8 Algorithms for Decentralized Optimization

Most of the treatment in the earlier chapters focused on *single-agent* optimization and learning methods, where a single learner (or agent) is trained to discriminate between classes. For example, in logistic regression and support vector machine methods, the learners are trained by stochastic gradient or subgradient methods, whose objective is to approach the minimizers of certain empirical or stochastic risks. For various reasons explained below, it is useful in several applications to consider scenarios where the optimization and learning efforts need to split among *multiple* dispersed agents or learners. In these cases, each individual agent or learner will have access to part of the training data, and agents will be linked by some graph topology that allows neighboring agents to share information with each other. We will derive various algorithms that will allow the connected agents, through continuous interactions with their neighbors, to attain or approach the performance of a single powerful centralized processor as if each agent had access to all data that is dispersed across the graph.

We will refer to these methods as *decentralized* algorithms since all processing will be performed locally and there will be no central processor involved in fusing data from the agents. Thus, in the context of decentralized algorithms, *both* the data *and* the processing are localized with the data being collected locally by individual agents and the processing being performed locally across neighborhoods. We will make a distinction between *decentralized* and *distributed* algorithms. In the latter case, the data is collected locally at the agents, some local processing may also be performed at the agents, but there is a *fusion center* that receives processed data from the agents, fuses the data, and shares the result back with the dispersed agents. The architecture of *federated learning* studied earlier in Chapter 5 is one example of a distributed algorithm. Sometimes the terminologies "*decentralized*" and "*distributed*" algorithms are used interchangeably in the literature;, with authors in some fields using the qualification "distributed" to refer to "decentralized" methods as well. We will maintain a distinction between both terms in the manner described above.

There are many reasons for the interest in decentralized implementations. First, a centralized solution with a fusion center is costly to maintain and has a serious point of failure. If the central processor fails, the entire solution method becomes mute. Second, decentralized solutions are robust to link and agent failures. If one agent fails, or if a link in the graph topology is dropped, it is usually the case that the remaining agents remain connected and the learning process can continue. Third, decentralized solutions provide the individual agents with more privacy and secrecy; agents keep their local and raw data and do not share it globally. In many situations involving sensitive data, agents may not be comfortable sharing their raw personal data with remote fusion centers in centralized implementations or even with their neighboring agents. Decentralized solutions provide one way out whereby agents will only need to share some processed statistic with their neighbors. Fourth, the data may already be available in dispersed locations, as happens with cloud computing, and it can be costly to transfer the data to a central location for processing. Decentralized learning procedures offer an attractive approach to dealing with such large data sets.

In summary, decentralized solutions are more robust than centralized solution, more resilient to failure, offer more privacy, and avoid the need to coalesce large data sets in single locations. At the same time, decentralized mechanisms can serve as important enablers for the design of robotic swarms, consisting of multiple agents moving in coordination, which can assist, for example, in the exploration of disaster areas.

8.1

As in previous chapters, we will continue to study consensus optimization problems of the form:

$$\min_{w} \sum_{k=1}^{K} J_k(w) \tag{8.1}$$

Observe that all local cost functions $J_k(\cdot)$ are evaluated at a common w and hence, problem (8.1) becomes a *global* optimization problem. A useful strategy when developing decentralized algorithms is the removal of global variable by replacement through local variables. To this end, we examine more closely the following alternative formulation:

$$\min_{w_k} \sum_{k=1}^{K} J_k(w_k) \text{ subject to } w_k = w_\ell \ \forall\ k, \ell. \tag{8.2}$$

The constraint enforcing pairwise equality ensures that the optimization problems (8.1) and (8.2) are in fact equivalent. This construction decouples the aggregate cost $\sum_{k=1}^{K} J_k(w_k)$ and tansfers the coupling into the constraint set $w_k = w_\ell$ for all pairs k and ℓ. Since equality is enforced for *every* pair of nodes in the network, this constraint set continues to be *global.* We can further reformulate (8.2) into the equivalent problem:

$$\min_{w_k} \sum_{k=1}^{K} J_k(w_k) \text{ subject to } w_k = w_\ell \ \forall\ \ell \in \mathcal{N}_k. \tag{8.3}$$

where $\mathcal{N}_k$ denotes the neighborhood of node k. It can be verified that (8.3) is equivalent to (8.2) as long as the graph is connected. We will now proceed to describe two approaches to deriving decentralized algorithms for the pursuit of solutions of (8.3), and hence, by equivalence (8.1). For brevity, we introduce the *aggregate* cost:

$$J(w) \triangleq \sum_{k=1}^{K} J_k(w) \tag{8.4}$$

8.2 PENALTY-BASED ALGORITHMS

The first class of algorithms, which we will refer to as penalty-based algorithms, are derived by transforming the constraint $w_k = w_\ell \; \forall \; \ell \in \mathcal{N}_k$ into a penalty. To this end, we examine the alternative problem formulation:

