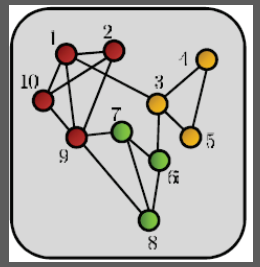


INFERENCE OVER NETWORKS

LECTURE #14: Multi-Agent Network Model

Professor Ali H. Sayed
UCLA Electrical Engineering

Course EE210B
Spring Quarter 2015

Proc. IEEE, vol. 102, no. 4, pp. 460-497, April 2014.
Foundations and Trends in Machine Learning, vol. 7, no. 4-5, pp. 311-801, July 2014.

Part IV: Multi-Agent Network Model

Course EE210B
Spring Quarter 2015

Proc. IEEE, vol. 102, no. 4, pp. 460-497, April 2014.
Foundations and Trends in Machine Learning, vol. 7, no. 4-5, pp. 311-801, July 2014.

References

Chapter 6 (Multi-Agent Network Model, pp. 368-431-447):

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, 2014.

Appendix A (Graph Laplacian and Network Connectivity):

A. H. Sayed, ``Diffusion adaptation over networks,'' in ***Academic Press Library in Signal Processing***, vol. 3, R. Chellapa and S. Theodoridis, *editors*, pp. 323-454, Academic Press, Elsevier, 2014.

Setting

Moving forward, we shall study several distributed strategies for the solution of adaptation, learning, and optimization problems by networked agents. In preparation for these discussions, we describe in this lecture the network model and comment on some of its properties.

Connected Networks

Course EE210B
Spring Quarter 2015

Proc. IEEE, vol. 102, no. 4, pp. 460-497, April 2014.
Foundations and Trends in Machine Learning, vol. 7, no. 4-5, pp. 311-801, July 2014.

Connected Networks

We focus in our treatment on *connected* networks with N agents. In a connected network, there always exists at least one path connecting any two agents: the agents may be connected directly by an edge if they are neighbors, or they may be connected by a path that passes through other intermediate agents. The topology of a network can be described in terms of graphs, vertices, and edges (e.g., [256]).

Connected Networks

Definition 6.1 (Graphs, vertices, and edges). A network of size N is generally represented by a graph consisting of N vertices (which we will refer to more frequently as nodes or agents), and a set of edges connecting the vertices to each other. An edge that connects a vertex to itself is called a self-loop. Vertices connected by edges are called neighbors.

Connected Networks

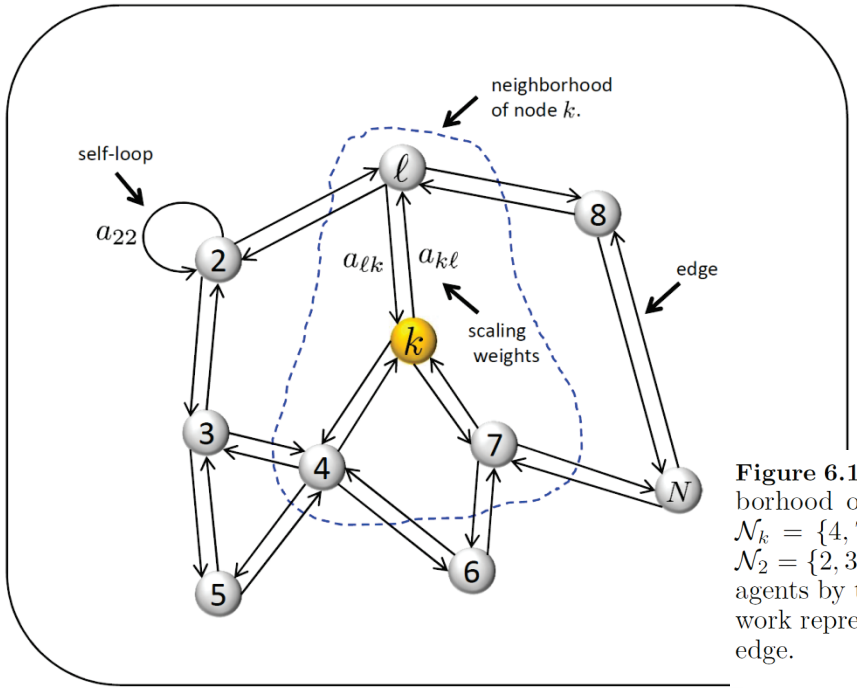


Figure 6.1: Agents that are linked by edges can share information. The neighborhood of agent k is marked by the broken line and consists of the set $\mathcal{N}_k = \{4, 7, \ell, k\}$. Likewise, the neighborhood of agent 2 consists of the set $\mathcal{N}_2 = \{2, 3, \ell\}$. For emphasis in the figure, we are representing edges between agents by two separate directed arrows with weights $\{a_{k\ell}, a_{\ell k}\}$. In future network representations, we will replace the two arrows by a single bi-directional edge.

