Multi-Agent Optimization and Learning
Lecture 4: Algorithms for Decentralized Optimization and Learning

Stefan Vlaski[†] and Ali H. Sayed[⋆]

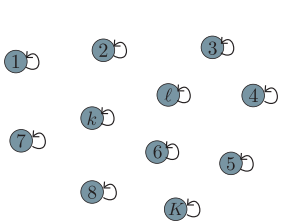
[†]Department of Electrical and Electronic Engineering, Imperial College London, UK
[⋆]Adaptive Systems Laboratory, École Polytechnique Fédérale de Lausanne, Switzerland

IEEE ICASSP 2024 Short Course

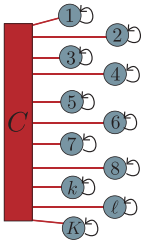
Imperial College London

EPFL

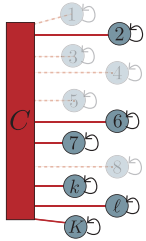
Communication Paradigms



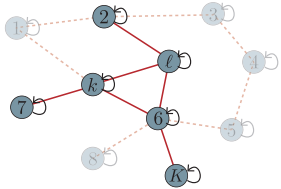
Non-cooperative



Centralized or parallel



Federated



Networked or decentralized

- In Lecture 2 we studied centralized and federated learning algorithms for multi-agent systems. These rely on a fusion center.
- Decentralized architectures instead rely on peer-to-peer interactions over a graph.
  - Robustness to node and link failure, communication efficiency and privacy considerations.
- We laid the groundwork in Lecture 3 by studying graphs and the fundamental problem of averaging over graphs. We now expand on these concepts to develop decentralized optimization algorithms.

Lecture Aims

- In this lecture we will systematically develop the most common algorithms for decentralized optimization and learning, which fall into the following three families:
  - **Penalty-based:** Distributed gradient descent [Tsitsiklis, Bertsekas and Athans 1986]
  - **Primal-dual:** Augmented Lagrangian [Towfic and Sayed 2015] and EXTRA [Shi, Ling, Wu and Yin 2015]
  - **Gradient-tracking:** DIGing [Nedić, Olshevsky and Shi 2017]
- We will also see how incremental arguments can be made to develop adapt-then-combine (ATC) variants of these algorithms with improved stability and performance properties:
  - **Penalty-based:** Diffusion [Lopes and Sayed 2008, Chen and Sayed 2012]
  - **Primal-dual:** Exact diffusion [Yuan, Ying, Zhao and Sayed 2017]
  - **Gradient-tracking:** NEXT [Di Lorenzo and Scutari 2016] and Aug-DGM [Xu, Zhu, Soh and Xie 2015]
- We build on optimization techniques from Lectures 1 and 2, and graph properties from Lecture 3.

From Global to Local Optimization Problems

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

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

Observe that all local cost functions $J_k(\cdot)$ are evaluated at a common w and hence, problem (1) becomes a *global* optimization problem. We'd like to remove global variable by replacement through local variables. To this end, we examine more closely the following alternative formulation:

$$\min_{w_k} \sum_{k=1}^{K} J_k(w_k) \text{ subject to } w_k = w_\ell \ \forall \ \ell \in \mathcal{N}_k. \tag{2}$$

where $\mathcal{N}_k$ denotes the neighborhood of node k. It is evident that (2) is equivalent to (1) as long as the graph is connected.

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:

$$\min_{w_k} \sum_{k=1}^{K} J_k(w_k) + \frac{\eta}{4} \sum_{k=1}^{K} \sum_{\ell \in \mathcal{N}_k} c_{\ell k} \|w_k - w_\ell\|^2 \tag{3}$$

where $c_{k\ell} = c_{\ell k} > 0$ 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 (3) to (2), we observe that disagreement between neighboring agents, i.e. deviations from the constraint set in (3), are penalized with weight $\eta c_{\ell k}$. The aggregate regularizer $\frac{\eta}{4} \sum_{k=1}^{K} \sum_{\ell \in \mathcal{N}_k} c_{\ell k} \|w_k - w_\ell\|^2$ is a useful measure of the variability of the weights w_k across the graph by means of local deviations $\|w_k - w_\ell\|^2$.

Network Quantities

The collection of variables w_k for $k = 1, \ldots, K$ describes the state of the network across agents. We define *network quantities*:

$$\mathcal{W} \triangleq \operatorname{col}\{w_1, w_2, \ldots, w_K\} \in \mathbb{R}^{MK} \tag{4}$$

Note that $\mathcal{W}$ is obtained by stacking w_k vertically, resulting in a vector of length MK. We will also let:

$$\mathcal{J}(\mathcal{W}) \triangleq \sum_{k=1}^{K} J_k(w_k) \tag{5}$$

We can then write (3) more compactly as:

$$\min_{\mathcal{W}} \mathcal{J}(\mathcal{W}) + \frac{\eta}{4} \sum_{k=1}^{K} \sum_{\ell \in \mathcal{N}_k} c_{\ell k} \|w_k - w_\ell\|^2 \tag{6}$$

