again conclude that a stable symmetric $\mathcal{D}$ guarantees a stable $\mathcal{B}$ for general left-stochastic matrices A_1 and A_2 since, for symmetric $\mathcal{D}$, it holds that $\rho(\mathcal{D}) = \|\mathcal{D}\|_{b,\infty}$.

Remark 1. The same argument (1.133) can be used to relax the requirement of symmetry and stability on the block diagonal matrix $\mathcal{D}$. Actually, as long as $\|\mathcal{D}\|_{b,\infty} < 1$, which is guaranteed by requiring the block diagonal entries of $\mathcal{D}$ to have their 2−induced norms bounded by one, we can again conclude that $\rho(\mathcal{B}) < 1$ so that $\mathcal{B}$ is stable.

Remark 2. The validity of property (1.130) for general left-stochastic matrices was already noted and exploited in earlier works, e.g., in Lemma 1 of [79] and in Lemma 2 of [125]. However, the statement of Lemma 1 in [79] left out the qualification "diagonal" for the center matrix, and the norm $\| \cdot \|_\rho$ that was used in the proof of the lemma in Appendix I of [79] should be replaced by the $\| \cdot \|_\infty$−norm. Although these corrections were already noted in the references [7, 105], we restate below the correct form of Lemma 1 from [79] for accuracy, and provide its adjusted proof using the current notation. That lemma deals with left-stochastic matrices A_1 and A_2 prior to extension by Kronecker products. Its correct statement should read as follows (the word "diagonal" is missing from the statement in [79]).

Restatement of Lemma 1 [79]: Let A_1, A_2, and D denote arbitrary $N \times N$ matrices, where A_1 and A_2 have real non-negative entries, with columns adding up to one, i.e., $\mathbb{1}^\mathsf{T} A_1 = \mathbb{1}^\mathsf{T}$, $\mathbb{1}^\mathsf{T} A_2 = \mathbb{1}^\mathsf{T}$. Then, the matrix $B = A_2^\mathsf{T} D A_1^\mathsf{T}$ is stable for *any* choice of A_1 and A_2 if, and only if, the diagonal matrix D is stable.

Proof: One immediate derivation is to employ the ∞-norm and to note that

$$\|A_1^\mathsf{T}\|_\infty = 1 = \|A_2^\mathsf{T}\|_\infty, \quad \|D\|_\infty = \rho(D) \tag{1.134}$$

and, hence,

$$\rho(B) \leq \|B\|_\infty \leq \|A_2^\mathsf{T}\|_\infty \cdot \|D\|_\infty \cdot \|A_1^\mathsf{T}\|_\infty = \rho(D) \tag{1.135}$$

It follows that a stable D guarantees a stable B. A second derivation that fixes the argument from Appendix I of [79] relies on using the ∞-norm instead of the ρ-norm used there. Thus, note that we can alternatively

argue that

$$
\begin{aligned}
\|B^i\|_\infty &\leq \left(\|A_2^\mathsf{T}\|_\infty\right)^i \cdot (\|D\|_\infty)^i \cdot \left(\|A_1^\mathsf{T}\|_\infty\right)^i \\
&= (\rho(D))^i \longrightarrow 0, \text{ as } i \to \infty
\end{aligned}
\tag{1.136}
$$

when D is stable. This completes the proof of Lemma 1 from [79].

Remark 3. These same arguments establish the validity of Lemma 2 from [125], which deals with a special case involving a matrix of the form $\mathcal{B} = \mathcal{A}_2^\mathsf{T}\mathcal{D}$ using the current notation (the matrix in [125] is denoted by $\mathcal{F} = C^\mathsf{T}M$ with $\mathcal{B} \leftarrow \mathcal{F}$, $\mathcal{A}_2 \leftarrow C$, and $\mathcal{D} \leftarrow M$; moreover, the matrix M is now generally non-symmetric but still block diagonal with stable entries of the form $M_{kk} = (I - P_k S_k)F$ for some matrices $\{P_k, S_k, F\}$ defined in [125]. When $\|M_{kk}\|_2 < 1$, these block diagonal entries will have 2−induced norms smaller than one so that $\|\mathcal{D}\|_{b,\infty} < 1$. Again, the ρ−norm used in the proof of Lemma 2 in [125] should be replaced by the $\|\cdot\|_{b,\infty}$ norm and the argument adjusted as in (1.133) or, alternatively, we note that

$$
\|\mathcal{B}^i\|_{b,\infty} \leq \left(\|\mathcal{A}_2^\mathsf{T}\|_{b,\infty}\right)^i \cdot (\|\mathcal{D}\|_{b,\infty})^i \longrightarrow 0, \text{ as } i \to \infty
\tag{1.137}
$$

∎

1.7.2 **DIFFUSION NETWORKS**

Given the superior stability properties of diffusion strategies for adaptation and learning over networks, we continue our presentation by focusing on this class of algorithms. The results in the previous section focus on MSE networks, which deal with mean-square-error cost functions. We now consider networks with more general costs, $\{J_k(w)\}$, and apply diffusion strategies to seek the global minimizer, w^o, of the aggregate cost function, $J^{\mathrm{glob}}(w)$, defined by (1.56). Without loss in generality, we consider ATC diffusion implementations of the form (1.88):

$$
\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} - \boldsymbol{\mu}_k(i)\widehat{\nabla_{w^\mathsf{T}} J_k}(\boldsymbol{w}_{k,i-1})
\tag{1.138a}
$$

$$
\boldsymbol{w}_{k,i} = \sum_{\ell \in \boldsymbol{N}_{k,i}} \boldsymbol{a}_{\ell k}(i)\,\boldsymbol{\psi}_{\ell,i}
\tag{1.138b}
$$

Similar conclusions will apply to CTA diffusion implementations.

Assumption 7 (**Conditions on cost functions**). *The aggregate cost $J^{glob}(w)$ in (1.56) is twice differentiable and satisfies a condition similar to (1.9) in Assumption 1 for some positive parameters $\nu_d \leq \delta_d$. Moreover, all individual costs $\{J_k(w)\}$ are assumed to be strongly-convex with their global minimizers located at w^o, as indicated earlier by (1.73).*

∎

As explained before following (1.73), references [5, 1, 42] present results on the case in which the individual costs are only convex and need not be strongly convex. These references also discuss the case in which the individual costs need not share

56   **CHAPTER 1** Asynchronous Adaptive Networks

minimizers. With each agent k in (1.88), we again associate a gradient noise vector:

$$s_{k,i}(w_{k,i-1}) \triangleq \widehat{\nabla_{w^\mathsf{T}} J_k}(w_{k,i-1}) - \nabla_{w^\mathsf{T}} J_k(w_{k,i-1}) \tag{1.139}$$

Assumption 8 (**Conditions on gradient noise**). *It is assumed that the first and second-order conditional moments of the gradient noise components satisfy:*

$$\mathbb{E}\left[s_{k,i}(w_{k,i-1}) \,|\, \mathcal{F}_{i-1}\right] = 0 \tag{1.140a}$$

$$\mathbb{E}\left[s_{k,i}(w_{k,i-1}) s_{\ell,i}^\mathsf{T}(w_{\ell,i-1}) \,|\, \mathcal{F}_{i-1}\right] = 0, \quad \forall\, k \neq \ell \tag{1.140b}$$

$$\mathbb{E}\left[\|s_{k,i}(w_{k,i-1})\|^2 \,|\, \mathcal{F}_{i-1}\right] \leq \beta_k^2 \|\widetilde{w}_{k,i-1}\|^2 + \sigma_{s,k}^2 \tag{1.140c}$$

almost surely for some nonnegative scalars β_k^2 and $\sigma_{s,k}^2$, and where $\mathcal{F}_{i-1}$ represents the collection of all random events generated by the iterates from across all agents, $\{w_{\ell,j}, \ell = 1, 2, \ldots, N\}$, up to time $i - 1$. Moreover, it is assumed that the limiting covariance matrix of $s_{k,i}(w^o)$ exists:

$$R_{s,k} \triangleq \lim_{i \to \infty} \mathbb{E}\left[s_{k,i}(w^o) s_{k,i}^\mathsf{T}(w^o) \,|\, \mathcal{F}_{i-1}\right] \tag{1.140d}$$