Connected Networks

We assume the graph is undirected so that if agent k is a neighbor of agent ℓ, then agent ℓ is also a neighbor of agent k. Any two neighbors can share information both ways over the edge connecting them. This fact does not necessarily mean that the flow of information between the agents is symmetrical [208]. This is because we shall assign a pair of nonnegative weights, $\{a_{k\ell}, a_{\ell k}\}$, to the edge connecting agents k and ℓ. The scalar $a_{\ell k}$ will be used by agent k to scale data it receives from agent ℓ; this scaling can be interpreted as a measure of the confidence

Connected Networks

that agent k assigns to its interaction with agent ℓ. The subscripts ℓ and k in $a_{\ell k}$, with ℓ coming before k, designate agent ℓ as the source and agent k as the sink. Likewise, the scalar $a_{k\ell}$ will be used by agent ℓ to scale the data it receives from agent k. In this case, agent k is the source and agent ℓ is the sink. The weights $\{a_{k\ell}, a_{\ell k}\}$ can be different, and one or both weights can also be zero. We can therefore refer to the graph representing the network as a *weighted* graph with weights $\{a_{\ell k}, a_{k\ell}\}$ attached to the edges.

Connected Networks

Figure 6.1 shows one example of a connected network. For emphasis in the figure, each edge between two neighboring agents is being represented (for now) by two directed arrows to indicate that information can flow both ways between the agents. The neighborhood of any agent k is denoted by $\mathcal{N}_k$ and it consists of all agents that are connected to k by edges; we assume by default that this set includes agent k regardless of whether agent k has a self-loop or not.

Connected Networks

Definition 6.2 (Neighborhoods over weighted graphs). 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, in addition to agent k itself. Any two neighboring agents k and ℓ have the ability to share information over the edge connecting them. Whether this exchange of information occurs, and whether it is uni-directional, bi-directional, or non-existent, will depend on the values of the weighting scalars $\{a_{k\ell}, a_{\ell k}\}$ assigned to the edge.

Strongly-Connected Networks

When at least one a_{kk} is positive for some agent k, the connected network will be said to be *strongly-connected*. In other words, a strongly-connected network contains at least one self-loop, as is the case with agent 2 in Figure 6.1. More formally, we adopt the following terminology and define connected networks over weighted graphs as follows.

Network Types

Definition 6.3 (Connected networks). We distinguish between three types of connected networks; the third class of *strongly-connected* networks will be the focus of our study:

(a) Weakly-connected network: A network is said to be weakly connected if paths with nonzero scaling weights can be found linking any two distinct vertices in *at least one* direction either directly when they are neighbors or by passing through intermediate vertices when they are not neighbors. In this way, it is possible for information to flow in *at least one* direction between any two distinct vertices in the network.

Network Types

(b) Connected network: A network is said to be connected if paths with nonzero scaling weights can be found linking any two distinct vertices in *both* directions either directly when they are neighbors or by passing through intermediate vertices when they are not neighbors. In this way, information can flow in *both* directions between any two distinct vertices in the network, although the forward path from a vertex k to some other vertex ℓ need not be the same as the backward path from ℓ to k.

Network Types

(c) Strongly-connected network: A network is said to be strongly-connected if it is a connected network with at least one self-loop with a positive scaling weight, meaning that $a_{kk} > 0$ for some vertex k. In this way, information can flow in both directions between any two distinct vertices in the network and, moreover, some vertices possess self-loops with positive weights.