Graph Laplacian

We have seen before that the term $\frac{\eta}{4}\sum_{k=1}^{K}\sum_{\ell\in\mathcal{N}_k} c_{\ell k}\|w_k - w_\ell\|^2$ consisting of the sum of pairwise penalties can also be rewritten more compactly in terms of the network quantity $\mathcal{W}$. Specifically, we have:

$$\frac{\eta}{4}\sum_{k=1}^{K}\sum_{\ell\in\mathcal{N}_k} c_{\ell k}\|w_k - w_\ell\|^2 = \frac{\eta}{2}\mathcal{W}^{\mathsf{T}}\mathcal{L}\mathcal{W} \tag{7}$$

where $\mathcal{L} = L \otimes I_M$ and we define the graph Laplacian matrix:

$$L = \operatorname{diag}\{C\mathbb{1}\} - C \tag{8}$$

Under those definitions we find:

$$\min_{\mathcal{W}} \mathcal{J}(\mathcal{W}) + \frac{\eta}{2}\mathcal{W}^{\mathsf{T}}\mathcal{L}\mathcal{W} \tag{9}$$

For brevity, we shall define:

$$\mathcal{J}^{\eta}(\mathcal{W}) \triangleq \mathcal{J}(\mathcal{W}) + \frac{\eta}{2}\mathcal{W}^{\mathsf{T}}\mathcal{L}\mathcal{W} \tag{10}$$

The regularization term $\frac{\eta}{2}\mathcal{W}^{\mathsf{T}}\mathcal{L}\mathcal{W}$ encourages solutions where individual blocks w_k and w_ℓ are close in the sense of the squared Euclidean distance.

Numerical Example

Let us consider a graph consisting of three nodes, arranged in an undirected line graph with symmetric weight matrix:

$$C = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \Longleftrightarrow L = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \tag{11}$$

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

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

$$\{w_1 = 1\} \qquad \{w_2 = 2\} \qquad \{w_3 = 3\}$$
$$\{w_1' = 2\} \qquad \{w_2' = 1\} \qquad \{w_3' = 3\}$$

Numerical Example

Note that both w and w' have the same energy, i.e., $\| w \|^2 = \| w' \|^2 = 14$. Nevertheless, if we compute the variation measure, we find:

$$w^{\mathsf{T}} \mathcal{L} w = 2 \text{ and } w'^{\mathsf{T}} \mathcal{L} w' = 5 \tag{12}$$

and hence $w^{\mathsf{T}} \mathcal{L} w < w'^{\mathsf{T}} \mathcal{L} w'$. 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 central.

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 threshold. We then generate $\mathcal{W}$ by sampling from a Gaussian Markov random field with distribution

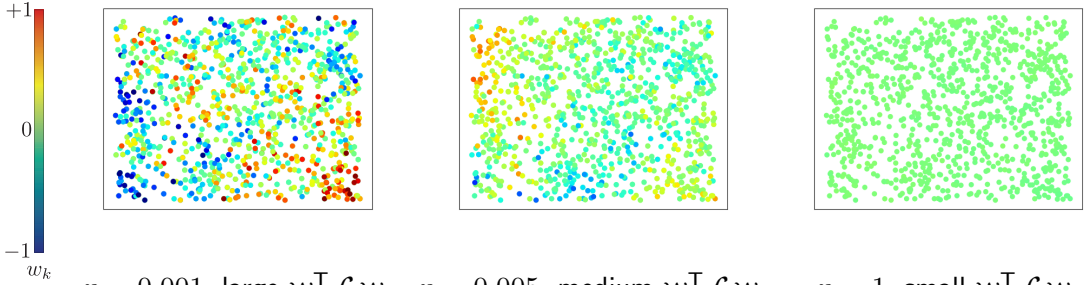
$$f(\mathcal{W}) = (2\pi\eta)^{-M(K-1)/2} \left(|\mathcal{L}|^*\right)^{1/2} e^{-\frac{\eta}{2}\mathcal{W}^{\mathsf{T}}\mathcal{L}\mathcal{W}} \tag{13}$$

Here, $|\mathcal{L}|^*$ denotes the pseudo-determinant of $\mathcal{L}$, i.e., the product of its non-zero eigenvalues. The quantity η represents a temperature parameter and smaller η is more likely to generate weight vectors w_k with large graphical variability, as measured by:

$$\mathcal{W}^{\mathsf{T}}\mathcal{L}\mathcal{W} = \frac{\eta}{2}\sum_{k=1}^{K}\sum_{\ell\in\mathcal{N}_k} c_{\ell k}\|w_k - w_\ell\|^2 \tag{14}$$

A Visual Example

We show the resulting distributions for varying η. Nodes are represented by dots and their local $w_k \in \mathbb{R}$ is represented by the color. Edge weights are determined by Euclidean distance.



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

Figure: Gauss Markov random field samples.