∎

We collect the error vectors and gradient noises from across all agents into $N \times 1$ *block* vectors, whose individual entries are of size $M \times 1$ each:

$$\widetilde{w}_i \triangleq \begin{bmatrix} \widetilde{w}_{1,i} \\ \widetilde{w}_{2,i} \\ \vdots \\ \widetilde{w}_{N,i} \end{bmatrix}, \quad s_i \triangleq \begin{bmatrix} s_{1,i} \\ s_{2,i} \\ \vdots \\ s_{N,i} \end{bmatrix} \tag{1.141}$$

and where we are dropping the argument $w_{k,i-1}$ from the $s_{k,i}(\cdot)$ for compactness of notation. Likewise, we introduce the following $N \times N$ *block* diagonal matrices, whose individual entries are of size $M \times M$ each:

$$\mathcal{M}_i = \text{diag}\{ \mu_1(i)I_M, \mu_2(i)I_M, \ldots, \mu_N(i)I_M \} \tag{1.142a}$$

$$\mathcal{H}_{i-1} = \text{diag}\{ H_{1,i-1}, H_{2,i-1}, \ldots, H_{N,i-1} \} \tag{1.142b}$$

where

$$H_{k,i-1} \triangleq \int_0^1 \nabla_w^2 J_k(w^o - t\widetilde{w}_{k,i-1})dt \tag{1.142c}$$

Now, in a manner similar to (1.29b), we can appeal to the mean-value theorem [35, 49, 5] to note that

$$\nabla_{w^\mathsf{T}} J_k(w_{k,i-1}) = -H_{k,i-1}\widetilde{w}_{k,i-1} \tag{1.143}$$

so that the approximate gradient vector can be expressed as:

$$\widehat{\nabla_{w^\mathsf{T}} J}_k(w_{k,i-1}) = -\boldsymbol{H}_{k,i-1}\widetilde{w}_{k,i-1} + s_{k,i}(w_{k,i-1}) \tag{1.144}$$

Subtracting w^o from both sides of (1.138a)–(1.138b), and using (1.144), we find that the network error vector evolves according to the following stochastic recursion:

$$\boxed{\widetilde{\boldsymbol{w}}_i = \boldsymbol{\mathcal{B}}_{i-1}\widetilde{\boldsymbol{w}}_{i-1} + \boldsymbol{\mathcal{A}}_i^\mathsf{T}\boldsymbol{\mathcal{M}}_i s_i} \tag{1.145a}$$

where

$$\boldsymbol{\mathcal{B}}_{i-1} \stackrel{\Delta}{=} \boldsymbol{\mathcal{A}}_i^\mathsf{T}(I_{NM} - \boldsymbol{\mathcal{M}}_i\boldsymbol{\mathcal{H}}_{i-1}) \tag{1.145b}$$

Recursion (1.145a) describes the evolution of the network error vector for general convex costs, $J_k(w)$, in a manner similar to recursion (1.119a) in the mean-square-error case. However, recursion (1.145a) is more challenging to deal with because of the presence of the random matrix $\boldsymbol{\mathcal{H}}_{i-1}$; this matrix is replaced by the constant term $\mathcal{R}$ in the earlier recursion (1.119a) because that example deals with MSE networks where the individual costs, $J_k(w)$, are quadratic in w and, therefore, their Hessian matrices are constant and independent of w. In that case, each matrix $\boldsymbol{H}_{k,i-1}$ in (1.142c) will evaluate to $2R_{u,k}$ and the matrix $\boldsymbol{\mathcal{H}}_{i-1}$ in (1.145b) will coincide with the matrix $\mathcal{R}$ defined by (1.119c).

The next statement ascertains that sufficiently small step-sizes exist that guarantee the MSE stability of the asynchronous diffusion strategy (1.138a)–(1.138b) [2, 5].

Lemma 4 (**MSE network stability**). *Consider an asynchronous network of N interacting agents running the ATC diffusion strategy (1.138a)–(1.138b). Assume the conditions in Assumptions 6, 7, and 8 hold. Let*

$$\mu_{x,\max} = \max_{1\le k\le N} \{\mu_{x,k}\} = \max_{1\le k\le N}\left(\bar{\mu}_k + \frac{\sigma_{\mu,k}^2}{\bar{\mu}_k}\right) \tag{1.146}$$

Then, there exists $\mu_o > 0$ such that for all $\mu_{x,\max} < \mu_o$:

$$\limsup_{i\to\infty} \mathbb{E}\|\widetilde{w}_{k,i}\|^2 = O(\mu_{x,\max}) \tag{1.147}$$

Proof. See App. IV of [2].

$\square$

Result (1.147) shows that the MSD of the network is in the order of $\mu_{x,\max}$. Therefore, sufficiently small step-sizes lead to sufficiently small MSDs. As was the case with the discussion in subsection 1.2.7, we can also seek a closed-form expression for the MSD performance of the asynchronous diffusion network and its agents. To do that, we first introduce the analog of Assumption 4 for the network case.

58   CHAPTER 1 Asynchronous Adaptive Networks

Assumption 9 (**Smoothness Conditions**). *The Hessian matrix of the individual cost functions* $\{J_k(w)\}$, *and the noise covariance matrices defined for each agent in a manner similar to (1.17a) and denoted by* $R_{s,k,i}(w)$, *are assumed to be locally Lipschitz continuous in a small neighborhood around* w^o:

$$\|\nabla_w^2 J_k(w^o + \delta w) - \nabla_w^2 J_k(w^o)\| \;\leq\; \tau_{k,d}\|\delta w\| \tag{1.148a}$$

$$\|R_{s,k,i}(w^o + \delta w) - R_{s,k,i}(w^o)\| \;\leq\; \tau_{k,s}\|\delta w\|^{\kappa} \tag{1.148b}$$

for small perturbation $\|\delta w\| \leq r_d$ *and for some* $\tau_{k,d}, \tau_{k,s} \geq 0$ *and* $1 \leq \kappa \leq 2$. ∎

Lemma 5 (**Asynchronous network MSD performance**). *Consider an asynchronous network of N interacting agents running the asynchronous diffusion strategy (1.138a)–(1.138b). Assume the conditions under Assumptions 6, 7, 8, and 1 hold. Assume further that the step-size parameter* $\mu_{x,\max}$ *is sufficiently small to ensure mean-square stability, as already ascertained by Lemma 4. Then,*

$$\mathrm{MSD}_{\mathrm{diff},k}^{\mathrm{asyn}} \approx \mathrm{MSD}_{\mathrm{diff,av}}^{\mathrm{asyn}} \approx \frac{1}{2}\mathrm{Tr}\left[\left(\sum_{k=1}^{N} \bar{\mu}_k \bar{p}_k H_k\right)^{-1}\left(\sum_{k=1}^{N}(\bar{\mu}_k^2 + \sigma_{\mu,k}^2)p_{c,kk}R_{s,k}\right)\right] \tag{1.149a}$$

where $H_k = \nabla_w^2 J_k(w^o)$. *Moreover, for large enough i, the convergence rate towards the above steady-state value is well approximated by the scalar:*

$$\alpha_{\mathrm{dist}}^{\mathrm{asyn}} = 1 - 2\lambda_{\min}\left(\sum_{k=1}^{N} \bar{\mu}_k \bar{p}_k H_k\right) + O\left(\mu_{x,\max}^{1+1/N^2}\right) \tag{1.149b}$$

Proof. See App. XII in [3]. □

Example 27 (**Gaussian regression data**). MSE performance expressions of the form (1.149a) are accurate to first-order in the step-size parameters, i.e., they are on the order of $O(\bar{\mu}_k)$. The same is true of expression (112) from [1] in the synchronous case. There are situations, however, where exact expressions for the MSE performance can be derived for multi-agent networks. We illustrate this possibility here for the case of MSE networks of the type described earlier in Example 13. We consider first synchronous networks and comment later on how the results should be adjusted to handle asynchronous behavior.

We refer to the same setting of Example 9 from [1] where we have N agents observing streaming data $\{\boldsymbol{d}_k(i), \boldsymbol{u}_{k,i}\}$ that satisfy the regression model:

$$\boldsymbol{d}_k(i) \;=\; \boldsymbol{u}_{k,i}w^o + \boldsymbol{v}_k(i) \tag{1.150a}$$

We assume the regression vectors are zero-mean Gaussian-distributed with diagonal covariance matrices denoted by Λ_k, say,

$$R_{u,k} = \mathbb{E}\,\boldsymbol{u}_{k,i}^{\mathsf{T}}\boldsymbol{u}_{k,i} \;\stackrel{\Delta}{=}\; \Lambda_k > 0 \tag{1.150b}$$

We further assume that the regression data is temporally white and independent over space so that

$$\mathbb{E}\,\boldsymbol{u}_{k,i}^{\mathsf{T}}\boldsymbol{u}_{\ell,j} = R_{u,k}\delta_{k,\ell}\delta_{i,j} \tag{1.150c}$$

We also assume that the measurement noise process $\mathbf{v}_k(i)$ is temporally white and independent over space so that $\mathbb{E}\,\mathbf{v}_k(i)\mathbf{v}_\ell(j) = \sigma^2_{v,k}\delta_{k,\ell}\delta_{i,j}$ in terms of the Kronecker delta sequence $\delta_{m,n}$. Likewise, the measurement noise $\mathbf{v}_k(i)$ and the regression data $\mathbf{u}_{\ell,j}$ are assumed to be independent of each other for all k, ℓ, i, j.