Example #D

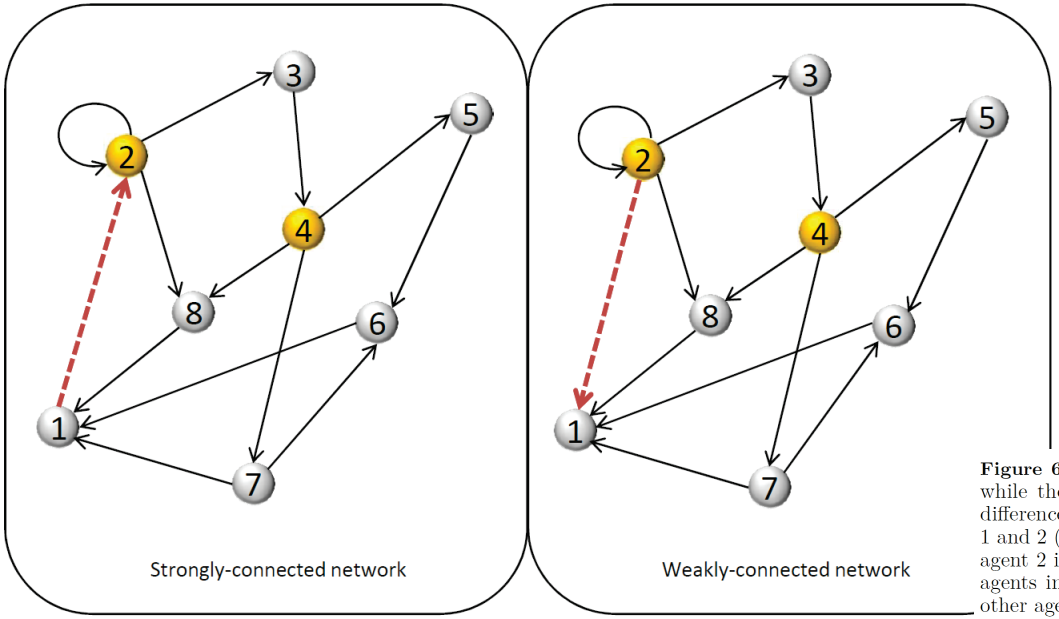


Figure 6.2: The graph on the left represents a strongly-connected network, while the graph on the right represents a weakly-connected network. The difference between both graphs is the reversal of the arrow connecting agents 1 and 2 (represented in broken form for emphasis). In the graph on the right, agent 2 is incapable of receiving (sensing) information from any of the other agents in the network, even though information from agent 2 can reach all other agents (directly or indirectly).

Example #D

Figure 6.2 illustrates these definitions by means of an example. The graph on the left represents a strongly-connected network: if we select any two agents k and ℓ, we can find paths linking them in both directions with positive weights on the edges along these paths. In the figure, we continue to represent edges between agents by two arrows. However, in order not to overwhelm the figure with combination weights, we are not showing arrows that correspond to zero weights on them; we are

Example #D

only showing arrows that correspond to positive weights. Thus, observe in the graph on the left that for agents 2 and 4, a valid path from 2 to 4 goes through agent 3 and one valid path for the reverse direction from 4 to 2 goes through agents 8 and 1. Similarly, paths can be determined linking all other combinations of agents in both directions.

Example #D

Consider now the graph on the right in Figure 6.2. In this graph, we simply reversed the direction of the arrow that emanated from agent 1 towards agent 2 in the graph on the left (and which is represented in broken form for emphasis). Observe that now information cannot reach agent 2 from any of the other agents in the network, even though information from agent 2 can reach all other agents. At the same time, the information from agent 1 cannot reach any other agent in the network and agent 1 is only at the receiving end. This graph therefore

Example #D

corresponds to a weakly-connected network. When some agents (like agent 2) are never able to receive information from other agents in the network, then these isolated agents will not be able to benefit from network interactions.

Strongly-Connected Networks

Course EE210B
Spring Quarter 2015

Proc. IEEE, vol. 102, no. 4, pp. 460-497, April 2014.
Foundations and Trends in Machine Learning, vol. 7, no. 4-5, pp. 311-801, July 2014.

Strongly-Connected Networks

Observe that since we will be dealing with weighted graphs, we are therefore defining connected networks not in terms of whether paths can be found connecting their vertices but in terms of whether these paths allow for the *meaningful* exchange of information between the vertices. This fact is reflected by the requirement that all scaling weights must be *positive* over at least one of the paths connecting any two distinct vertices.

Strongly-Connected Networks

This is a useful condition for the study of adaptation and learning over networks. As we are going to see in future chapters, agents will exchange information over the edges linking them. The information will be scaled by weights $\{a_{k\ell}, a_{\ell k}\}$. Therefore, for information to flow between agents, it is not sufficient for paths to exist linking these agents. It is also necessary that the information is not annihilated by zero scaling while it traverses the path. If information is never able to arrive at some particular agent, ℓ_o, because scaling is annihilating it before reaching ℓ_o then, for all practical (adaptation and learning) purposes, agent ℓ_o is disconnected from the other agents in the network

Strongly-Connected Networks

even if information can still flow in the other direction from agent ℓ_o to the other agents. In this situation, agent ℓ_o will not benefit from cooperation with other agents in the network, while the other agents will benefit from information provided by agent ℓ_o. The assumption of a connected network therefore ensures that information will be flowing between any two arbitrary agents in the network and that this flow of information is bi-directional: information flows from k to ℓ and from ℓ to k, although the paths over which the flows occur need not be the same and the manner by which information is scaled over these paths can also be different.

Strongly-Connected Networks

The condition of a strongly-connected network implies that the network is connected and, additionally, there is at least one agent in the network that trusts its own information and will assign some positive weight to it. This is a reasonable condition and is characteristic of many real networks, especially biological networks. If $a_{kk} = 0$ for all k, then this means that all agents will be ignoring their individual information and will be relying instead on information received from other agents. The terminology of "strongly-connected networks" is perhaps somewhat excessive because it may unnecessarily convey the impression that the network needs to have more connectivity than is actually necessary.

Strongly-Connected Networks

The strong connectivity of a network translates into a useful property to be satisfied by the scaling weights $\{a_{\ell k}\}$; this property will be exploited to great effect in our analysis so we derive it here. 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 $\ell \notin \mathcal{N}_k$ — see Figure 6.3. In this way, the row index in (ℓ, k) designates the source agent and the column index designates the sink agent (or destination).