The Distributed Gradient Descent (DGD) Algorithm

Applying gradient-descent to (9), we obtain the recursion:

$$\begin{aligned}\mathcal{W}_i &= \mathcal{W}_{i-1} - \mu\left(\nabla\mathcal{J}(\mathcal{W}_{i-1}) + \eta\mathcal{L}\,\mathcal{W}_{i-1}\right) \\ &= (I - \mu\eta\mathcal{L})\,\mathcal{W}_{i-1} - \mu\nabla\mathcal{J}(\mathcal{W}_{i-1})\end{aligned} \tag{15}$$

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

$$\mathcal{W}_i = \mathcal{A}^{\mathsf{T}}\,\mathcal{W}_{i-1} - \mu\nabla\mathcal{J}(\mathcal{W}_{i-1}) \tag{16}$$

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

$$A \triangleq I - L \tag{17}$$

Returning to Node-Level Quantities

We'd like to return to node-level quantities. To this end, note first that the gradient of the aggregate cost $\mathcal{J}(\mathcal{W})$, in light of its sum-structure (5), inherits the block-structure of $\mathcal{W}$, and specifically:

$$\nabla_{\mathcal{W}^{\mathsf{T}}}\mathcal{J}(\mathcal{W}_{i-1}) = \begin{pmatrix} \nabla_{w_1^{\mathsf{T}}} J_1(w_{1,i-1}) \\ \nabla_{w_2^{\mathsf{T}}} J_2(w_{2,i-1}) \\ \vdots \\ \nabla_{w_K^{\mathsf{T}}} J_K(w_{K,i-1}) \end{pmatrix} \tag{18}$$

We can then expand (16):

$$\begin{pmatrix} w_{1,i} \\ w_{2,i} \\ \vdots \\ w_{K,i} \end{pmatrix} = \mathcal{A}^{\mathsf{T}} \begin{pmatrix} w_{1,i-1} \\ w_{2,i-1} \\ \vdots \\ w_{K,i-1} \end{pmatrix} - \mu \begin{pmatrix} \nabla J_1(w_{1,i-1}) \\ \nabla J_2(w_{2,i-1}) \\ \vdots \\ \nabla J_K(w_{K,i-1}) \end{pmatrix} = \begin{pmatrix} \sum_{\ell=1}^{K} a_{\ell 1} w_{\ell,i-1} \\ \sum_{\ell=1}^{K} a_{\ell 2} w_{\ell,i-1} \\ \vdots \\ \sum_{\ell=1}^{K} a_{\ell K} w_{\ell,i-1} \end{pmatrix} - \mu \begin{pmatrix} \nabla J_1(w_{1,i-1}) \\ \nabla J_2(w_{2,i-1}) \\ \vdots \\ \nabla J_K(w_{K,i-1}) \end{pmatrix} \tag{19}$$

Returning to Node-Level Quantities

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

$$w_{k,i} = \sum_{\ell=1}^{K} a_{\ell k} w_{\ell,i-1} - \mu \nabla J_k(w_{k,i-1}) \tag{20}$$

To verify that the aggregation step can indeed be performed over a graph, recall that:

$$c_{\ell k} = 0 \quad \forall \quad \ell \notin \mathcal{N}_k \tag{21}$$

By definition, we have that:

$$A \overset{(17)}{=} I - L \overset{(8)}{=} I - (\mathrm{diag}\{C\mathbb{1}\} - C) \tag{22}$$

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

$$a_{\ell k} = 0 \quad \forall \quad \ell \notin \mathcal{N}_k \tag{23}$$

and (20) is equivalent to:

$$w_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} w_{\ell,i-1} - \mu \nabla J_k(w_{k,i-1}) \tag{24}$$

Diffusion Algorithm

We now develop the adapt-then-combine (ATC) diffusion algorithm, which can be interpreted as an incremental variant of DGD. Recalling the penalized objective:

$$\min_{\mathcal{W}} \mathcal{J}(\mathcal{W}) + \frac{\eta}{2} \mathcal{W}^{\mathsf{T}} \mathcal{L} \mathcal{W} \tag{25}$$

if we take a gradient step first along $\mathcal{J}(\mathcal{W})$ and subsequently along $\frac{\eta}{2} \mathcal{W}^{\mathsf{T}} \mathcal{L} \mathcal{W}$, we obtain:

$$\psi_i = \mathcal{W}_{i-1} - \mu \nabla \mathcal{J}(\mathcal{W}_{i-1}) \tag{26}$$

$$\mathcal{W}_i = (I - \mu\eta\mathcal{L})\, \psi_i = \mathcal{A}^{\mathsf{T}} \psi_i \tag{27}$$