The Gaussian assumption on the regression data is useful because, for this case, fourth-order moments of the regression vectors can be evaluated in closed-form (and such fourth-order moment calculations will arise in the process of determining closed-form expressions for the MSE). Specifically, for any matrix $Q \geq 0$ of size $M \times M$, it holds for independent *real-valued* Gaussian regressors that [28] (there is a typo in reproducing this expression in the second equation of App. II in [79]):

$$\mathbb{E}\,\boldsymbol{u}^{\mathsf{T}}_{k,i}\boldsymbol{u}_{k,i}Q\boldsymbol{u}^{\mathsf{T}}_{\ell,i}\boldsymbol{u}_{\ell,i} = R_{u,k}QR_{u,\ell} + \delta_{k,\ell}\left\{R_{u,k}\mathrm{Tr}\,(QR_{u,k}) + R_{u,k}QR_{u,k}\right\} \tag{1.150d}$$

We assume the network is running a diffusion strategy of the form:

$$\begin{cases} \boldsymbol{\phi}_{k,i-1} = \displaystyle\sum_{\ell\in\mathcal{N}_k} a_{1,\ell k}\boldsymbol{w}_{\ell,i-1} \\[2mm] \boldsymbol{\psi}_{k,i} = \boldsymbol{\phi}_{k,i-1} + 2\mu_k\boldsymbol{u}^{\mathsf{T}}_{k,i}\left[\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}\boldsymbol{\phi}_{k,i-1}\right] \\[2mm] \boldsymbol{w}_{k,i} = \displaystyle\sum_{\ell\in\mathcal{N}_k} a_{2,\ell k}\boldsymbol{\psi}_{\ell,i} \end{cases} \tag{1.151a}$$

which includes both the ATC and CTA LMS diffusion forms as special cases. We know from (77b) in [1] that the error network vector evolves according to the dynamics

$$\widetilde{\boldsymbol{w}}_i = \mathcal{A}^{\mathsf{T}}_2(I - \mathcal{M}\mathcal{R}_i)\mathcal{A}^{\mathsf{T}}_1\widetilde{\boldsymbol{w}}_{i-1} - \mathcal{A}^{\mathsf{T}}_2\mathcal{M}\boldsymbol{z}_i, \quad i \geq 0 \tag{1.151b}$$

which is defined in terms of the following $N \times 1$ column vectors whose entries are of size $M \times 1$ each:

$$\boldsymbol{z}_i \triangleq \begin{bmatrix} 2\boldsymbol{u}^{\mathsf{T}}_{1,i}\boldsymbol{v}_1(i) \\ 2\boldsymbol{u}^{\mathsf{T}}_{2,i}\boldsymbol{v}_2(i) \\ \vdots \\ 2\boldsymbol{u}^{\mathsf{T}}_{N,i}\boldsymbol{v}_N(i) \end{bmatrix}, \quad \boldsymbol{s}_i \triangleq \begin{bmatrix} s_{1,i}(\boldsymbol{\phi}_{1,i-1}) \\ s_{2,i}(\boldsymbol{\phi}_{2,i-1}) \\ \vdots \\ s_{N,i}(\boldsymbol{\phi}_{N,i-1}) \end{bmatrix} \tag{1.151c}$$

Moreover,

$$\mathcal{A}_1 \triangleq A_1 \otimes I_M, \quad \mathcal{A}_2 \triangleq A_2 \otimes I_M \tag{1.152a}$$

while the quantities

$$\mathcal{M} \triangleq \mathrm{diag}\{\,\mu_1 I_M,\ \mu_2 I_M,\ \dots,\ \mu_N I_M\,\} \tag{1.152b}$$

$$\mathcal{R}_i \triangleq \mathrm{diag}\left\{2\boldsymbol{u}^{\mathsf{T}}_{1,i}\boldsymbol{u}_{1,i}, 2\boldsymbol{u}^{\mathsf{T}}_{2,i}\boldsymbol{u}_{2,i}, \dots, 2\boldsymbol{u}^{\mathsf{T}}_{N,i}\boldsymbol{u}_{N,i}\right\} \tag{1.152c}$$

$$\mathcal{R} \triangleq \mathbb{E}\,\mathcal{R}_i = \mathrm{diag}\{2R_{u,1}, 2R_{u,2}, \dots, 2R_{u,N}\} \tag{1.152d}$$

$$\mathcal{S} \triangleq \mathbb{E}\,\boldsymbol{z}_i\boldsymbol{z}^{\mathsf{T}}_i = \mathrm{diag}\left\{4\sigma^2_{v,1}R_{u,1}, \dots, 4\sigma^2_{v,N}R_{u,N}\right\} \tag{1.152e}$$

are $N \times N$ *block* diagonal matrices, whose individual entries are of size $M \times M$ each. Let Σ be any $N \times N$ non-negative definite block matrix that we are free to choose, with blocks of size $M \times M$. Computing the Σ−weigthed squared norm of the error vector in (1.151b) under expectation gives (see the derivation leading to (269) in [7] or (38)–(39) in [79]):

$$\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_i\|^2_\Sigma = \mathbb{E}\,\|\widetilde{\boldsymbol{w}}_{i-1}\|^2_{\Sigma'} + \mathbb{E}\left(\boldsymbol{z}^{\mathsf{T}}_i\mathcal{M}\mathcal{A}_2\Sigma\mathcal{A}^{\mathsf{T}}_2\mathcal{M}\boldsymbol{z}_i\right) \tag{1.153a}$$

where the deterministic weighting matrix Σ' is given by:

$$\Sigma' = \mathcal{A}_1\left\{\mathcal{A}_2\Sigma\mathcal{A}^{\mathsf{T}}_2 - \mathcal{A}_2\Sigma\mathcal{A}^{\mathsf{T}}_2\mathcal{M}\mathcal{R} - \mathcal{R}\mathcal{M}\mathcal{A}_2\Sigma\mathcal{A}^{\mathsf{T}}_2 + \mathbb{E}\left(\mathcal{R}_i\mathcal{M}\mathcal{A}_2\Sigma\mathcal{A}^{\mathsf{T}}_2\mathcal{M}\mathcal{R}_i\right)\right\}\mathcal{A}^{\mathsf{T}}_1 \tag{1.153b}$$

We can evaluate the last expectations in (1.153a)–(1.153b) in closed-form. But first we need to introduce a convenient block-vector notation, denoted by bvec(·). Thus, given an $N \times N$ block matrix, with

60   **CHAPTER 1** Asynchronous Adaptive Networks

blocks of size $M \times M$ each, say for $N = 3$,

$$X = \begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix} \tag{1.154a}$$

its block vectorization is obtained as follows. We first vectorize each of the block entries and define the column vector $x_{k\ell} = \text{vec}(X_{k\ell})$; this operation stacks the columns of $X_{k\ell}$ on top of each other. Subsequently, the quantity $\text{bvec}(X)$ is obtained by stacking the vectors $\{x_{k\ell}\}$ on top of each other:

$$\text{bvec}(X) \triangleq \text{col}\{x_{11}, x_{21}, x_{31}, x_{12}, x_{22}, x_{32}, x_{13}, x_{23}, x_{33}\} \tag{1.154b}$$

The following two useful properties can be easily verified for any block matrices $\{A, B, \Sigma\}$ of compatible dimensions [152, 153]:

$$\text{bvec}(A\Sigma B) = (B^\mathsf{T} \otimes_b A)\text{bvec}(\Sigma) \tag{1.154c}$$

$$\text{Tr}(A^\mathsf{T} B) = (\text{bvec}(A))^\mathsf{T} \text{bvec}(B) \tag{1.154d}$$

where the notation $A \otimes_b B$ denotes the block Kronecker product of two block matrices A and B (assumed here to be both of size $N \times N$ with $M \times M$ blocks); the $k\ell$-th block of $A \otimes_b B$ has size $NM^2 \times NM^2$ and is given by [153]:

$$[A \otimes_b B]_{k\ell} = \begin{bmatrix} A_{k\ell} \otimes B_{11} & A_{k\ell} \otimes B_{12} & \dots & A_{k\ell} \otimes B_{1N} \\ A_{k\ell} \otimes B_{21} & A_{k\ell} \otimes B_{22} & \dots & A_{k\ell} \otimes B_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{k\ell} \otimes B_{N1} & A_{k\ell} \otimes B_{N2} & \dots & A_{k\ell} \otimes B_{NN} \end{bmatrix} \tag{1.154e}$$

in terms of the traditional Kronecker product operation. Using the block Kronecker properties (1.154c)–(1.154d) we now find that the last expectation in (1.153a) is given by:

$$\mathbb{E}\left(z_i^\mathsf{T} \mathcal{M}\mathcal{A}_2 \Sigma \mathcal{A}_2^\mathsf{T} \mathcal{M} z_i\right) = b^\mathsf{T} \sigma \tag{1.155a}$$

$$\sigma \triangleq \text{bvec}(\Sigma) \tag{1.155b}$$

$$b \triangleq (\mathcal{A}_2^\mathsf{T} \otimes_b \mathcal{A}_2^\mathsf{T})(\mathcal{M} \otimes_b \mathcal{M})\text{bvec}(\mathcal{S}) \tag{1.155c}$$

This is the same expression (56) from [78] for the case of CTA diffusion, and the same expression (42) from [79]) for CTA and ATC diffusion (except that this latter reference used the traditional vec(.) and Kronecker notation $\otimes$ instead of bvec(.) and the block Kronecker notation $\otimes_b$).