$$\min_{w_k} \sum_{k=1}^{K} J_k(w_k) + \frac{\eta}{2} \sum_{k=1}^{K} \sum_{\ell \in \mathcal{N}_k} c_{\ell k} \|w_k - w_\ell\|^2 \tag{8.5}$$

where $c_{\ell k} > 0$ denotes the weight used to penalize the squared deviation of w_k from w_ℓ. We also allow for a common scaling factor $\eta > 0$. Comparing (8.5) to (8.3), we observe that disagreement between neighboring agents, i.e. deviations from the constraint set in (8.5), are penalized with weight $\eta c_{\ell k}$. The aggregate regularizer $\frac{\eta}{2} \sum_{k=1}^{K} \sum_{\ell \in \mathcal{N}_k} c_{\ell k} \|w_k - w_\ell\|^2$ is a useful measure of the variability of the weights w_k across the graph by means of local deviations $\|w_k - w_\ell\|^2$. We illustrate this in the following two examples.

Example 8.1 (**A numerical example.**) Let us consider a graph consisting of three nodes, arranged in an undirected line graph with symmetric weight matrix:

$$C = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \Longleftrightarrow L \overset{(8.14)}{=} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \tag{8.6}$$

To simplify calculations, consider scalar weight vectors $w_k \in \mathbb{R}$ and two instances of $\mathcal{w}$:

$$\{w_1 = 1; w_2 = 2; w_3 = 3\} \Longrightarrow \mathcal{w} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \text{ or } \{w_1' = 2; w_2' = 1; w_3' = 3\} \Longrightarrow \mathcal{w}' = \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix} \tag{8.7}$$

We illustrate the line graph and associated weight vectors in Fig. 8.1. Note that both $\mathcal{w}$ and $\mathcal{w}'$ have the same energy, i.e., $\|\,\mathcal{w}\,\|^2 = \|\,\mathcal{w}'\,\|^2 = 14$. Nevertheless, if we compute the variation measure, we find:

$$\mathcal{w}^{\mathsf{T}} \mathcal{L}\, \mathcal{w} = 2 \text{ and } \mathcal{w}'^{\mathsf{T}} \mathcal{L}\, \mathcal{w}' = 5 \tag{8.8}$$

and hence $\mathcal{w}^{\mathsf{T}} \mathcal{L}\, \mathcal{w} < \mathcal{w}'^{\mathsf{T}} \mathcal{L}\, \mathcal{w}'$. This result quantifies the fact that the weight vectors $\{w_1 = 1; w_2 = 2; w_3 = 3\}$ vary more smoothly than $\{w_1' = 2; w_2' = 1; w_3' = 3\}$ over the line graph described by the weight matrix C or Laplacian matrix L, where node 2 is

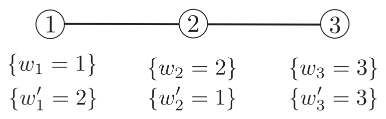
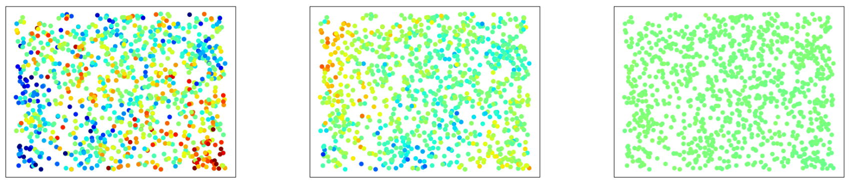


Figure 8.1 Schematic of the line graph from Example 8.1 with associated weights w and w'.



$\eta = 0.001$, large $\mathrm{w}^{\mathsf{T}}\,\mathcal{L}\,\mathrm{w}$ $\eta = 0.005$, medium $\mathrm{w}^{\mathsf{T}}\,\mathcal{L}\,\mathrm{w}$ $\eta = 1$, small $\mathrm{w}^{\mathsf{T}}\,\mathcal{L}\,\mathrm{w}$

Figure 8.2 Gauss Markov random field samples.

central.

Example 8.2 (**A visual example.**) To illustrate the graphical variation measure on a larger graph, we place $K = 1000$ nodes uniformly at random in a two-dimensional plane, and construct a graph C and Laplacian L geometrically by connecting nodes whose Euclidean distance is smaller than a threhshold. We then generate w by sampling from a Gaussian Markov random field with distribution

$$f(\mathrm{w}) = (2\pi\eta)^{-M(K-1)/2}\left(|\mathcal{L}|^*\right)^{1/2} e^{-\frac{\eta}{2}\mathrm{w}^{\mathsf{T}}\,\mathcal{L}\,\mathrm{w}} \tag{8.9}$$

Here η represents a temperature parameter and smaller η is more likely to generate weight vectors w_k with large graphical variability. We show the resulting distributions in Fig. 8.2 for varying η.