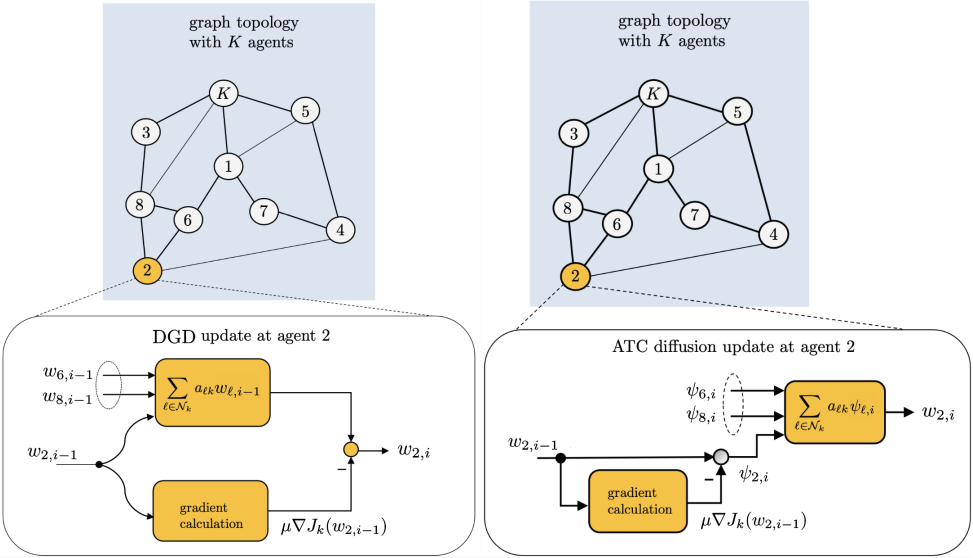
or in node quantities:

$$\psi_{k,i} = w_{k,i-1} - \mu \nabla J_k(w_{k,i-1}) \tag{28}$$

$$w_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \psi_{\ell,i} \tag{29}$$

This strategy is known as the *diffusion* strategy. It has an *adapt-then-combine* (ATC) form, since the adaptation step (28) and the combination step (29) occur in sequence.

Penalty-Based Algorithms for Decentralized Optimization



Bias of the DGD Algorithm

Since the DGD algorithm (24) corresponds to a gradient-descent recursion on the penalized cost (9), we know its limiting point to be:

$$\mathcal{W}_\eta^o \triangleq \arg\min_{\mathcal{W}} \mathcal{J}(\mathcal{W}) + \frac{\eta}{2}\mathcal{W}^{\mathsf{T}}\mathcal{L}\mathcal{W} \tag{30}$$

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

$$w^o \triangleq \arg\min_{w} \sum_{k=1}^{K} J_k(w) \tag{31}$$

Since $\mathcal{W}_\eta^o$ is the minimizer of (30), we have by the optimality conditions:

$$\nabla\mathcal{J}(\mathcal{W}_\eta^o) + \eta\mathcal{L}\mathcal{W}_\eta^o = 0 \tag{32}$$

Bias of the DGD Algorithm

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

$$\mathcal{L}\left(\mathbb{1} \otimes w^o\right) = 0 \tag{33}$$

and hence (32) is equivalent to:

$$\nabla \mathcal{J}(\mathbb{1} \otimes w^o) = 0 \Longleftrightarrow \nabla J_k(w^o) = 0 \tag{34}$$

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 distributed gradient descent algorithm exhibits a bias whenever the local costs $J_k(w)$ have distinct local minimizers.

Bias-Correction For Decentralized Optimization

- We have seen how the penalty-method can be used to motivate the DGD and diffusion strategies for decentralized optimization.
- The resulting algorithms are simple and effective, but exhibit a bias unless $\mu \to 0$.
- We now proceed to develop two families of algorithms relying on:
  - Primal-dual techniques
  - Gradient-tracking techniques
- In both cases, this will result in removal of the bias at the expense of some additional computations and/or communication.

Primal-Dual Algorithms

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:

$$\min_{w_k} \sum_{k=1}^{K} J_k(w_k) \text{ subject to } \frac{1}{4}\sum_{k=1}^{K}\sum_{\ell\in\mathcal{N}_k} c_{\ell k}\|w_k - w_\ell\|^2 = 0 \tag{35}$$

It can be readily verified, that problem (37) is equivalent to problem (2), and hence (1), as long as the graph is connected. Indeed, we have that:

$$\frac{1}{4}\sum_{k=1}^{K}\sum_{\ell\in\mathcal{N}_k} c_{\ell k}\|w_k - w_\ell\|^2 = 0 \Longleftrightarrow w_k = w_\ell \;\forall\; \ell \in \mathcal{N}_k \tag{36}$$

which precisely corresponds to the set of constraints in (2).