Let us now evaluate the last expectation in (1.153a). Let $Q = \mathcal{M}\mathcal{A}_2 \Sigma \mathcal{A}_2^\mathsf{T} \mathcal{M}$ so that we can rewrite more compactly:

$$\mathcal{K} \triangleq \mathbb{E}\left(\mathcal{R}_i \mathcal{M}\mathcal{A}_2 \Sigma \mathcal{A}_2^\mathsf{T} \mathcal{M}\mathcal{R}_i\right) = \mathbb{E}\,\mathcal{R}_i Q \mathcal{R}_i \tag{1.156a}$$

The $k\ell$-th block entry in this matrix is given by

$$\mathcal{K}_{k\ell} = 4\,\mathbb{E}\,u_{k,i}^\mathsf{T} u_{k,i} Q_{k\ell} u_{\ell,i}^\mathsf{T} u_{\ell,i} \tag{1.156b}$$

in terms of the $k\ell$-th block of Q. Using property (1.150d) for Gaussian regressors, we get

$$\mathcal{K}_{k\ell} = 4R_{u,k} Q_{k\ell} R_{u,\ell} + 4\delta_{k\ell}\{R_{u,k}\text{Tr}\left[Q_{kk} R_{u,k}\right] + R_{u,k} Q_{kk} R_{u,k}\} \tag{1.156c}$$

It is clear from the above expression that the matrix $\mathcal{K}$ has the following general form involving two block

diagonal matrices:

$$\frac{1}{4}\mathcal{K} \quad = \quad \mathcal{R}Q\mathcal{R} + Z_1 + Z_2 \tag{1.156d}$$

$$Z_1 \quad \triangleq \quad \text{blkdiag}\,\{R_{u,k}Q_{kk}R_{u,k}\} \tag{1.156e}$$

$$Z_2 \quad \triangleq \quad \text{blkdiag}\,\{R_{u,k}\text{Tr}\,(Q_{kk}R_{u,k})\} \tag{1.156f}$$

Introduce the block diagonal matrices (written for $N = 3$):

$$\mathcal{L}_1 \quad \triangleq \quad \begin{bmatrix} I_{M^2} & & & & & & & & \\ & 0 & & & & & & & \\ & & 0 & & & & & & \\ \hline & & & 0 & & & & & \\ & & & & I_{M^2} & & & & \\ & & & & & 0 & & & \\ \hline & & & & & & 0 & & \\ & & & & & & & 0 & \\ & & & & & & & & I_{M^2} \end{bmatrix} \tag{1.157a}$$

$$\mathcal{L}_2 \quad \triangleq \quad \begin{bmatrix} \lambda_1\lambda_1^{\mathsf{T}} & & & & & & & & \\ & 0 & & & & & & & \\ & & 0 & & & & & & \\ \hline & & & 0 & & & & & \\ & & & & \lambda_2\lambda_2^{\mathsf{T}} & & & & \\ & & & & & 0 & & & \\ \hline & & & & & & 0 & & \\ & & & & & & & 0 & \\ & & & & & & & & \lambda_3\lambda_3^{\mathsf{T}} \end{bmatrix} \tag{1.157b}$$

where $\lambda_k = \text{vec}(\Lambda_k)$. Then, it can be verified that

$$\text{bvec}(Z_1) \quad = \quad \mathcal{L}_1(\mathcal{R} \otimes_b \mathcal{R})\text{bvec}(Q) \tag{1.158a}$$

$$\text{bvec}(Z_2) \quad = \quad \mathcal{L}_2\text{bvec}(Q) \tag{1.158b}$$

Noting that

$$\text{bvec}(Q) = (\mathcal{M} \otimes_b \mathcal{M})(\mathcal{A}_2 \otimes_b \mathcal{A}_2)\sigma \tag{1.159}$$

we conclude that

$$\text{bvec}(\mathcal{K}) \quad = \quad 4\mathcal{X}(\mathcal{M} \otimes_b \mathcal{M})(\mathcal{A}_2 \otimes_b \mathcal{A}_2)\sigma \tag{1.160a}$$

$$\mathcal{X} \quad \triangleq \quad (I + \mathcal{L}_1)(\mathcal{R} \otimes_b \mathcal{R}) + \mathcal{L}_2 \tag{1.160b}$$

Note that the matrix $\mathcal{X}$ has the following block-diagonal structure:

$$\mathcal{X} \quad \triangleq \quad \text{diag}\,\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_N\} \tag{1.160c}$$

$$\mathcal{X}_k \quad \triangleq \quad \text{diag}\,\left\{\mathcal{X}_k^{(1)}, \mathcal{X}_k^{(2)}, \ldots, \mathcal{X}_k^{(N)}\right\} \tag{1.160d}$$

$$\mathcal{X}_k^{(\ell)} \quad = \quad \begin{cases} \Lambda_k \otimes \Lambda_\ell, & \text{when } k \neq \ell \\ \lambda_k\lambda_k^{\mathsf{T}} + 2\Lambda_k \otimes \Lambda_k, & \text{when } k = \ell \end{cases} \tag{1.160e}$$

which is the same structure derived through equations (57)–(67) in [78]. Substituting (1.160a) into (1.153b) and using again (1.154c)–(1.154d) we find that the block vectorized forms of the weighting matrices $\{\Sigma, \Sigma'\}$ are related via (the expression for $\mathcal{F}$ below fixes the typo in Eq. (44) from [79]; in the

62   CHAPTER 1 Asynchronous Adaptive Networks

derivation in this example we considered the special case in which $\mathcal{S}_m = I$ in (44) from [79]):

$$\sigma' \quad = \quad \mathcal{F}\sigma \tag{1.161a}$$

$$\mathcal{F} \quad \stackrel{\Delta}{=} \quad (\mathcal{A}_1 \otimes_b \mathcal{A}_1)\{I - (\mathcal{R}\mathcal{M} \otimes_b I) - (I \otimes_b \mathcal{R}\mathcal{M}) + 4(\mathcal{M} \otimes_b \mathcal{M})\mathcal{X}\}(\mathcal{A}_2 \otimes_b \mathcal{A}_2) \tag{1.161b}$$

This is the same expression for $\mathcal{F}$ in Eq. (69) from [78] for the case of CTA diffusion (where $A_2 = I_N$). This is also the same expression for $\mathcal{F}$ in Eq. (41) from [79] for CTA and ATC diffusion (except that this latter reference wrote $\otimes$ instead $\otimes_b$). Substituting (1.155a) and the above expression for σ' into (1.153a), and using the compact notation $\|x\|_\sigma^2$ for $\|x\|_\Sigma^2$, we rewrite (1.153a) in the form

$$\mathbb{E}\|\widetilde{w}_i\|_\sigma^2 \ = \ \mathbb{E}\|\widetilde{w}_{i-1}\|_{\mathcal{F}\sigma}^2 \ + \ b^\mathsf{T}\sigma \tag{1.162}$$

In seady-state, as $i \to \infty$, the mean-square-error approaches

$$\lim_{i \to \infty} \mathbb{E}\|\widetilde{w}_i\|_{(I-\mathcal{F})\sigma}^2 \ = \ b^\mathsf{T}\sigma \tag{1.163}$$

As explained in Sec. 6.6 of [7], the network MSD can be assessed by selecting σ to satisfy

$$(I - \mathcal{F})\sigma \ = \ \frac{1}{N}\mathrm{bvec}(I_{NM}) \tag{1.164}$$

which leads to the desired expression

$$\mathrm{MSD}_{\mathrm{diff,av}}^{\mathrm{sync}} \ = \ \frac{1}{N}b^\mathsf{T}(I - \mathcal{F})^{-1}\mathrm{bvec}(I_{NM}) \tag{1.165}$$

which is expression (105a) from [78].