It is important to recognize that, in contrast to the sequence of equivalent reformulations (8.1)–(8.3), the penalized problem (8.5) is no longer an equivalent reformulation, and is instead an *approximation* for the problems (8.1)–(8.3). In particular, while we can expect sufficiently large η to ensure $w_k \approx w_\ell$, since the penalty paid for the deviation $\|w_k - w_\ell\|^2$ is high, *exact* equality will not be guaranteed for any finite η. Nevertheless, we will be able to precisely quantify the bias induced by replacing (8.1) by the penalized approximation (8.5). More importantly, we will see that (8.5) yields itself to a class of simple, decentralized algorithms with strong performance guarantees, even in the pursuit of (8.1).

8.2.1 Network Quantities

The collection of variables w_k for $k = 1, \ldots, K$ describes the state of the network across agents. For compactness, and in particular through most of the analysis we will be conducting, it is useful to define *network quantities*. To this end, let:

$$\mathcal{W} \triangleq \text{col}\{w_1, w_2, \dots, w_K\} \in \mathbb{R}^{MK} \tag{8.10}$$

Note that $\mathcal{W}$ is obtained by stacking w_k vertically, resulting in a vector of length MK. Throughout this text, caligraphic variables will be used to denote *network-level* quantities. We will also let:

$$\mathcal{J}(\mathcal{W}) \triangleq \sum_{k=1}^{K} J_k(w_k) \tag{8.11}$$

We can then write (8.5) more compactly as:

$$\min_{\mathcal{W}} \mathcal{J}(\mathcal{W}) + \frac{\eta}{2} \sum_{k=1}^{K} \sum_{\ell \in \mathcal{N}_k} c_{\ell k} \|w_k - w_\ell\|^2 \tag{8.12}$$

We have seen before (see Prob. 7.1) that the term $\frac{\eta}{2}\sum_{k=1}^{K}\sum_{\ell\in\mathcal{N}_k} c_{\ell k}\|w_k - w_\ell\|^2$ consisting of the sum of pairwise penalties can also be rewritten more compactly in terms of the network quantity $\mathcal{W}$. Specifically, we have:

$$\frac{\eta}{2} \sum_{k=1}^{K} \sum_{\ell \in \mathcal{N}_k} c_{\ell k} \|w_k - w_\ell\|^2 = \frac{\eta}{2} \mathcal{W}^{\mathsf{T}} \mathcal{L} \, \mathcal{W} \tag{8.13}$$

where $\mathcal{L} = L \otimes I_M$ and we defined:

$$L = \text{diag}\{C\mathbb{1}\} - C \tag{8.14}$$

Under those definitions we find:

$$\min_{\mathcal{W}} \mathcal{J}(\mathcal{W}) + \frac{\eta}{2} \mathcal{W}^{\mathsf{T}} \mathcal{L} \, \mathcal{W} \tag{8.15}$$

For brevity, we shall define:

$$\mathcal{J}^{\eta}(\mathcal{W}) \triangleq \mathcal{J}(\mathcal{W}) + \frac{\eta}{2} \mathcal{W}^{\mathsf{T}} \mathcal{L} \, \mathcal{W} \tag{8.16}$$

The objective $\mathcal{J}^{\eta}(\mathcal{W})$ takes the form of an unconstrained optimization problem in the extended network variable $\mathcal{W}$, while the regularization term $\frac{\eta}{2}\mathcal{W}^{\mathsf{T}}\mathcal{L}\,\mathcal{W}$ encourages that solutions where individual blocks w_k and w_ℓ are close in the sense of the squared Euclidean distance. We can hence approach this problem using the first-order optimization tools we developed in Chapter 2.

8.2.2 The Consensus+Innovations Algorithm

Having approximated (8.1) through (8.15), we are now ready to present the first decentralized algorithm for the (approximate) solution of (8.1). Applying gradient descent to (8.15), we obtain the recursion:

$$\begin{aligned} \mathcal{W}_i &= \mathcal{W}_{i-1} - \mu \left(\nabla_{\mathcal{W}} \mathcal{J}(\mathcal{W}_{i-1}) + \eta \mathcal{L} \, \mathcal{W}_{i-1} \right) \\ &= (I - \mu\eta\mathcal{L}) \, \mathcal{W}_{i-1} - \mu \nabla_{\mathcal{W}} \mathcal{J}(\mathcal{W}_{i-1}) \end{aligned} \tag{8.17}$$

For simplicity, it is common to couple the regularization parameter η to the step-size μ and let $\eta = \mu^{-1}$, resulting in:

$$\mathcal{W}_i = \mathcal{A}^{\mathsf{T}}\,\mathcal{W}_{i-1} - \mu \nabla_{\mathcal{W}} \mathcal{J}(\mathcal{W}_{i-1}) \tag{8.18}$$

where we defined $\mathcal{A} = A \otimes I_M$ with:

$$A \triangleq I - L \tag{8.19}$$

While (8.18) is a useful and compact representation for studying the evolution of iterates across the network and over time, returning to the agent-level quantities will interesting intuition. To this end, note first that the gradient of the aggregate cost $\mathcal{J}(\mathcal{W})$, in light of its sum-structure (8.11), inherits the block-structure of $\mathcal{W}$, and specifically:

$$\nabla_{\mathcal{W}} \mathcal{J}(\mathcal{W}_{i-1}) = \begin{pmatrix} \nabla_{w_1} J_1(w_{1,i-1}) \\ \nabla_{w_2} J_2(w_{2,i-1}) \\ \vdots \\ \nabla_{w_K} J_K(w_{K,i-1}) \end{pmatrix} \tag{8.20}$$