Strongly-Connected Networks

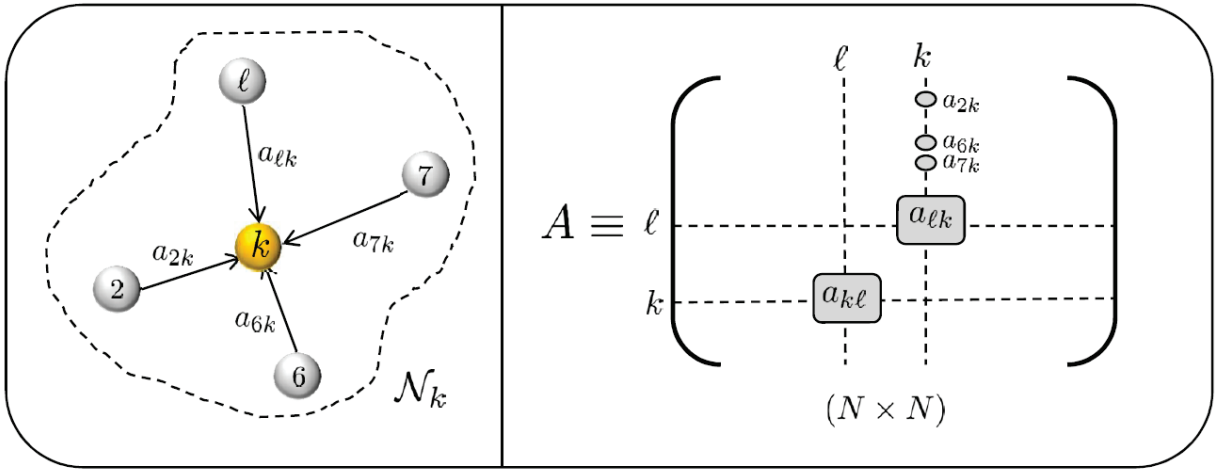


Figure 6.3: We associate an $N \times N$ combination matrix A with every network of N agents. The $(\ell, k)-$th entry of A contains the combination weight $a_{\ell k}$, which scales the data arriving at agent k and originating from agent ℓ.

Strongly-Connected Networks

We refer to A as the *combination* matrix or combination policy. Even though the entries of A are non-negative (and several of them can be zero), it turns out that for combination matrices A that originate from strongly-connected networks, there exists an integer power of A such that all its entries are strictly positive, i.e., there exists some finite integer $n_o > 0$ such that

$$[A^{n_o}]_{\ell k} > 0 \tag{6.1}$$

for all $1 \leq \ell, k \leq N$. Combination matrices that satisfy this property are called *primitive* matrices.

Strongly-Connected Networks

Lemma 6.1 (Combination matrices of strongly-connected networks). The combination matrix of a strongly-connected network is a primitive matrix.

Proof

Proof. Pick two arbitrary agents ℓ and k. Since the network is assumed to be connected, then this implies that there exists a sequence of agent indices $(\ell, m_1, m_2, \ldots, m_{n\ell k-1}, k)$ of shortest length that forms a path from agent ℓ to agent k, say, with $n_{\ell k}$ nonzero scaling weights $\{a_{\ell m_1}, a_{m_1,m_2}, \ldots, a_{m_{n\ell k-1},k}\}$:

$$\ell \xrightarrow{a_{\ell m_1}} m_1 \xrightarrow{a_{m_1,m_2}} m_2 \longrightarrow \ldots \longrightarrow m_{n\ell k-1} \xrightarrow{a_{m_{n\ell k}-1,k}} k \qquad [n_{\ell k} \text{ edges}] \tag{6.2}$$

From the rules of matrix multiplication, the $(\ell, k)-$th entry of the $n_{\ell k}-$th power of A is given by:

$$[A^{n_{\ell k}}]_{\ell k} = \sum_{m_1=1}^{N} \sum_{m_2=1}^{N} \cdots \sum_{m_{n_{\ell k}-1}=1}^{N} a_{\ell m_1} a_{m_1 m_2} \cdots a_{m_{n_{\ell k}-1} k} \tag{6.3}$$

Proof

We already know that the sum in (6.3) should be nonzero because of the existence of the aforementioned path linking agents ℓ and k with *nonzero* scaling weights. It follows that $[A^{n_{\ell k}}]_{\ell k} > 0$. This means that the matrix A is *irreducible*; a matrix A with nonnegative entries is said to irreducible if, and only if, for every pair of indices (ℓ, k), there exists a finite integer $n_{\ell k} > 0$ such that $[A^{n_{\ell k}}]_{\ell k} > 0$; which is what we have established so far. We assume that $n_{\ell k}$ is the smallest integer that satisfies this property. Note that under irreducibility, the power $n_{\ell k}$ is allowed to be dependent on the indices (ℓ, k). Therefore, network connectivity ensures the irreducibility of A.