To illustrate how these results are adjusted for asynchronous behavior, we consider the case in which the step-size parameters $\{\mu_k\}$ in (1.151a) are replaced by random values $\{\boldsymbol{\mu}_k(i)\}$. We denote the mean and variances of these random variables as follows:

$$\bar{\mu}_k \quad \stackrel{\Delta}{=} \quad \mathbb{E}\,\boldsymbol{\mu}_k(i) \tag{1.166a}$$

$$\boldsymbol{\mathcal{M}}_i \quad \stackrel{\Delta}{=} \quad \mathrm{diag}\{\boldsymbol{\mu}_1(i)I_M, \boldsymbol{\mu}_2(i)I_M, \ldots, \boldsymbol{\mu}_N(i)I_M\} \tag{1.166b}$$

$$\bar{\mathcal{M}} \quad \stackrel{\Delta}{=} \quad \mathbb{E}\,\boldsymbol{\mathcal{M}}_i \tag{1.166c}$$

$$C_\mu \quad \stackrel{\Delta}{=} \quad \mathbb{E}(\boldsymbol{\mathcal{M}}_i - \bar{\mathcal{M}}) \otimes_b (\boldsymbol{\mathcal{M}}_i - \bar{\mathcal{M}}) \tag{1.166d}$$

If we now repeat the same analysis, expressions (1.153a)–(1.153b) are replaced by

$$\mathbb{E}\|\widetilde{w}_i\|_\Sigma^2 \ = \ \mathbb{E}\|\widetilde{w}_{i-1}\|_{\Sigma'}^2 \ + \ \mathbb{E}\left(z_i^\mathsf{T}\boldsymbol{\mathcal{M}}_i\mathcal{A}_2\Sigma\mathcal{A}_2^\mathsf{T}\boldsymbol{\mathcal{M}}_iz_i\right) \tag{1.167a}$$

$$\Sigma' \ = \ \mathcal{A}_1\left\{\mathcal{A}_2\Sigma\mathcal{A}_2^\mathsf{T} - \mathcal{A}_2\Sigma\mathcal{A}_2^\mathsf{T}\bar{\mathcal{M}}\mathcal{R} - \mathcal{R}\bar{\mathcal{M}}\mathcal{A}_2\Sigma\mathcal{A}_2^\mathsf{T} + \mathbb{E}\left(\mathcal{R}_i\boldsymbol{\mathcal{M}}_i\mathcal{A}_2\Sigma\mathcal{A}_2^\mathsf{T}\boldsymbol{\mathcal{M}}_i\mathcal{R}_i\right)\right\}\mathcal{A}_1^\mathsf{T} \tag{1.167b}$$

We can evaluate the last expectations in (1.167a)–(1.167b) as follows. First we have:

$$\mathbb{E}\left(z_i^\mathsf{T}\boldsymbol{\mathcal{M}}_i\mathcal{A}_2\Sigma\mathcal{A}_2^\mathsf{T}\boldsymbol{\mathcal{M}}_iz_i\right) \ = \ b^\mathsf{T}\sigma \tag{1.168a}$$

where

$$b \quad \stackrel{\Delta}{=} \quad \mathbb{E}\,\mathrm{bvec}\left(\mathcal{A}_2^\mathsf{T}\boldsymbol{\mathcal{M}}_i\mathcal{S}\boldsymbol{\mathcal{M}}_i\mathcal{A}_2\right)$$

$$= \quad (\mathcal{A}_2^\mathsf{T} \otimes_b \mathcal{A}_2^\mathsf{T})\mathbb{E}(\boldsymbol{\mathcal{M}}_i \otimes_b \boldsymbol{\mathcal{M}}_i)\mathrm{bvec}(\mathcal{S})$$

$$= \quad (\mathcal{A}_2^\mathsf{T} \otimes_b \mathcal{A}_2^\mathsf{T})(C_\mu + \bar{\mathcal{M}} \otimes_b \bar{\mathcal{M}})\mathrm{bvec}(\mathcal{S}) \tag{1.168b}$$

Second, we have:

$$\mathrm{bvec}\left\{\mathbb{E}\left(\mathcal{R}_i\boldsymbol{\mathcal{M}}_i\mathcal{A}_2\Sigma\mathcal{A}_2^\mathsf{T}\boldsymbol{\mathcal{M}}_i\mathcal{R}_i\right)\right\} \ = \ \mathbb{E}(\boldsymbol{\mathcal{M}}_i \otimes_b \boldsymbol{\mathcal{M}}_i)\mathbb{E}(\mathcal{R}_i \otimes_b \mathcal{R}_i)(\mathcal{A}_2 \otimes_b \mathcal{A}_2)\sigma$$

$$= \ (C_\mu + \bar{\mathcal{M}} \otimes_b \bar{\mathcal{M}})\mathbb{E}(\mathcal{R}_i \otimes_b \mathcal{R}_i)(\mathcal{A}_2 \otimes_b \mathcal{A}_2)\sigma \tag{1.169}$$

so that we replace the quantities $\mathcal{M}$ and $(\mathcal{M} \otimes_b \mathcal{M})$ in the expressions (1.155c) and (1.161b) for b and $\mathcal{F}$ by $\bar{\mathcal{M}}$ and $(C_\mu + \bar{\mathcal{M}} \otimes_b \bar{\mathcal{M}})$, respectively.

∎

1.8 **CONCLUDING REMARKS**

This chapter provides an overview of asynchronous strategies for adaptation, learning, and optimization over networks including non-cooperative, centralized, consensus, and diffusion strategies. Particular attention is given to the constant step-size case in order to examine solutions that are able to adapt and learn continuously from streaming data. The presentation complements the results from [1, 5]. We introduced a fairly general model for asynchronous behavior that allows for random step-sizes, link failures, random topology variations, and random combination coefficients. We examined the mean-square-error performance and stability properties under asynchronous events and recovered results for synchronous operation as a special case. The results indicate that asynchronous networks are robust, resilient to failure, and remain mean-square stable for sufficiently small step-sizes.

There are of course several other aspects of distributed strategies that are not covered in this work. Comments on these aspects can be found in [1, 5, 7], including issues related to (a) the noisy exchange of information over links (e.g., [7, 14] and [108]–[112]); (b) the use of gossip strategies (e.g., [12, 15, 20, 22, 71, 113, 114]); (c) the exploitation of sparsity constraints (e.g., [115]–[118]); (d) the solution of constrained optimization problems (e.g., [13, 100, 119, 120, 121]); (e) the use of distributed solutions of the recursive least-squares type (e.g., [7, 72, 98]); (f) the development of distributed state-space solutions (e.g., [72, 97] and [122]–[128]); (g) the study of incremental-based strategies (e.g., [129]–[141]); (h) the study of distributed solutions under multi-task environments [142]–[146]; (i) the case of non-smooth risk functions in the context of subgradient learning [24, 25, 69, 116]; and (j) the incorporation of proximal operators into the distributed setting [147]–[151].

64   **CHAPTER 1** Asynchronous Adaptive Networks

REFERENCES

[1] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, April 2014.

[2] X. Zhao and A. H. Sayed, "Asynchronous adaptation and learning over networks — Part I: Modeling and stability analysis," *IEEE Trans. Signal Processing*, vol. 63, no. 4, pp. 811–826, Feb. 2015.

[3] X. Zhao and A. H. Sayed, "Asynchronous adaptation and learning over networks – Part II: Performance analysis," *IEEE Trans. Signal Processing*, vol. 63, no. 4, pp. 827–842, Feb. 2015.

[4] X. Zhao and A. H. Sayed, "Asynchronous adaptation and learning over networks – Part III: Comparison analysis," *IEEE Trans. Signal Processing*, vol. 63, no. 4, pp. 843–858, Feb. 2015.

[5] A. H. Sayed, *Adaptation, Learning, and Optimization over Networks*, Foundations and Trends in Machine Learning, vol. 7, issue 4–5, pp. 311–801, NOW Publishers, Boston-Delft, July 2014.

[6] A. H. Sayed, S-Y. Tu, J. Chen, X. Zhao, and Z. Towfic, "Diffusion strategies for adaptation and learning over networks," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 155–171, May 2013.