We can then expand (8.18):

$$\begin{aligned} \begin{pmatrix} w_{1,i-1} \\ w_{2,i-1} \\ \vdots \\ w_{K,i-1} \end{pmatrix} &= \mathcal{A}^{\mathsf{T}} \begin{pmatrix} w_{1,i-1} \\ w_{2,i-1} \\ \vdots \\ w_{K,i-1} \end{pmatrix} - \mu \begin{pmatrix} \nabla_{w_1} J_1(w_{1,i-1}) \\ \nabla_{w_2} J_2(w_{2,i-1}) \\ \vdots \\ \nabla_{w_K} J_K(w_{K,i-1}) \end{pmatrix} \\ &= \begin{pmatrix} \sum_{\ell=1}^K a_{\ell 1} w_{\ell,i-1} \\ \sum_{\ell=1}^K a_{\ell 2} w_{\ell,i-1} \\ \vdots \\ \sum_{\ell=1}^K a_{\ell K} w_{\ell,i-1} \end{pmatrix} - \mu \begin{pmatrix} \nabla_{w_1} J_1(w_{1,i-1}) \\ \nabla_{w_2} J_2(w_{2,i-1}) \\ \vdots \\ \nabla_{w_K} J_K(w_{K,i-1}) \end{pmatrix} \end{aligned} \tag{8.21}$$

Since the recursion has now been fully decoupled into blocks, we can simply write for the k-th block:

$$w_{k,i} = \sum_{\ell=1}^{K} a_{\ell k} w_{\ell,i-1} - \mu \nabla_{w_k} J_k(w_{k,i-1}) \tag{8.22}$$

By construction, recursion (8.22) is fully decentralized, and every iteration requires the evaluation of a single *local* gradient $\nabla_{w_k} J_k(w_{k,i-1})$ along with aggregation of estimates $\sum_{\ell=1}^K a_{\ell k} w_{\ell,i-1}$. To verify that the aggregation step can indeed be performed over a graph, recall that the penalized formulation (8.5) is based on the problem (8.3), rather than (8.2). While (8.2) and (8.3) are equivalent in the sense that solving either (8.2) or (8.3) is equivalent to solving (8.1), penalizing (8.3), and hence only penalizing differences within neighborhoods, ensures that:

$$c_{\ell k} = 0 \quad \forall \quad k \notin \mathcal{N}_\ell \tag{8.23}$$

By definition, we have that:

$$A \overset{(8.19)}{=} I - L \overset{(8.14)}{=} I - (\text{diag}\{C\mathbb{1}\} - C) \tag{8.24}$$

and hence the combination matrix A inherits the sparsity structure from C. Then:

$$a_{\ell k} = 0 \quad \forall \quad k \notin \mathcal{N}_\ell \tag{8.25}$$

and (8.22) is equivalent to:

$$w_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} w_{\ell,i-1} - \mu \nabla_{w_k} J_k(w_{k,i-1}) \tag{8.26}$$

Lemma 8.1 (**Properties of the Combination Matrix**). *The combination matrix $A = I - L$ with $L = (\text{diag}\{C\mathbb{1}\} - C)$ is primitive and doubly-stochastic, satisfying:*

$$A = A^\mathsf{T} \tag{8.27}$$

$$A\mathbb{1} = \mathbb{1} \tag{8.28}$$

Proof: Symmetry of A follows directly from symmetry of C. For the eigenvector condition, we have:

$$A\mathbb{1} = (I - L)\mathbb{1} = \mathbb{1} - L\mathbb{1} = \mathbb{1} - (\text{diag}\{C\mathbb{1}\} - C)\,\mathbb{1} = \mathbb{1} \tag{8.29}$$

□

8.2.3 Bias of the Consensus+Innovations Algorithm

Since the consensus algorithm (8.26) corresponds to a gradient descent recursion on the penalized cost (8.15), we know its limiting point to be:

$$\mathcal{w}_\eta^o \triangleq \arg\min_{\mathcal{W}} \mathcal{J}(\mathcal{w}) + \frac{\eta}{2}\, \mathcal{w}^\mathsf{T} \mathcal{L}\, \mathcal{w} \tag{8.30}$$

We contrast this limiting point to the minimizer of the actual aggregate cost:

$$w^o \triangleq \arg\min_{w} \sum_{k=1}^{K} J_k(w) \tag{8.31}$$

Since $\mathcal{w}_\eta^o$ is the minimizer of (8.30), we have by the optimality conditions:

$$\nabla \mathcal{J}(\mathcal{w}_\eta^o) + \eta \mathcal{L}\, \mathcal{w}_\eta^o = 0 \tag{8.32}$$