Primal-Dual Algorithms

As before, we can write more compactly:

$$\min_{\mathcal{W}} \mathcal{J}(\mathcal{W}) \text{ subject to } \frac{1}{2}\mathcal{W}^{\mathsf{T}}\mathcal{L}\mathcal{W} = 0 \tag{37}$$

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

$$L = BB = B^{\mathsf{T}}B \tag{38}$$

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

$$0 = \frac{1}{2}\mathcal{W}^{\mathsf{T}}\mathcal{L}\mathcal{W} = \frac{1}{2}\mathcal{W}^{\mathsf{T}}\mathcal{B}^{\mathsf{T}}\mathcal{B}\mathcal{W} = \frac{1}{2}\|\mathcal{B}\mathcal{W}\|^2 \iff \mathcal{B}\mathcal{W} = 0 \tag{39}$$

We can hence reformulate (37) into the equivalent problem:

$$\min_{\mathcal{W}} \mathcal{J}(\mathcal{W}) \text{ subject to } \mathcal{B}\mathcal{W} = 0 \tag{40}$$

Primal-Dual Algorithms

The augmented Lagrangian for problem (40) corresponds to:

$$\mathcal{L}(\mathcal{W},\lambda) = \mathcal{J}(\mathcal{W}) + \eta\lambda^{\mathsf{T}}\mathcal{B}\,\mathcal{W} + \frac{\eta}{2}\|\mathcal{B}\,\mathcal{W}\|^2 = \mathcal{J}(\mathcal{W}) + \eta\lambda^{\mathsf{T}}\mathcal{B}\,\mathcal{W} + \frac{\eta}{2}\,\mathcal{W}^{\mathsf{T}}\,\mathcal{L}\,\mathcal{W} \tag{41}$$

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:

$$\begin{aligned}
\mathcal{W}_i &= \mathcal{W}_{i-1} - \mu\nabla\mathcal{J}(\mathcal{W}_{i-1}) - \mu\eta\mathcal{B}^{\mathsf{T}}\lambda_{i-1} - \mu\eta\mathcal{L}\,\mathcal{W}_{i-1} \\
&= (I - \mu\eta\mathcal{L})\,\mathcal{W}_{i-1} - \mu\nabla\mathcal{J}(\mathcal{W}_{i-1}) - \mu\eta\mathcal{B}^{\mathsf{T}}\lambda_{i-1} \\
&= \mathcal{A}^{\mathsf{T}}\,\mathcal{W}_{i-1} - \mu\nabla\mathcal{J}(\mathcal{W}_{i-1}) - \mu\eta\mathcal{B}^{\mathsf{T}}\lambda_{i-1} \qquad (42) \\
\lambda_i &= \lambda_{i-1} + \mu\eta\mathcal{B}\,\mathcal{W}_i \qquad (43)
\end{aligned}$$

Comparison with the DGD recursion (16) unveils that the primal update (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 (43)

EXTRA Algorithm

An evident drawback of the primal-dual augmented Lagrangian algorithm (42)–(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 (42):

$$\mathcal{W}_{i-1} = \mathcal{A}^{\mathsf{T}}\,\mathcal{W}_{i-2} - \mu\nabla\mathcal{J}(\mathcal{W}_{i-2}) - \mu\eta\mathcal{B}^{\mathsf{T}}\lambda_{i-2} \tag{44}$$

Subtracting (44) from (42), we find:

$$\begin{aligned}
\mathcal{W}_i - \mathcal{W}_{i-1} &= \mathcal{A}^{\mathsf{T}}\left(\mathcal{W}_{i-1} - \mathcal{W}_{i-2}\right) - \mu\left(\nabla\mathcal{J}(\mathcal{W}_{i-1}) - \nabla\mathcal{J}(\mathcal{W}_{i-2})\right) - \mu\eta\mathcal{B}^{\mathsf{T}}\left(\lambda_{i-1} - \lambda_{i-2}\right) \\
&\overset{(43)}{=} \mathcal{A}^{\mathsf{T}}\left(\mathcal{W}_{i-1} - \mathcal{W}_{i-2}\right) - \mu\left(\nabla\mathcal{J}(\mathcal{W}_{i-1}) - \nabla\mathcal{J}(\mathcal{W}_{i-2})\right) - \mu^2\eta^2\mathcal{B}^{\mathsf{T}}\mathcal{B}\,\mathcal{W}_{i-1} \\
&= \mathcal{A}^{\mathsf{T}}\left(\mathcal{W}_{i-1} - \mathcal{W}_{i-2}\right) - \mu\left(\nabla\mathcal{J}(\mathcal{W}_{i-1}) - \nabla\mathcal{J}(\mathcal{W}_{i-2})\right) - \mu^2\eta^2\mathcal{L}\,\mathcal{W}_{i-1}
\end{aligned} \tag{45}$$

where the last line follows since $\mathcal{B}$ is the symmetric square root of $\mathcal{L}$.