[7] A. H. Sayed, "Diffusion adaptation over networks," in *E-Reference Signal Processing*, vol. 3, R. Chellapa and S. Theodoridis, *Eds.*, pp. 323–454, Academic Press, 2014. Also available as arXiv:1205.4220v1 [cs.MA], May 2012.

[8] P. Di Lorenzo, S. Barbarossa, and A. H. Sayed, "Bio-inspired decentralized radio access based on swarming mechanisms over adaptive networks," *IEEE Trans. Signal Processing*, vol. 61, no. 12, pp. 3183–3197, June 2013.

[9] P. Chainais and C. Richard, "Learning a common dictionary over a sensor network," *Proc. IEEE CAMSAP*, pp. 1–5, Saint Martin, Dec. 2013.

[10] J. Chen, A. H. Sayed, and Z. Towfic, "Dictionary learning over distributed models," *IEEE Trans. Signal Process.* vol. 63, issue 4, pp. 1001–1016, February 2015.

[11] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Autom. Control*, vol. 31, no. 9, pp. 803–812, Sep. 1986.

[12] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. on Information Theory,* vol. 52, no. 6, pp. 2508–2530, Jun. 2006.

[13] K. Srivastava and A. Nedic, "Distributed asynchronous constrained stochastic optimization," *IEEE J. Sel. Topics. Signal Process.*, vol. 5, no. 4, pp. 772–790, Aug. 2011.

[14] S. Kar and J. M. F. Moura, "Distributed consensus algorithms in sensor networks: Link failures and channel noise," *IEEE Trans. Signal Process.*, vol. 57, no. 1, pp. 355–369, Jan. 2009.

[15] S. Kar and J. M. F. Moura, "Convergence rate analysis of distributed gossip (linear parameter) estimation: Fundamental limits and tradeoffs," *IEEE Journal on Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 674–690, Aug. 2011.

[16] S. Kar and J. M. F. Moura, "Sensor networks with random links: Topology design for distributed consensus," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3315–3326, July 2008.

[17] D. Jakovetic, J. Xavier, and J. M. F. Moura, "Weight optimization for consensus algorithms with correlated switching topology," *IEEE Trans. Signal Process.*, vol. 58, no. 7, pp. 3788–3801, July 2010.

[18] D. Jakovetic, J. Xavier, and J. M. F. Moura, "Cooperative convex optimization in networked systems: Augmented Lagranian algorithms with directed Gossip communication," *IEEE Trans. Signal Process.,* vol. 59, no. 8, pp. 3889–3902, Aug. 2011.

[19] S. Kar and J. M. F. Moura, "Distributed consensus algorithms in sensor netowrks: Quantized data and random link failures," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1383–1400, Mar. 2010.

65

66   CHAPTER 1 Asynchronous Adaptive Networks

[20] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," *IEEE Trans. on Signal Processing*, vol. 57, no. 7, pp. 2748–2761, July 2009.

[21] T. C. Aysal, A. D. Sarwate, and A. G. Dimakis, "Reaching consensus in wireless networks with probabilistic broadcast," in *Proc. Allerton Conf. Commun., Control, Comput.*, Allerton House, IL, Sept. and Oct. 2009, pp. 732–739.

[22] C. Lopes and A. H. Sayed, "Diffusion adaptive networks with changing topologies," *Proc. IEEE ICASSP*, pp. 3285–3288, Las Vegas, April 2008.

[23] N. Takahashi and I. Yamada, "Link probability control for probabilistic diffusion least-mean squares over resource-constrained networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.* (ICASSP), pp. 3518–3521, Dallas, TX, Mar. 2010.

[24] B. Ying and A. H. Sayed, "Performance limits of stochastic sub-gradient learning, Part I: Single agent case," *submitted for publication*, Also available as arXiv1511.07902, April 2017.

[25] B. Ying and A. H. Sayed, "Performance limits of stochastic sub-gradient learning, Part II: Multi-agent case," *submitted for publication*, Also available as arXiv1704.06025, April 2017.

[26] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, NJ, 2002.

[27] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice Hall, NJ, 1985.

[28] A. H. Sayed, *Adaptive Filters*, Wiley, NJ, 2008.

[29] A. H. Sayed, *Fundamentals of Adaptive Filtering*, Wiley, NJ, 2003.

[30] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation*, Prentice Hall, NJ, 2000.

[31] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2007.

[32] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 4th edition, Academic Press, 2008.

[33] D. W. Hosmer and S. Lemeshow, *Applied Logistic Regression*, 2nd edition, Wiley, NJ, 2000.

[34] B. T. Poljak and Y. Z. Tsypkin, "Pseudogradient adaptation and training algorithms," *Autom. Remote Control,* vol. 12, pp. 83–94, 1973.

[35] B. Poljak, *Introduction to Optimization*, Optimization Software, NY, 1987.

[36] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, 1st edition, Athena Scientific, Singapore, 1997.

[37] Y. Z. Tsypkin, *Adaptation and Learning in Automatic Systems*, Academic Press, NY, 1971.

[38] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[39] D. Bertsekas, *Convex Analysis and Optimization*, Athena Scientific, 2003.

[40] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Kluwer Academic Publishers, 2004.

[41] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Trans. Signal Processing*, vol. 60, no. 8, pp. 4289–4305, Aug. 2012.

[42] J. Chen and A. H. Sayed, "On the learning behavior of adaptive networks — Part I: Transient analysis," *IEEE Trans. Information Theory*, vol. 61, no. 6, pp. 3487–3517, June 2015.

[43] B. Widrow and M. E. Hoff, Jr., "Adaptive switching circuits," *IRE WESCON Conv. Rec.*, Pt. 4, pp. 96–104, 1960.

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

[45] W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. 2, Wiley, NY, 1971.

[46] G. J. Hahn and S. Shapiro, *Statistical Models in Engineering*, Wiley, NJ, 1994.

[47] M. Abramowitz and I. Stegun, Eds., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover, NY, 1972.

[48] G. E. Andrews, R. Askey and R. Roy, *Special Functions*, Cambridge University Press, Cambridge, 1999.

[49] W. Rudin, *Principles of Mathematical Analysis*, McGraw-Hill, 1976.

[50] N. R. Yousef and A. H. Sayed, "A unified approach to the steady-state and tracking analysis of adaptive filters," *IEEE Trans. Signal Processing*, vol. 49, no. 2, pp. 314–324, February 2001.

[51] T. Y. Al-Naffouri and A. H. Sayed, "Transient analysis of data-normalized adaptive filters," *IEEE Trans. Signal Processing*, vol. 51, no. 3, pp. 639–652, Mar. 2003.

[52] J. Chen and A. H. Sayed, "On the learning behavior of adaptive networks — Part II: Performance analysis," *IEEE Trans. Information Theory*, vol. 61, no. 6, pp. 3518–3548, June 2015.

[53] A. Papoulis and S. U. Pilla, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, NY, 2002.

[54] R. Durret, *Probability Theory and Examples*, 2nd edition, Duxbury Press, 1996.

[55] R. M. Dudley, *Real Analysis and Probability*, 2nd edition, Cambridge University Press, 2003.

[56] B. Widrow, J. M. McCool, M. G. Larimore, and C. R. Johnson Jr., "Stationary and nonstationary learning characterisitcs of the LMS adaptive filter," *Proc. IEEE*, vol. 64, no. 8, pp. 1151–1162, Aug. 1976.

[57] L. Horowitz and K. Senne, "Performance advantage of complex LMS for controlling narrowband adaptive arrays," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 29, no. 3, pp. 722–736, Jun. 1981.

[58] S. Jones, R. C. III, and W. Reed, "Analysis of error-gradient adaptive linear estimators for a class of stationary dependent processes," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 318–329, Mar. 1982.

[59] W. A. Gardner, "Learning characterisitcs of stochastic-gradient-descent algorithms: A general study, analysis, and critique," *Signal Process.*, vol. 6, no. 2, pp. 113–133, Apr. 1984.

[60] A. Feuer and E. Weinstein, "Convergence analysis of LMS filters with uncorrelated Gaussian data," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 33, no. 1, pp. 222–230, Feb. 1985.

[61] J. B. Foley and F. M. Boland, "A note on the convergence analysis of LMS adaptive filters with Gaussian data," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 36, no. 7, pp. 1087–1089, Jul. 1988.

[62] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, NY, 2000.

[63] Z. Towfic, J. Chen, and A. H. Sayed, "On the generalization ability of distributed online learners," *Proc. IEEE Workshop on Machine Learning for Signal Processing* (MLSP), Santander, Spain, pp. 1–6, Sep. 2012.

[64] X. Zhao and A. H. Sayed, "Performance limits for distributed estimation over LMS adaptive networks," *IEEE Trans. Signal Processing*, vol. 60, no. 10, pp. 5107–5124, Oct. 2012.