Let us assume that $\mathcal{w}_\eta^o = \mathbb{1} \otimes w^o$. Since $L\mathbb{1} = 0$, it holds that:

$$\mathcal{L}\,(\mathbb{1} \otimes w^o) = 0 \tag{8.33}$$

and hence (8.32) is equivalent to:

$$\nabla \mathcal{J}(\mathbb{1} \otimes w^o) = 0 \Longleftrightarrow \nabla J_k(w^o) = 0 \tag{8.34}$$

However, this would imply that w^o is the minimizer for each individual local cost $J_k(w)$. This does not hold in general, and hence we conclude that the consensus+innovations algorithm exhibits a bias whenever the local costs $J_k(w)$ have distinct local minimizers.

8.3 PRIMAL-DUAL-BASED ALGORITHMS

We will now present an alternative methodology for deriving decentralized algorithms for the pursuit of (8.1). This class of algorithms directly tackles the extended, constrained problem (8.3) by introducing dual variables, rather than transforming (8.3) into a penalized and unconstrained, but biased problem (8.5). This construction avoids the addition of a bias, and is hence particularly useful in cases where exact gradient vectors are available and exact convergence is desirable.

In constructing the penalty-based consensus and diffusion approaches, we introduced the penalty term $\frac{1}{2}\sum_{k=1}^{K}\sum_{\ell\in\mathcal{N}_k} c_{\ell k}\|w_k - w_\ell\|^2$ and added it as a penalty to the extended cost $\mathcal{J}(\mathcal{W})$. Instead, we will now leave the same penalty term as a constraint and force it to be equal to zero, resulting in:

$$\min_{w_k} \sum_{k=1}^{K} J_k(w_k) \text{ subject to } \frac{1}{2}\sum_{k=1}^{K}\sum_{\ell\in\mathcal{N}_k} c_{\ell k}\|w_k - w_\ell\|^2 = 0 \tag{8.35}$$

It can be readily verified, that problem (8.37) is equivalent to problem (8.3), and hence (8.1), as long as the graph is connected. Indeed, we have that:

$$\frac{1}{2}\sum_{k=1}^{K}\sum_{\ell\in\mathcal{N}_k} c_{\ell k}\|w_k - w_\ell\|^2 = 0 \Longleftrightarrow w_k = w_\ell \ \forall\ \ell \in \mathcal{N}_k \tag{8.36}$$

which precisely corresponds to the set of constraints in (8.3). Following arguments analogous to those in section 8.2.1, we can write more compactly:

$$\min_{\mathcal{W}} \mathcal{J}(\mathcal{W}) \text{ subject to } \frac{1}{2}\mathcal{W}^{\mathsf{T}}\mathcal{L}\,\mathcal{W} = 0 \tag{8.37}$$

Since $L = L^{\mathsf{T}}$ is symmetric and positive semi-definite, it admits a square root of the form:

$$L = BB = B^{\mathsf{T}}B \tag{8.38}$$

where $B = B^{\mathsf{T}}$. We then have:

$$0 = \frac{1}{2}\mathcal{W}^{\mathsf{T}}\mathcal{L}\,\mathcal{W} = \frac{1}{2}\mathcal{W}^{\mathsf{T}}\mathcal{B}^{\mathsf{T}}\mathcal{B}\,\mathcal{W} = \frac{1}{2}\|\mathcal{B}\,\mathcal{W}\|^2 \Longleftrightarrow \mathcal{B}\,\mathcal{W} = 0 \tag{8.39}$$

We can hence reformulate (8.37) into the equivalent problem:

$$\min_{\mathcal{W}} \mathcal{J}(\mathcal{W}) \text{ subject to } \mathcal{B}\,\mathcal{W} = 0 \tag{8.40}$$

The augmented Lagrangian for problem (8.40) corresponds to:

$$\mathcal{L}(\mathcal{w},\lambda) = \mathcal{J}(\mathcal{w}) + \eta\lambda^{\mathsf{T}}\mathcal{B}\,\mathcal{w} + \frac{\eta}{2}\|\mathcal{B}\,\mathcal{w}\|^2 = \mathcal{J}(\mathcal{w}) + \eta\lambda^{\mathsf{T}}\mathcal{B}\,\mathcal{w} + \frac{\eta}{2}\,\mathcal{w}^{\mathsf{T}}\,\mathcal{L}\,\mathcal{w} \tag{8.41}$$