Proof

We now go a step further and show that *strong* network connectivity ensures the primitiveness of A. Recall from Definition 6.3 in the text that a strongly connected network is a connected network with the additional requirement that there exists at least one agent with a self-loop. We now verify that an irreducible matrix A with at least one positive diagonal element is necessarily primitive so that a common power n_o satisfies (6.1) for all (ℓ, k) (see, e.g., [168, p. 678] and [220]).

Proof

Since the network is strongly connected, this means that there exists at least one agent k_o with $a_{k_o,k_o} > 0$. We know from (6.3) that for any agent ℓ in the network, it holds that $[A^{n_{\ell k_o}}]_{\ell k_o} > 0$. Then,

$$\begin{aligned}
\left[A^{(n_{\ell k_o}+1)}\right]_{\ell k_o} &= [A^{n_{\ell k_o}} A]_{\ell k_o} \\
&= \sum_{m=1}^{N} [A^{n_{\ell k_o}}]_{\ell m} \; a_{m k_o} \\
&\geq [A^{n_{\ell k_o}}]_{\ell k_o} \; a_{k_o,k_o} \\
&> 0 \qquad (6.4)
\end{aligned}$$

Proof

so that the positivity of the $(\ell, k_o)-$th entry is maintained at higher powers of A once it is satisfied at power $n_{\ell k_o}$. The integers $\{n_{\ell k_o}\}$ are bounded by N. Let

$$m_o \stackrel{\Delta}{=} \max_{1 \le \ell \le N} \{n_{\ell k_o}\} \tag{6.5}$$

Then, the above result implies that

$$[A^{m_o}]_{\ell k_o} > 0, \quad \text{for all } \ell \tag{6.6}$$

Proof

so that the entries on the k_o-th column of A^{m_o} are all positive. Similarly, repeating the argument (6.3) we can verify that for arbitrary agents (k,ℓ), with the roles of k and ℓ now reversed, there exists a path of length $n_{k\ell}$ such that $[A^{n_{k\ell}}]_{k\ell} > 0$. For the same agent k_o with $a_{k_o,k_o} > 0$ as above, it holds that

$$\begin{aligned}
\left[A^{(n_{k_o\ell}+1)}\right]_{k_o\ell} &= [AA^{n_{k_o\ell}}]_{k_o\ell} \\
&= \sum_{m=1}^{N} a_{k_o m}\,[A^{n_{k_o\ell}}]_{m\ell} \\
&\geq a_{k_o,k_o}\,[A^{n_{k_o\ell}}]_{k_o\ell} \\
&> 0 \qquad (6.7)
\end{aligned}$$

Proof

so that the positivity of the $(k_o, \ell)-$th entry is maintained at higher powers of A once it is satisfied at power $n_{k_o\ell}$. Likewise, the integers $\{n_{k_o\ell}\}$ are bounded by N. Let

$$m'_o \;\stackrel{\Delta}{=}\; \max_{1\le\ell\le N} \{n_{k_o\ell}\} \tag{6.8}$$

Then, the above result implies that

$$\left[A^{m'_o}\right]_{k_o\ell} > 0, \quad \text{for all } \ell \tag{6.9}$$

so that the entries on the k_o-th row of $A^{m'_o}$ are all positive.

Now, let $n_o = m_o + m'_o$ and let us examine the entries of the matrix A^{n_o}.

Proof

We can write schematically

$$A^{n_o} = A^{m_o} A^{m'_o} = \begin{bmatrix} \times & \times & + & \times \\ \times & \times & + & \times \\ \times & \times & + & \times \\ \times & \times & + & \times \end{bmatrix} \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ + & + & + & + \\ \times & \times & \times & \times \end{bmatrix} \tag{6.10}$$

where the plus signs are used to refer to the *positive* entries on the k_o−th column and row of A^{m_o} and $A^{m'_o}$, respectively, and the $\times$ signs are used to refer to the remaining entries of A^{m_o} and $A^{m'_o}$, which are necessarily nonnegative. It is clear from the above equality that the resulting entries of A^{n_o} will all be positive, and we conclude that A is primitive.

□

Perron-Frobenius Theorem

One important consequence of the primitiveness of A is that a famous result in matrix theory, known as the *Perron-Frobenius* Theorem [27, 113, 189] allows us to characterize the eigen-structure of A in the following manner — see Lemma F.4 in the appendix:

(a) The matrix A has a *single* eigenvalue at one.

(b) All other eigenvalues of A are strictly inside the unit circle (and, hence, have magnitude strictly less than one). Therefore, the spectral radius of A is equal to one, $\rho(A) = 1$.

Perron-Frobenius Theorem

(c) With proper sign scaling, all entries of the right-eigenvector of A corresponding to the single eigenvalue at one are *positive*. Let p denote this right-eigenvector, with its entries $\{p_k\}$ normalized to add up to one, i.e.,

$$Ap = p, \quad \mathbb{1}^{\mathsf{T}} p = 1, \quad p_k > 0, \quad k = 1, 2, \ldots, N \tag{6.11}$$

We refer to p as the *Perron eigenvector* of A. All other eigenvectors of A associated with the other eigenvalues will have at least one negative or complex entry.