EXTRA Algorithm

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

$$\mathcal{W}_i = 2\mathcal{A}^{\mathsf{T}}\mathcal{W}_{i-1} - \mathcal{A}^{\mathsf{T}}\mathcal{W}_{i-2} - \mu\left(\nabla\mathcal{J}(\mathcal{W}_{i-1}) - \nabla\mathcal{J}(\mathcal{W}_{i-2})\right) \tag{46}$$

It can be insightful to decompose (46) into two subsequent steps:

$$\phi_i = \mathcal{A}^{\mathsf{T}}\mathcal{W}_{i-1} - \mu\nabla\mathcal{J}(\mathcal{W}_{i-1}) \tag{47}$$

$$\mathcal{W}_i = \phi_i + \mathcal{A}^{\mathsf{T}}\mathcal{W}_{i-1} - \phi_{i-1} \tag{48}$$

Returning to node-level quantities:

$$\phi_{k,i} = \sum_{\ell\in\mathcal{N}_k} a_{\ell k} w_{\ell,i-1} - \mu\nabla J_k(w_{k,i-1}) \tag{49}$$

$$w_{k,i} = \phi_{k,i} + \sum_{\ell\in\mathcal{N}_k} a_{\ell k} w_{\ell,i-1} - \phi_{k,i-1} \tag{50}$$

The first EXTRA step is identical to DGD, but this is followed by a correction step.

Bias of the EXTRA Algorithm

Our motivation for introducing primal-dual arguments in developing a decentralized optimization algorithm was that penalty-based algorithms exhibit a bias. We will study the learning dynamics of penalty-based and primal-dual learning algorithms in great detail in the next lecture 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 (49)–(50) converges to some limiting point $\mathcal{W}_\infty$, that we leave unspecified. This *fixed-point* will satisfy:

$$\mathcal{W}_\infty = 2\mathcal{A}^{\mathsf{T}}\mathcal{W}_\infty - \mathcal{A}^{\mathsf{T}}\mathcal{W}_\infty - \mu\left(\nabla\mathcal{J}(\mathcal{W}_\infty) - \nabla\mathcal{J}(\mathcal{W}_\infty)\right) = \mathcal{A}^{\mathsf{T}}\mathcal{W}_\infty \tag{51}$$

After rearranging, it follows from Perron-Frobenius as long as A is primitive that:

$$\left(I - \mathcal{A}^{\mathsf{T}}\right)\mathcal{W}_\infty = 0 \Longleftrightarrow \mathcal{A}^{\mathsf{T}}\mathcal{W}_\infty = \mathcal{W}_\infty \Longleftrightarrow w_{k,\infty} = w_{\ell,\infty}\ \forall\ k,\ell \tag{52}$$

We conclude that the limiting point $\mathcal{W}_\infty$ is *consensual*.

Bias of the EXTRA Algorithm

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 (42) by $\mathbb{1}^\mathsf{T} \otimes I_M$ from the left, we have:

$$\begin{aligned}
&\left(\mathbb{1}^\mathsf{T} \otimes I_M\right) \mathcal{W}_i \\
&= \left(\mathbb{1}^\mathsf{T} \otimes I_M\right) \mathcal{A}^\mathsf{T} \mathcal{W}_{i-1} - \mu \left(\mathbb{1}^\mathsf{T} \otimes I_M\right) \nabla \mathcal{J}(\mathcal{W}_{i-1}) - \mu\eta \left(\mathbb{1}^\mathsf{T} \otimes I_M\right) \mathcal{B}^\mathsf{T} \lambda_{i-1} \\
&\overset{(a)}{=} \left(\mathbb{1}^\mathsf{T} \otimes I_M\right) \mathcal{W}_{i-1} - \mu \left(\mathbb{1}^\mathsf{T} \otimes I_M\right) \nabla \mathcal{J}(\mathcal{W}_{i-1})
\end{aligned} \tag{53}$$

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

$$\left(\mathbb{1}^\mathsf{T} \otimes I_M\right) \nabla \mathcal{J}(\mathcal{W}_\infty) = \sum_{k=1}^{K} \nabla J_k(w_{k,\infty}) = 0 \tag{54}$$

But we saw in (52) that $\mathcal{W}_\infty$ is consensual, and hence $w_{k,\infty} = w_\infty$ for all k and:

$$\sum_{k=1}^{K} \nabla J_k(w_\infty) = 0 \iff w_\infty = w^o \iff \mathcal{W}_\infty = \mathbb{1} \otimes w^o \tag{55}$$

Exact Diffusion Algorithm

We can deviate from the derivation of the EXTRA algorithm using incremental updates analogous to the diffusion algorithm. We begin with (41), repeated here for reference:

$$\mathcal{L}(\mathcal{W},\lambda) = \mathcal{J}(\mathcal{W}) + \eta\lambda^{\mathsf{T}}\mathcal{B}\,\mathcal{W} + \frac{\eta}{2}\,\mathcal{W}^{\mathsf{T}}\,\mathcal{L}\,\mathcal{W} \tag{56}$$