We can then alternately perform a gradient *descent* step on the primal variable $\mathcal{w}$ along with a gradient *ascent* step on the dual variable λ. The resulting algorithm amounts to:

$$\begin{aligned}\mathcal{w}_i &= \mathcal{w}_{i-1} - \mu\nabla\mathcal{J}(\mathcal{w}_{i-1}) - \mu\eta\mathcal{B}^{\mathsf{T}}\lambda_{i-1} - \mu\eta\mathcal{L}\,\mathcal{w}_{i-1}\\ &= (I - \mu\eta\mathcal{L})\,\mathcal{w}_{i-1} - \mu\nabla\mathcal{J}(\mathcal{w}_{i-1}) - \mu\eta\mathcal{B}^{\mathsf{T}}\lambda_{i-1}\\ &= \mathcal{A}^{\mathsf{T}}\,\mathcal{w}_{i-1} - \mu\nabla\mathcal{J}(\mathcal{w}_{i-1}) - \mu\eta\mathcal{B}^{\mathsf{T}}\lambda_{i-1} \qquad (8.42)\\ \lambda_i &= \lambda_{i-1} + \mu\eta\mathcal{B}\,\mathcal{w}_i \qquad (8.43)\end{aligned}$$

A closer examination of the primal update (8.42) and comparison with the consensus recursion (8.18) unveils that the primal update (8.42) essentially corresponds to a consensus update with an additional correction term $-\mu\mathcal{B}\lambda_{i-1}$ that is governed by the dual recursion (8.43). It is this corrections step that ensures eventual convergence to the minimizer of the original aggregate problem (8.1), rather than the biased minimizer of the penalized problem (8.15).

8.3.1 The EXTRA Algorithm

An evident drawback of the primal-dual augmented Lagrangian algorithm (8.42)–(8.43) is the propagation of dual variables λ_i. These dual variables not only add complexity to the update relations at every individual node, but increase the communication requirement of the algorithm. Thankfully, the dual recursion can be eliminated following a simple observation. For the iteration from time $i-2$ to time $i-1$, we have from (8.42):

$$\mathcal{w}_{i-1} = \mathcal{A}^{\mathsf{T}}\,\mathcal{w}_{i-2} - \mu\nabla\mathcal{J}(\mathcal{w}_{i-2}) - \mu\eta\mathcal{B}^{\mathsf{T}}\lambda_{i-2} \tag{8.44}$$

Subtracting (8.44) from (8.42), we find:

$$\begin{aligned}\mathcal{w}_i - \mathcal{w}_{i-1} &= \mathcal{A}^{\mathsf{T}}\left(\mathcal{w}_{i-1} - \mathcal{w}_{i-2}\right) - \mu\left(\nabla\mathcal{J}(\mathcal{w}_{i-1}) - \nabla\mathcal{J}(\mathcal{w}_{i-2})\right) - \mu\eta\mathcal{B}^{\mathsf{T}}\left(\lambda_{i-1} - \lambda_{i-2}\right)\\ &\overset{(8.43)}{=} \mathcal{A}^{\mathsf{T}}\left(\mathcal{w}_{i-1} - \mathcal{w}_{i-2}\right) - \mu\left(\nabla\mathcal{J}(\mathcal{w}_{i-1}) - \nabla\mathcal{J}(\mathcal{w}_{i-2})\right) - \mu^2\eta^2\mathcal{B}^{\mathsf{T}}\mathcal{B}\,\mathcal{w}_{i-1}\\ &= \mathcal{A}^{\mathsf{T}}\left(\mathcal{w}_{i-1} - \mathcal{w}_{i-2}\right) - \mu\left(\nabla\mathcal{J}(\mathcal{w}_{i-1}) - \nabla\mathcal{J}(\mathcal{w}_{i-2})\right) - \mu^2\eta^2\mathcal{L}\,\mathcal{w}_{i-1}\end{aligned} \tag{8.45}$$

After setting $\eta = \mu^{-1}$ and rearranging:

$$\mathcal{w}_i = 2\mathcal{A}^{\mathsf{T}}\,\mathcal{w}_{i-1} - \mathcal{A}^{\mathsf{T}}\,\mathcal{w}_{i-2} - \mu\left(\nabla\mathcal{J}(\mathcal{w}_{i-1}) - \nabla\mathcal{J}(\mathcal{w}_{i-2})\right) \tag{8.46}$$

It can be insightful to decompose (8.46) into two subsequent steps:

$$\phi_i = \mathcal{A}^{\mathsf{T}}\,\mathcal{w}_{i-1} - \mu\nabla\mathcal{J}(\mathcal{w}_{i-1}) \tag{8.47}$$

$$\mathcal{w}_i = \phi_i + \mathcal{w}_{i-1} - \phi_{i-1} \tag{8.48}$$

Returning to node-level quantities:

$$\phi_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} w_{\ell,i-1} - \mu \nabla J_k(w_{k,i-1}) \tag{8.49}$$

$$w_{k,i} = \phi_{k,i} + w_{k,i-1} - \phi_{k,i-1} \tag{8.50}$$

We observe that (8.49) is identical to the recursion of the consensus algorithm, with the difference being that the intermediate estimate $\phi_{k,i}$ obtained through (8.49) is subsequently corrected by adding $w_{k,i-1} - \phi_{k,i-1}$. As we will see in the analysis that follows and in future chapters, the purpose of this correction term is to counteract the bias encountered by consensus, a theme that will be common throughout the other algorithms we will study in the sequel.