Graph Laplacian

Course EE210B
Spring Quarter 2015

Proc. IEEE, vol. 102, no. 4, pp. 460-497, April 2014.
Foundations and Trends in Machine Learning, vol. 7, no. 4-5, pp. 311-801, July 2014.

Reference

Appendix A (Graph Laplacian and Network Connectivity):

A. H. Sayed, ``Diffusion adaptation over networks," in ***Academic Press Library in Signal Processing***, vol. 3, R. Chellapa and S. Theodoridis, *editors*, pp. 323-454, Academic Press, Elsevier, 2014.

Setting

Consider a network consisting of N nodes and L edges connecting the nodes to each other. In the constructions below, we only need to consider the edges that connect distinct nodes to each other; these edges do not contain any self-loops that may exist in the graph and which connect nodes to themselves directly. In other words, when we refer to the L edges of a graph, we are excluding self-loops from this set; but we are still allowing loops of at least length 2 (i.e., loops generated by paths covering at least 2 edges).

Laplacian Matrix

The neighborhood of any node k is denoted by $\mathcal{N}_k$ and it consists of all nodes that node k can share information with; these are the nodes that are connected to k through edges, in addition to node k itself. The degree of node k, which we denote by n_k, is defined as the positive integer that is equal to the size of its neighborhood:

$$n_k \triangleq |\mathcal{N}_k| \tag{573}$$

Since $k \in \mathcal{N}_k$, we always have $n_k \geq 1$. We further associate with the network an $N \times N$ Laplacian matrix, denoted by $\mathcal{L}$. The matrix $\mathcal{L}$ is symmetric and its entries are defined as follows [64–66]:

Laplacian Matrix

$$[\mathcal{L}]_{k\ell} = \begin{cases} n_k - 1, & \text{if } k = \ell \\ -1, & \text{if } k \neq \ell \text{ and nodes } k \text{ and } \ell \text{ are neighbors} \\ 0, & \text{otherwise} \end{cases} \tag{574}$$

Note that the term $n_k - 1$ measures the number of edges that are incident on node k, and the locations of the $-1's$ on row k indicate the nodes that are connected to node k. We also associate with the graph an $N \times L$ incidence matrix, denoted by $\mathcal{I}$. The entries of $\mathcal{I}$ are defined as follows. Every column of $\mathcal{I}$ represents one edge in the graph. Each edge connects two nodes and its column will display two nonzero entries at the rows corresponding to these

Incidence Matrix

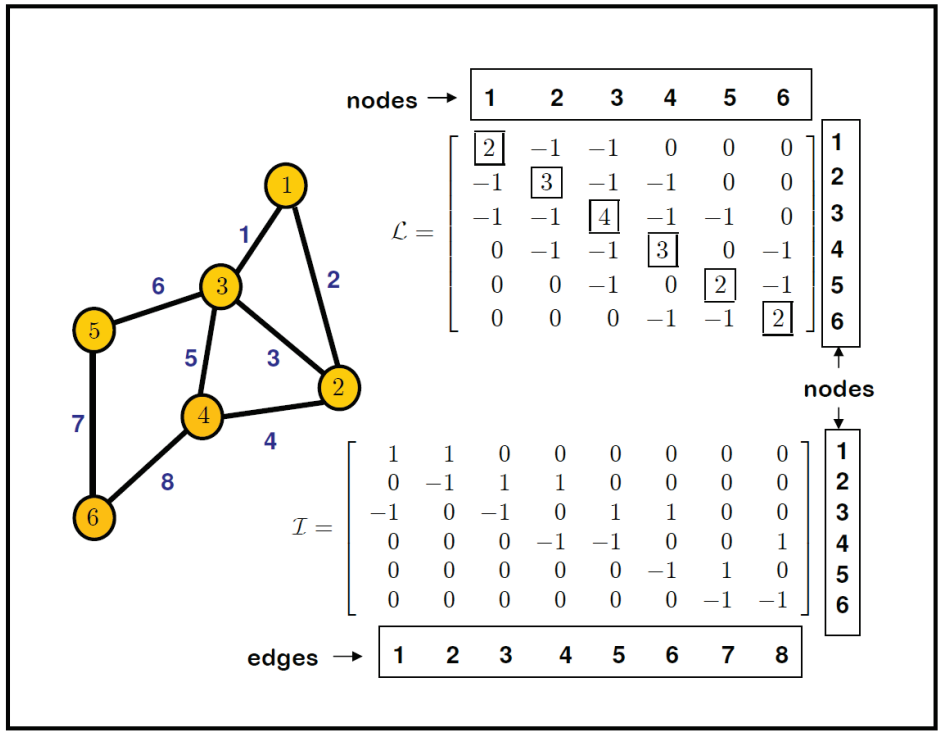
nodes: one entry will be $+1$ and the other entry will be -1. For directed graphs, the choice of which entry is positive or negative can be used to identify the nodes from which edges emanate (source nodes) and the nodes at which edges arrive (sink nodes). Since we are dealing with undirected graphs, we shall simply assign positive values to lower indexed nodes and negative values to higher indexed nodes:

$$[\mathcal{I}]_{ke} = \begin{cases} +1, & \text{if node } k \text{ is the lower-indexed node connected to edge } e \\ -1, & \text{if node } k \text{ is the higher-indexed node connected to edge } e \\ 0, & \text{otherwise} \end{cases} \tag{575}$$