Instead of performing straight gradient-descent ascent, which yields EXTRA, we descend incrementally, first along $\mathcal{J}(\mathcal{W})$ and subsequently along the remaining terms. This yields:

$$\psi_i = \mathcal{W}_{i-1} - \mu\nabla\mathcal{J}(\mathcal{W}_{i-1}) \tag{57}$$

$$\mathcal{W}_i = \psi_i - \mu\eta\mathcal{L}\psi_i - \mu\eta\mathcal{B}^{\mathsf{T}}\lambda_{i-1} \tag{58}$$

$$\lambda_i = \lambda_{i-1} + \mu\eta\mathcal{B}\,\mathcal{W}_i \tag{59}$$

With the choice $\eta = \mu^{-1}$ and $\mathcal{A}^{\mathsf{T}} = I - \mu\eta\mathcal{L} = I - \mathcal{L}$, we have:

$$\psi_i = \mathcal{W}_{i-1} - \mu\nabla\mathcal{J}(\mathcal{W}_{i-1}) \tag{60}$$

$$\mathcal{W}_i = \mathcal{A}^{\mathsf{T}}\psi_i - \mathcal{B}^{\mathsf{T}}\lambda_{i-1} \tag{61}$$

$$\lambda_i = \lambda_{i-1} + \mathcal{B}\,\mathcal{W}_i \tag{62}$$

Exact Diffusion Algorithm

We can write this compactly as:

$$\mathcal{W}_i = \mathcal{A}^{\mathsf{T}} \left(\mathcal{W}_{i-1} - \mu \nabla \mathcal{J}(\mathcal{W}_{i-1})\right) - \mathcal{B}^{\mathsf{T}} \lambda_{i-1} \tag{63}$$

$$\lambda_i = \lambda_{i-1} + \mathcal{B}\, \mathcal{W}_i \tag{64}$$

We now follow a similar argument to EXTRA to eliminate the dual variable. The primal update at time $i-1$ is evaluated to:

$$\mathcal{W}_{i-1} = \mathcal{A}^{\mathsf{T}} \left(\mathcal{W}_{i-2} - \mu \nabla \mathcal{J}(\mathcal{W}_{i-2})\right) - \mathcal{B}^{\mathsf{T}} \lambda_{i-2} \tag{65}$$

Subtracting:

$$\begin{aligned} \mathcal{W}_i - \mathcal{W}_{i-1} &= \mathcal{A}^{\mathsf{T}} \left(\mathcal{W}_{i-1} - \mathcal{W}_{i-2} - \mu \nabla \mathcal{J}(\mathcal{W}_{i-1}) + \mu \nabla \mathcal{J}(\mathcal{W}_{i-2})\right) - \mathcal{B}^{\mathsf{T}} \left(\lambda_{i-1} - \lambda_{i-2}\right) \\ &= \mathcal{A}^{\mathsf{T}} \left(\mathcal{W}_{i-1} - \mathcal{W}_{i-2} - \mu \nabla \mathcal{J}(\mathcal{W}_{i-1}) + \mu \nabla \mathcal{J}(\mathcal{W}_{i-2})\right) - \mathcal{B}^{\mathsf{T}} \mathcal{B}\, \mathcal{W}_{i-1} \\ &= \mathcal{A}^{\mathsf{T}} \left(\mathcal{W}_{i-1} - \mathcal{W}_{i-2} - \mu \nabla \mathcal{J}(\mathcal{W}_{i-1}) + \mu \nabla \mathcal{J}(\mathcal{W}_{i-2})\right) - \mathcal{L}\, \mathcal{W}_{i-1} \end{aligned} \tag{66}$$

After rearranging:

$$\mathcal{W}_i = \mathcal{A}^{\mathsf{T}} \left(2\,\mathcal{W}_{i-1} - \mathcal{W}_{i-2} - \mu \nabla \mathcal{J}(\mathcal{W}_{i-1}) + \mu \nabla \mathcal{J}(\mathcal{W}_{i-2})\right) \tag{67}$$

Exact Diffusion Algorithm

We can formulate this relation in multiple steps as:

$$\psi_i = \mathcal{W}_{i-1} - \mu \nabla \mathcal{J}(\mathcal{W}_{i-1}) \tag{68}$$

$$\phi_i = \psi_i + \mathcal{W}_{i-1} - \psi_{i-1} \tag{69}$$

$$\mathcal{W}_i = \mathcal{A}^{\mathsf{T}} \phi_i \tag{70}$$

which is the Exact diffusion algorithm in network form. In terms of local quantities:

$$\psi_{k,i} = w_{k,i-1} - \mu \nabla J_k(w_{k,i-1}) \tag{71}$$

$$\phi_{k,i} = \psi_{k,i} + w_{k,i-1} - \psi_{k,i-1} \tag{72}$$

$$w_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \phi_{\ell,i} \tag{73}$$

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 Lecture 4. Recall that the distributed gradient descent algorithm (20) takes the form:

$$w_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} w_{\ell,i-1} - \mu \nabla J_k(w_{k,i-1}) \tag{74}$$

The mixing term on the right-hand side encourages consensus, while descent along the negative direction of the gradient $\nabla J_k(w_{k,i-1})$ encourages optimality. Of course descent in this case only occurs along the local objective 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:

$$w_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} w_{\ell,i-1} - \frac{\mu}{K} \sum_{\ell=1}^{K} \nabla J_\ell(w_{\ell,i-1}) \tag{75}$$

DIGing Algorithm

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 technique to compute the average in a decentralized manner:

$$w_{k,i} = \sum_{\ell\in\mathcal{N}_k} a_{\ell k} w_{\ell,i-1} - \mu g_{k,i-1} \tag{76}$$

$$g_{k,i} = \sum_{\ell\in\mathcal{N}_k} a_{\ell k} g_{\ell,i-1} + \nabla J_k(w_{k,i}) - \nabla J_k(w_{k,i-1}) \tag{77}$$

This version of gradient-tracking is known as *DIGing*. Gradient-tracking algorithms are also bias-free.