8.3.2 Bias of the EXTRA Algorithm

Our motivation for introducing primal-dual arguments in developing a decentralized optimization algorithm was that penalty-based algorithms, as shown in Section 8.2.3 exhibit a bias. We will study the learning dynamics of penalty-based and primal-dual learning algorithms in great detail in future chapters and quantify this bias and trade-off precisely. In this section, we simply provide a high-level justification for why we can expect primal-dual algorithms to exhibit no bias. To this end, let us assume that the EXTRA recursion (8.49)–(8.50) converges to some limiting point $\mathcal{W}_\infty$, which we leave unspecified. This *fixed-point* will satisfy:

$$\mathcal{W}_\infty = 2\mathcal{A}^\mathsf{T} \mathcal{W}_\infty - \mathcal{A}^\mathsf{T} \mathcal{W}_\infty - \mu \left(\nabla \mathcal{J}(\mathcal{W}_\infty) - \nabla \mathcal{J}(\mathcal{W}_\infty) \right) = \mathcal{A}^\mathsf{T} \mathcal{W}_\infty \tag{8.51}$$

After rearranging:

$$\left(I - \mathcal{A}^\mathsf{T} \right) \mathcal{W}_\infty = 0 \overset{(6.18)}{\Longleftrightarrow} w_{k,\infty} = w_{\ell,\infty} \ \forall\ k, \ell \tag{8.52}$$

We conclude that the limiting point $\mathcal{W}_\infty$ is *consensual*. Let us now argue that indeed $w_{k,\infty} = w^o$, meaning that $\mathcal{W}_\infty = \mathbb{1} \otimes w^o$ is optimal. If we multiply (8.42) by $\mathbb{1}^\mathsf{T} \otimes I_M$ from the left, we have:

$$\begin{aligned} &\left(\mathbb{1}^\mathsf{T} \otimes I_M \right) \mathcal{W}_i \\ =\ &\left(\mathbb{1}^\mathsf{T} \otimes I_M \right) \mathcal{A}^\mathsf{T} \mathcal{W}_{i-1} - \mu \left(\mathbb{1}^\mathsf{T} \otimes I_M \right) \nabla \mathcal{J}(\mathcal{W}_{i-1}) - \mu\eta \left(\mathbb{1}^\mathsf{T} \otimes I_M \right) \mathcal{B}^\mathsf{T} \lambda_{i-1} \\ \overset{(a)}{=}\ &\left(\mathbb{1}^\mathsf{T} \otimes I_M \right) \mathcal{W}_{i-1} - \mu \left(\mathbb{1}^\mathsf{T} \otimes I_M \right) \nabla \mathcal{J}(\mathcal{W}_{i-1}) \end{aligned} \tag{8.53}$$

where (a) follows since $A\mathbb{1} = \mathbb{1}$ and $L\mathbb{1} = 0 \Longleftrightarrow B\mathbb{1} = 0$. Evaluating this relation at $\mathcal{W}_i = \mathcal{W}_{i-1} = \mathcal{W}_\infty$. Then:

$$\left(\mathbb{1}^\mathsf{T} \otimes I_M \right) \nabla \mathcal{J}(\mathcal{W}_{i-1}) = \sum_{k=1}^{K} \nabla J_k(w_{k,\infty}) \tag{8.54}$$

But we saw in (8.52) that $\mathcal{w}_\infty$ is consensual, and hence $w_{k,\infty} = w_\infty$ for all k. Hence:

$$\sum_{k=1}^{K} \nabla J_k(w_\infty) = 0 \Longleftrightarrow \mathcal{w}_\infty = \mathbb{1} \otimes w^o \tag{8.55}$$

8.4 GRADIENT-TRACKING-BASED ALGORITHMS

An alternative approach to compensating for the bias introduced by penalty-based algorithms is by means of a technique known as *gradient-tracking*, based on the dynamic consensus algorithm we introduced in Section 7.4.2. Recall that the consensus+innovations algorithm (8.22) takes the form:

$$w_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} w_{\ell,i-1} - \mu \nabla J_k(w_{k,i-1}) \tag{8.56}$$

The mixing term on the righthand-side encourages consensus, while descent along the negative direction of the gradient $\nabla_{w_k} J_k(w_{k,i-1})$ encourages optimality. Of course descent in this case only occurs along the local objectove of agent k. If we had access to global information (say in the form of a fusion center), we would prefer to implement the recursion:

$$w_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} w_{\ell,i-1} - \frac{\mu}{K} \sum_{\ell=1}^{K} \nabla J_\ell(w_{\ell,i-1}) \tag{8.57}$$

The quantity $\frac{1}{K}\sum_{k=1} \nabla J_\ell(w_{\ell,i-1})$ is in the form of an average of the local signals $\nabla J_\ell(w_{\ell,i-1})$. We can hence leverage the dynamic consensus (7.61) recursion to compute the average in a decentralized manner:

$$w_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} w_{\ell,i-1} - \mu g_{k,i-1} \tag{8.58}$$

$$g_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} g_{\ell,i-1} + \nabla J_k(w_{k,i}) - \nabla J_k(w_{k,i-1}) \tag{8.59}$$

8.4.1 Bias of Gradient-Tracking Algorithms

Decentralized algorithms based on the gradient-tracking mechanism can generally also be shown to be unbiased. The argument is similar to the one we made for primal-dual algorithms, and is established in Prob. 8.1.