Example

The figure shows the example of a network with $N = 6$ nodes and $L = 8$ edges. Its Laplacian and incidence matrices are also shown and these have sizes 6×6 and 6×8, respectively. Consider, for example, column 6 in the incidence matrix. This column corresponds to edge 6, which links nodes 3 and 5. Therefore, at location $\mathcal{I}_{36}$ we have a $+1$ and at location $\mathcal{I}_{56}$ we have -1. The other columns of $\mathcal{I}$ are constructed in a similar manner.

Example


nodes → 1 2 3 4 5 6
$\mathcal{L} =$
2 −1 −1 0 0 0
−1 3 −1 −1 0 0
−1 −1 4 −1 −1 0
0 −1 −1 3 0 −1
0 0 −1 0 2 −1
0 0 0 −1 −1 2
1 2 3 4 5 6
nodes
nodes
$\mathcal{I} =$
1 1 0 0 0 0 0 0
0 −1 1 1 0 0 0 0
−1 0 −1 0 1 1 0 0
0 0 0 −1 −1 0 0 1
0 0 0 0 0 −1 1 0
0 0 0 0 0 0 −1 −1
1 2 3 4 5 6
edges → 1 2 3 4 5 6 7 8
1
2
3
4
5
6
1
2
3
4
5
6
7
8

Laplacian & Incidence Matrices

Observe that the Laplacian and incidence matrices of a graph are related as follows:

$$\mathcal{L} = \mathcal{I}\,\mathcal{I}^T \tag{576}$$

The Laplacian matrix conveys useful information about the topology of the graph. The following is a classical result from graph theory [64–67].

Network Connectivity

***Lemma* B.1. (Laplacian and Network Connectivity)** *Let*

$$\theta_1 \geq \theta_2 \geq \ldots \geq \theta_N \tag{577}$$

denote the ordered eigenvalues of $\mathcal{L}$*. Then the following properties hold:*

(a) $\mathcal{L}$ *is symmetric nonnegative-definite so that* $\theta_i \geq 0$*.*

(b) The rows of $\mathcal{L}$ *add up to zero so that* $\mathcal{L}\mathbb{1} = 0$*. This means that* $\mathbb{1}$ *is a right eigenvector of* $\mathcal{L}$ *corresponding to the eigenvalue zero.*

Network Connectivity

(c) *The smallest eigenvalue is always zero,* $\theta_N = 0$. *The second smallest eigenvalue,* θ_{N-1}, *is called the algebraic connectivity of the graph.*

(d) *The number of times that zero is an eigenvalue of* $\mathcal{L}$ *(i.e., its multiplicity) is equal to the number of* **separated yet connected subgraphs**.

(e) *The algebraic connectivity of a connected graph is nonzero, i.e.,* $\theta_{N-1} \neq 0$. *In other words, a graph is connected if, and only if, its algebraic connectivity is nonzero.*

Proof

Proof. Property (a) follows from the identity $\mathcal{L} = \mathcal{I}\,\mathcal{I}^T$. Property (b) follows from the definition of $\mathcal{L}$. Note that for each row of $\mathcal{L}$, the entries on the row add up to zero. Property (c) follows from properties (a) and (b) since $\mathcal{L}\mathbb{1} = 0$ implies that zero is an eigenvalue of $\mathcal{L}$. For part (d), assume the network consists of two separate connected subgraphs. Then, the Laplacian matrix would have a block diagonal structure, say, of the form $\mathcal{L} = \text{diag}\{\mathcal{L}_1, \mathcal{L}_2\}$, where $\mathcal{L}_1$ and $\mathcal{L}_2$ are the Laplacian matrices of the smaller subgraphs. The smallest eigenvalue of each of these Laplacian matrices would in turn be zero and unique by property (e). More generally, if the graph consists of m connected subgraphs, then the multiplicity of zero as an eigenvalue of $\mathcal{L}$ must be m. To establish property (e), first observe that if the algebraic connectivity is nonzero then it is obvious that the graph must be connected. Otherwise, if the graph were disconnected, then its Laplacian matrix would be block diagonal and the algebraic multiplicity of zero as an eigenvalue of $\mathcal{L}$ would be larger than one so that θ_{N-1} would be zero, which is a contradiction. For the converse statement,