[65] J. Chen and A. H. Sayed, "Distributed Pareto optimization via diffusion strategies," *IEEE J. Selected Topics in Signal Processing*, vol. 7, no. 2, pp. 205–220, April 2013.

[66] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. Control Lett.*, vol. 53, no. 1, pp. 65–78, Sep. 2004.

[67] J. Tsitsiklis and M. Athans, "Convergence and asymptotic agreement in distributed decision problems," *IEEE Trans. Autom. Control*, vol. 29, no. 1, pp. 42–50, Jan. 1984.

[68] A. Nedic and A. Ozdaglar, "Cooperative distributed multi-agent optimization," in *Convex Optimization in Signal Processing and Communications*, Y. Eldar and D. Palomar (Eds.), Cambridge University Press, pp. 340-386, 2010.

[69] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[70] B. Johansson, T. Keviczky, M. Johansson, and K. Johansson, "Subgradient methods and consensus algorithms for solving convex optimization problems," *Proc. IEEE CDC*, pp. 4185–4190, Cancun, Mexico, December 2008.

[71] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proc. IEEE*, vol. 98, no. 11, pp. 1847–1864, Nov. 2010.

[72] L. Xiao, S. Boyd and S. Lall, "A space-time diffusion scheme peer-to-peer least-squares-estimation," *Proc. Information Processing in Sensor Networks (IPSN)*, pp. 168–176, Nashville, TN, April 2006.

[73] W. Ren and R. W. Beard, "Consensus seeking in multi-agent systems under dynamically changing interaction topologies," *IEEE Trans. on Automatic Control*, vol. 50, pp. 655–661, May 2005.

[74] R. Olfati-Saber and J. Shamma, "Consensus filters for sensor networks and distributed sensor fusion," *Proc. 44th IEEE Conference on Decision and Control (CDC)*, pp. 6698–6703, Seville, Spain, Dec. 2005.

[75] S. Barbarossa, and G. Scutari, "Bio-inspired sensor network design," *IEEE Signal Processing Magazine*, vol. 24, no. 3, pp. 26–35, May 2007.

[76] S. Sardellitti, M. Giona, and S. Barbarossa, "Fast distributed average consensus algorithms based on advection-diffusion processes," *IEEE Trans. Signal Processing*, vol. 58, no. 2, pp. 826–842, Feb. 2010.

[77] P. Braca, S. Marano, and V. Matta, "Running consensus in wireless sensor networks," *Proc. 11th International Conference on Information Fusion*, pp. 1–6, Cologne, Germany, June 2008.

[78] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3122–3136, July 2008.

[79] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1035–1048, Mar. 2010.

[80] S-Y. Tu and A. H. Sayed, "Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks," *IEEE Trans. on Signal Processing*, vol. 60, no. 12, pp. 6217–6234, Dec. 2012.

[81] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 2003.

[82] S.-Y. Tu and A. H. Sayed, "Mobile adaptive networks," *IEEE J. Sel. Topics. Signal Process.*, vol. 5, no. 4, pp. 649–664, Aug. 2011.

[83] F. Cattivelli and A. H. Sayed, "Modeling bird flight formations using diffusion adaptation," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2038–2051, May 2011.

[84] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction," *Proc. International Conference on Machine Learning (ICML)*, Bellevue, WA, pp. 713–720, Jun. 2011.

[85] A. Agarwal and J. Duchi, "Distributed delayed stochastic optimization," *Proc. Neural Information Processing Systems (NIPS)*, Granada, Spain, pp. 873–881, Dec. 2011.

[86] J. B. Predd, S. B. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 56–69, Jul. 2006.

[87] Z. J. Towfic, J. Chen, and A. H. Sayed, "Collaborative learning of mixture models using diffusion adaptation," *Proc. IEEE Workshop Mach. Learn. Signal Process. (MLSP)*, Beijing, China, pp. 1–6, Sep. 2011.

[88] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd edition, The John Hopkins University Press, Baltimore, 1996.

[89] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, SIAM, PA, 1994.

[90] S. U. Pillai, T. Suel, and S. Cha, "The Perron–Frobenius theorem: Some of its applications," *IEEE Signal Process. Mag.*, vol. 22, no. 2, pp. 62–75, Mar. 2005.

[91] C. G. Lopes and A. H. Sayed, "Distributed processing over adaptive networks," in *Proc. Adaptive Sensor Array Processing Workshop*, MIT Lincoln Laboratory, MA, pp.1–5, June 2006.

[92] A. H. Sayed and C. G. Lopes, "Adaptive processing over distributed networks," *IEICE Trans. Fund. of Electron., Commun. and Comput. Sci.*, vol. E90-A, no. 8, pp. 1504–1510, 2007.

[93] C. G. Lopes and A. H. Sayed, "Diffusion least-mean-squares over adaptive networks," *Proc. IEEE ICASSP*, Honolulu, Hawaii, vol. 3, pp. 917-920, April 2007.

[94] C. G. Lopes and A. H. Sayed, "Steady-state performance of adaptive diffusion least-mean squares," *Proc. IEEE Workshop on Statistical Signal Processing* (SSP), pp. 136-140, Madison, WI, Aug. 2007.

[95] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS algorithms with information exchange," *Proc. Asilomar Conf. Signals, Syst. Comput.*, Pacific Grove, CA, pp. 251–255, Nov. 2008.

[96] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "A diffusion RLS scheme for distributed estimation over adaptive networks," *Proc. IEEE Workshop on Signal Process. Advances Wireless Comm. (SPAWC)*, Helsinki, Finland, pp. 1–5, June 2007.

[97] F. S. Cattivelli and A. H. Sayed, "Diffusion mechanisms for fixed-point distributed Kalman smoothing," *Proc. EUSIPCO*, Lausanne, Switzerland, pp. 1–4, Aug. 2008.

[98] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Trans. Signal Process.*, vol. 56, no. 5, pp. 1865–1877, May 2008.

[99] S. S. Ram, A. Nedic, and V. V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *J. Optim. Theory Appl.*, vol. 147, no. 3, pp. 516–545, 2010.

[100] S. Lee and A. Nedic, "Distributed random projection algorithm for convex optimization," *IEEE J. Selected Topics in Signal Processing*, vol. 7, no. 2, pp. 221–229, Apr. 2013.

[101] P. Bianchi, G. Fort, and W. Hachem, "Performance of a distributed stochastic approximation algorithm," *IEEE Trans. Information Theory*, vol. 59, no. 11, pp. 7405–7418, Nov. 2013.

[102] S. S. Stankovic, M. S. Stankovic, and D. S. Stipanovic, "Decentralized parameter estimation by consensus based stochastic approximation," *IEEE Trans. on Autom. Control*, vol. 56, no. 3, pp. 531–543, Mar. 2011.

[103] J. Chen and A. H. Sayed, "On the limiting behavior of distributed optimization strategies," *Proc. 50th Annual Allerton Conference on Communication, Control, and Computing*, pp. 1535–1542, Monticello, IL, Oct. 2012.

[104] J. A. Bondy and U. S. R. Murty, *Graph Theory*, Springer, 2008.

[105] X. Zhao, S-Y. Tu, and A. H. Sayed, "Diffusion adaptation over networks under imperfect information exchange and non-stationary data," *IEEE Trans. Signal Processing*, vol. 60, no. 7, pp. 3460–3475, July 2012.

[106] N. Takahashi, I. Yamada, and A. H. Sayed, "Diffusion least-mean-squares with adaptive combiners: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 58, no. 9, pp. 4795–4810, Sep. 2010.

[107] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," vol. 11, pp. 985–1042, Sept. 2010.

[108] S-Y. Tu and A. H. Sayed, "Adaptive networks with noisy links," *Proc. IEEE Globecom*, pp. 1–5, Houston, TX, December 2011.

[109] R. Abdolee and B. Champagne, "Diffusion LMS algorithms for sensor networks over non-ideal inter-sensor wireless channels," *Proc. IEEE Int. Conf. Dist. Comput. Sensor Systems* (DCOSS), pp. 1–6, Barcelona, Spain, June 2011.

[110] A. Khalili, M. A. Tinati, A. Rastegarnia, and J. A. Chambers, "Steady state analysis of diffusion LMS adaptive networks with noisy links," *IEEE Trans. Signal Processing*, vol. 60, no. 2,

pp. 974–979, Feb. 2012.

[111] X. Zhao and A. H. Sayed, "Combination weights for diffusion strategies with imperfect information exchange," *Proc. IEEE ICC*, pp. 648–652, Ottawa, Canada, June 2012.