8.5 INCREMENTAL VARIANTS

So far we have developed three algorithms for decentralized optimiztion, one for each major family. Namely, we encountered the consensus+innovations algorithm (8.22) as an example from penalty-based family, the EXTRA algorithm (8.49)–(8.50) as an example of a primal-dual algorithm, and the gradient-tracking algorithm (8.58)–(8.59) as an example of a gradient-tracking-based strategy. Most popular algorithms for decentralized optimization and learning fall into one of these three families, although we can of course envision many variations. We comment in this section in particular on *incremental* variants of the algorithms we have encountered thus far. As we will see when we develop stability and convergence guarantees for these strategies in future chapters, incremental constructions yield several advantages, particularly with regards to stability over their pure counterparts.

8.5.1 Diffusion Algorithm (Penalty-Based)

Beginning with penalty-based algorithms, recal that the penalized consensus problem (8.11) served as a starting point for the consensus+innovations algorithm. Instead of taking a single gradient step relative to the objective (8.11), we can perform the update incrementally, first descending along the objective $\mathcal{J}(\mathcal{w})$ and then along the penalty term $\frac{\eta}{2}\,\mathcal{w}^{\mathsf{T}}\,\mathcal{L}\,\mathcal{w}$. We find:

$$\psi_i = \mathcal{w}_{i-1} - \mu\nabla\mathcal{J}(\mathcal{w}_{i-1}) \tag{8.60}$$

$$\mathcal{w}_i = (I - \mu\eta\mathcal{L})\,\psi_i = \mathcal{A}^{\mathsf{T}}\psi_i \tag{8.61}$$

or in node quantities:

$$\psi_{k,i} = w_{k,i} - \mu\nabla J_k(w_{k,i-1}) \tag{8.62}$$

$$w_{k,i} = \sum_{\ell\in\mathcal{N}_k} a_{\ell k}\psi_{\ell,i} \tag{8.63}$$

This strategy is known as the *diffusion* strategy. We contrast this recursion with the consensus+innovations strategy, repeated here for reference:

$$w_{k,i} = \sum_{\ell\in\mathcal{N}_k} a_{\ell k} w_{\ell,i} - \mu\nabla J_k(w_{k,i-1}) \tag{8.64}$$

The distinguishing feature between the two strategies is that mixing in the case of the diffusion strategy occurs after agents have descended along their local objective function, while in the case of the consensus+innovations algorithm, agents descent *after* mixing. This subtle detail turns out to have very significant impact on the stability properties of the resulting recursions, as we will see in future chapters.

8.5.2 Exact Diffusion (Primal-Dual-Based)

We can deviate from the derivation of the EXTRA algorithm in the same incremental manner (see Prob. 8.3), and develop the *Exact diffusion* algorithm:

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} - \mu \widehat{\nabla J}_k(\boldsymbol{w}_{k,i-1}) \tag{8.65}$$

$$\boldsymbol{\phi}_{k,i} = \boldsymbol{\psi}_{k,i} + \boldsymbol{w}_{k,i-1} - \boldsymbol{\psi}_{k,i-1} \tag{8.66}$$

$$\boldsymbol{w}_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \boldsymbol{\phi}_{\ell,i-1} \tag{8.67}$$

8.5.3 (Primal-Dual-Based)

We can deviate from the derivation of the NEXT algorithm in the same incremental manner (see Prob. 8.4), and develop the *Aug-DGM* algorithm:

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{\ell,i-1} - \mu \boldsymbol{g}_{k,i-1} \tag{8.68}$$

$$\boldsymbol{\phi}_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \boldsymbol{\psi}_{\ell,i-1} \tag{8.69}$$

$$\boldsymbol{g}_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \left(\boldsymbol{g}_{\ell,i} + \widehat{\nabla J}_k(\boldsymbol{w}_{k,i-1}) - \widehat{\nabla J}_k(\boldsymbol{w}_{k,i-2}) \right) \tag{8.70}$$

8.6 PROBLEMS

8.1 Establish an analogous argument to the one in Section 8.3.2 for the gradient-tracking recursion (8.58)–(8.59).

8.2 For a deterministic optimization problem of your choice, implement the consensus+innovation, EXTRA and gradient-tracking algorithms and show that consensus+innovations exhibits a bias, while EXTRA and gradient-tracking converge exactly. How do these findings change with the choice of the step-size μ?

8.3 Show the exact incremental adjustments to the derivation of the EXTRA algorithm in Section 8.3 that lead to Exact diffusion (8.65)–(8.67).

8.4 Show the exact incremental adjustments to the derivation of the NEXT algorithm in Section 8.4 that lead to Aug-DGM (8.68)–(8.70).