NEXT and Aug-DGM

We can mirror the argument leading to the diffusion algorithm in the gradient-tracking framework of DIGing, and develop the *NEXT* algorithm:

$$\psi_{k,i} = w_{k,i-1} - \mu g_{k,i-1} \tag{78}$$

$$w_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \psi_{\ell,i} \tag{79}$$

$$g_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} g_{\ell,i-1} + \nabla J_k(w_{k,i}) - \nabla J_k(w_{k,i-1}) \tag{80}$$

If additionally, we compute the gradient tracking term incrementally, we find *Aug-DGM*:

$$\psi_{k,i} = w_{k,i-1} - \mu g_{k,i-1} \tag{81}$$

$$w_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \psi_{\ell,i} \tag{82}$$

$$g_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \left(g_{\ell,i-1} + \nabla J_\ell(w_{\ell,i}) - \nabla J_\ell(w_{\ell,i-1}) \right) \tag{83}$$

Computational and Communication Complexity

Particularly when implementing decentralized algorithms on edge devices, it is important to consider implications on computational and communciation complexity.

- Penalty-based algorithms (DGD and diffusion) are the simplest and require one gradient computation and one exchange of parameters per iteration.
- Primal-dual algorithms (EXTRA and Exact diffusion) require some additional computation to perform the correction step, but still require only one gradient computation and one exchange of model parameters per iteration.
- Gradient-tracking algorithms additionally require the propagation of the tracking variable $g_{k,i}$. This requires storage of the gradient computed at the previous time instance, as well as the exchange of the tracking variable in addition to model parameters at every iteration. This approximately doubles the communication cost relative to penalty-based and primal-dual algorithms. The recursions of DIGing and NEXT allow parameters and tracking variables to be exchanged in the same time slot, while Aug-DGM requires sequential communication of parameters and tracking variables.

Conclusion and Outlook

- We developed a number of decentralized optimization algorithms falling into the three broad families:
  - **Penalty-based:** Distributed gradient descent (DGD)
  - **Primal-dual:** Augmented Lagrangian and EXTRA
  - **Gradient-tracking:** DIGing
- We also saw how incremental arguments can be made to develop adapt-then-combine (ATC) variants of these algorithms with improved stability and performance properties:
  - **Penalty-based:** Diffusion
  - **Primal-dual:** Exact diffusion
  - **Gradient-tracking:** NEXT and Aug-DGM
- Penalty-based methods exhibit a bias, while the other two families correct the bias using primal-dual or gradient-tracking techniques.
- In the next lecture, we examine in detail the performance of all of these algorithms.

References and Further Reading

- General references and surveys:
  - A. H. Sayed, *Inference and Learning from Data*, Cambridge University Press, 2022.
  - A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, July 2014.
  - A. H. Sayed, "Adaptive Networks," in *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460-497, 2014.
  - S. Vlaski, S. Kar, A. H. Sayed and J. M. F. Moura, "Networked Signal and Information Processing: Learning by multiagent systems," in *IEEE Signal Processing Magazine*, vol. 40, no. 5, pp. 92-105, July 2023,

References and Further Reading

- Specific algorithms:
  - J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.
  - C. G. Lopes and A. H. Sayed, "Diffusion Least-Mean Squares Over Adaptive Networks: Formulation and Performance Analysis," in *IEEE Trans. Signal Processing*, vol. 56, no. 7, pp. 3122-3136, July 2008.
  - J. Chen and A. H. Sayed, "Diffusion Adaptation Strategies for Distributed Optimization and Learning Over Networks," in *IEEE Trans. Signal Processing*, vol. 60, no. 8, pp. 4289-4305, Aug. 2012.
  - W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

References and Further Reading

- Specific algorithms (continued):
  - Z. J. Towfic and A. H. Sayed, "Stability and performance limits of adaptive primal-dual networks," *IEEE Trans. Signal Processing*, vol. 63, no. 11, pp. 2888–2903, 2015.
  - J. Xu, S. Zhu, Y. C. Soh, and L. Xie, "Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes," in *Proc. IEEE CDC*, 2015, pp. 2055–2060.
  - P. Di Lorenzo and G. Scutari, "NEXT: In-network nonconvex optimization," *IEEE Trans. Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 120–136, 2016.
  - K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, "Exact diffusion strategy for optimization by networked agents," in *Proc. 25th Eur. Signal Process. Conf.*, 2017.
  - A. Nedić, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017.