[112] G. Mateos, I. D. Schizas, and G. B. Giannakis, "Performance analysis of the consensus-based distributed LMS algorithm," *EURASIP J. Adv. Signal Process.*, pp. 1–19, 2009, 10.1155/2009/981030, Article ID 981030.

[113] D. Shah, "Gossip algorithms," *Found. Trends Netw.*, vol. 3, pp. 1–125, 2009.

[114] O. L. Rortveit, J. H. Husoy, and A. H. Sayed, "Diffusion LMS with communications constraints," *Proc. 44th Asilomar Conference on Signals, Systems and Computers,* Pacific Grove, CA, pp. 1645–1649, Nov. 2010.

[115] P. Di Lorenzo and A. H. Sayed, "Sparse distributed learning based on diffusion adaptation," *IEEE Trans. Signal Processing*, vol. 61, no. 6, pp. 1419–1433, March 2013.

[116] S. Chouvardas, K. Slavakis, Y. Kopsinis, S. Theodoridis, "A sparsity-promoting adaptive algorithm for distributed learning," *IEEE Transactions on Signal Processing*, vol. 60, no. 10, pp. 5412–5425, Oct. 2012.

[117] S. Chouvardas, G. Mileounis, N. Kalouptsidis, and S. Theodoridis, "A greedy sparsity-promoting LMS for distributed adaptive learning in diffusion networks," *Proc. ICASSP*, pp. 5415–5419, Vancouver, BC, Canada, 2013.

[118] Y. Liu, C. Li and Z. Zhang, "Diffusion sparse least-mean squares over networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4480–4485, Aug. 2012.

[119] F. Yan, S. Sundaram, S. V. N. Vishwanathan, and Y. Qi, "Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties," *IEEE Trans. Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2483–2493, Nov. 2013.

[120] S. Theodoridis, K. Slavakis, and I. Yamada, "Adaptive learning in a world of projections: A unifying framework for linear and nonlinear classification and regression tasks," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 97–123, Jan. 2011.

[121] Z. Towfic and A. H. Sayed, "Adaptive penalty-based distributed stochastic convex optimization," *IEEE Trans. Signal Process.*, vol. 62, no. 15, pp. 3924–3938, August 2014.

[122] R. Olfati-Saber, "Kalman-consensus filter: Optimality, stability, and performance," *Proc. IEEE CDC*, pp. 7036–7042, Shangai, China, 2009.

[123] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," *Proc. 46th IEEE Conf. Decision Control*, pp. 5492–5498, New Orleans, LA, Dec. 2007.

[124] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion strategies for distributed Kalman filtering: Formulation and performance analysis," *Proc. IAPR Workshop on Cognitive Inf. Process.*(CIP), Santorini, Greece, pp. 36–41, June 2008.

[125] F. Cattivelli and A. H. Sayed, "Diffusion strategies for distributed Kalman filtering and smoothing," *IEEE Trans. Automatic Control*, vol. 55, no. 9, pp. 2069–2084, Sep. 2010.

[126] F. Cattivelli and A. H. Sayed, "Diffusion distributed Kalman filtering with adaptive weights," *Proc. Asilomar Conference on Signals, Systems and Computers*, pp. 908–912, Pacific Grove, CA, Nov. 2009.

[127] U. A. Khan and J. M. F. Moura, "Distributing the Kalman filter for large-scale systems," *IEEE Trans. Signal Processing*, vol. 56, no. 10, pp. 4919–4935, Oct. 2008.

[128] P. Alriksson and A. Rantzer, "Distributed Kalman filtering using weighted averaging," *Proc. 17th Int. Symp. Math. Thy Net. Sys* (MTNS), pp. 1–6, Kyoto, Japan, 2006.

[129] D. P. Bertsekas, "A new class of incremental gradient methods for least squares problems," *SIAM J. Optim.*, vol. 7, no. 4, pp. 913–926, 1997.

[130] D. P. Bertsekas, *Nonlinear Programming*, 2nd edition, Athena Scientific, Belmont, MA, 1999.

[131] A. Nedic and D. P. Bertsekas, "Incremental subgradient methods for nondifferentiable optimization," *SIAM J. Optim.,* vol. 12, no. 1, pp. 109–138, 2001.

[132] M. G. Rabbat and R. D. Nowak, "Quantized incremental algorithms for distributed optimization," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 798–808, 2005.

[133] E. S. Helou and A. R. De Pierro, "Incremental subgradients for constrained convex optimization: A unified framework and new methods," *SIAM J. on Optimization*, vol. 20, pp. 1547–1572, 2009.

[134] B. Johansson, M. Rabi, and M. Johansson, "A randomized incremental subgradient method for distributed optimization in networked systems," *SIAM J. on Optimization*, vol. 20, pp. 1157–1170, 2009.

[135] D. Blatt, A. O. Hero, and H. Gauchman, "A convergent incremental gradient method with a constant step size," *SIAM J. Optimization*, vol. 18, pp. 29–51, 2008.

[136] A. H. Sayed and C. Lopes, "Distributed recursive least-squares strategies over adaptive networks," *Proc. 40th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, pp. 233–237, Oct.-Nov., 2006.

[137] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Trans. Signal Process.,* vol. 55, no. 8, pp. 4064–4077, Aug. 2007.

[138] A. H. Sayed and F. Cattivelli, "Distributed adaptive learning mechanisms," *Handbook on Array Processing and Sensor Networks,* S. Haykin and K. J. Ray Liu, *Eds.*, pp. 695–722, Wiley, NJ, 2009.

[139] L. Li, J. Chambers, C. G. Lopes, and A. H. Sayed, "Distributed estimation over an adaptive incremental network based on the affine projection algorithm," *IEEE Trans. Signal Processing*, vol. 58, no. 1, pp. 151–164, Jan. 2010.

[140] F. Cattivelli and A. H. Sayed, "Analysis of spatial and incremental LMS processing for distributed estimation," *IEEE Trans. Signal Processing*, vol. 59, no. 4, pp. 1465-1480, April 2011.

[141] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "A collaborative training algorithm for distributed learning," *IEEE Trans. Information Theory*, vol. 55, no. 4, pp. 1856–1871, April 2009.

[142] R. Nassif, C. Richard, A. Ferrari, and A. H. Sayed, "Multitask diffusion adaptation over asynchronous networks," *IEEE Trans. Signal Processing*, vol. 64, no. 11, pp. 2835-2850, June 2016.

[143] A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal estimation in fully connected sensor networks – Part I: Sequential node updating," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5277–5291, Oct. 2010.

[144] N. Bogdanovic, J. Plata-Chaves, and K. Berberidis, "Distributed diffusion-based LMS for node-specific parameter estimation over adaptive networks," *Proc. IEEE ICASSP*, pp. 7223–7227, Florence, Italy, May 2014.

[145] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Trans. Signal Processing*, vol. 62, no. 16, pp. 4129–4144, August 2014.

[146] J. Chen, C. Richard, and A. H. Sayed, "Diffusion LMS over multitask networks," *IEEE Trans. Signal Process.*, vol. 63, no. 11, pp. 2733–2748, June 2015.

[147] A. I. Chen and A. Ozdaglar, "A fast distributed proximal gradient method," *Proc. Annual Allerton Conference on Communication, Control, and Computing*, pp. 601–608, Allerton, USA, Oct. 2012.

[148] W. Wee and I. Yamada, "A proximal splitting approach to regularized distributed adaptive estimation in diffusion network," *in Proc. IEEE ICASSP*, pp. 5420–5424, Vancouver, Canada, May 2013,

[149] S. Vlaski, L. Vandenberghe, and A. H. Sayed, "Diffusion stochastic optimization with non-

smooth regularizers," *Proc. IEEE ICASSP*, pp. 4149–4153, Shanghai, China, March 2016.

[150] S. Vlaski and A. H. Sayed, "Proximal diffusion for stochastic costs with non-differentiable regularizers," *Proc. IEEE ICASSP*, pp. 3352–3356, Brisbane, Australia, April 2015.

[151] R. Nassif, A. Ferrari, C. Richard, and A. H. Sayed, "Proximal multitask learning over networks with sparsity-inducing coregularization," *IEEE Trans. Signal Processing*, vol. 64, no. 23, pp. 6329–6344, Dec. 2016.

[152] D. S. Tracy and R. P. Singh, "A new matrix product and its applications in partitioned matrix differentiation," *Statistica Neerlandica*, vol. 26, no. 4, pp. 143–157, 1972.

[153] R. H. Koning and H. Neudecker and T. Wansbeek, "Block Kronecker products and the vecb operator, *Linear Algebra Appl.*, vol. 149, pp. 165–184, Apr. 1991.