Proof

assume the graph is connected and let x denote an arbitrary eigenvector of $\mathcal{L}$ corresponding to the eigenvalue at zero, i.e., $\mathcal{L}x = 0$. We already know that $\mathcal{L}\mathbb{1} = 0$ from property (b). Let us verify that x must be proportional to the vector $\mathbb{1}$ so that the algebraic multiplicity of the eigenvalue at zero is one. Thus note that $x^T\mathcal{L}x = 0$. If we denote the individual entries of x by x_k, then this identity implies that for each node k:

$$\sum_{\ell\in\mathcal{N}_k}(x_k - x_\ell)^2 \;=\; 0$$

It follows that the entries of x within each neighborhood have equal values. But since the graph is connected, we conclude that all entries of x must be equal. It follows that the eigenvector x is proportional to the vector $\mathbb{1}$, as desired. □

Network Objective

Course EE210B
Spring Quarter 2015

Proc. IEEE, vol. 102, no. 4, pp. 460-497, April 2014.
Foundations and Trends in Machine Learning, vol. 7, no. 4-5, pp. 311-801, July 2014.

Network Objective

In the remaining lectures we will be interested in showing how network cooperation can be exploited to solve a variety of problems in an advantageous manner. We are particularly interested in formulations that can solve adaptation, learning, and optimization problems in a decentralized and online manner in response to *streaming* data. It turns out that useful commonalities run across these three domain problems. For this reason, we shall keep the development general enough and then show, by means of examples, how the results can be used to handle many situations of interest as special cases.

Network Objective

Thus, consider a connected network consisting of a total of N agents, labeled $k = 1, 2, \ldots, N$. We associate with each agent a twice-differentiable individual *cost* function, denoted by $J_k(w) \in \mathbb{R}$. This function is sometimes called the *utility* function in applications involving resource management issues and the *risk* function in machine learning applications; it may be called by other names in other domains. We adopt the generic terminology of a "cost" function. The function $J_k(w) \in \mathbb{R}$ is itself *real-valued*. However, for generality, its argument $w \in \mathbb{C}^M$ is assumed to be possibly *complex-valued*, say, of size $M \times 1$. This set-up is illustrated in Figure 6.4 where we are now representing the bi-directional edges between agents by single segment lines for ease of representation.

Network Objective

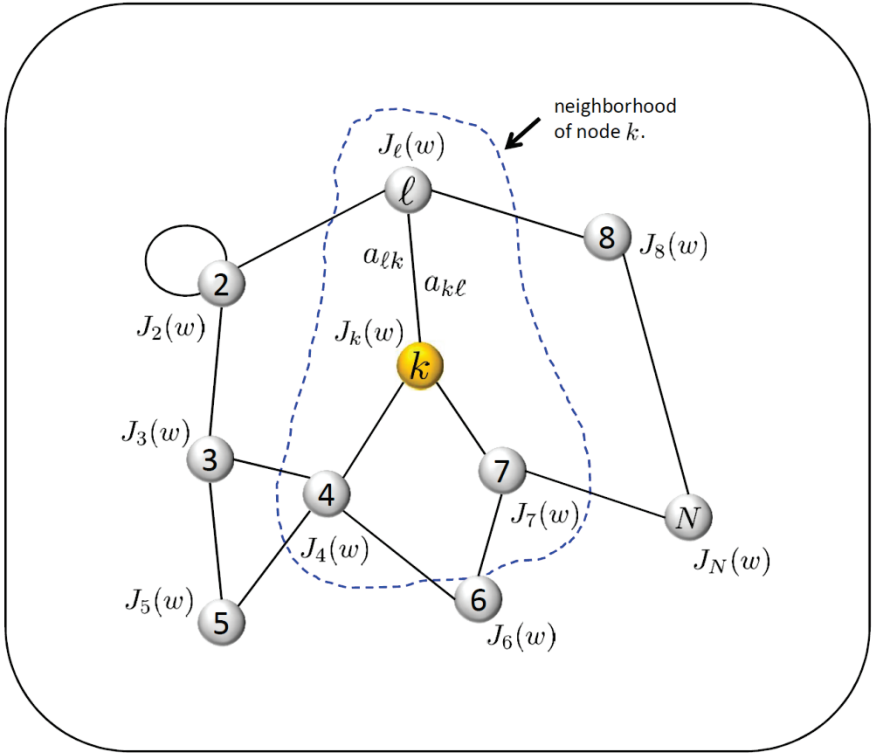


Figure 6.4: A cost function $J_k(w)$ is associated with each individual agent k in the network. The bi-directional edges between agents are being represented by single segment lines for ease of representation. Information can flow both ways over these edges with scalings $\{a_{k\ell}, a_{\ell k}\}$.

Aggregate Cost

The objective of the network of agents is still to seek the unique minimizer of the aggregate cost function, $J^{\text{glob}}(w)$, defined earlier by (5.19) and which we repeat below

$$J^{\text{glob}}(w) \triangleq \sum_{k=1}^{N} J_k(w) \tag{6.12}$$

Assumptions

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 $J^{\text{glob}}(w)$ satisfies the conditions of Assumption 5.1 with parameters $\{\nu_d, \delta_d, \kappa_d\}$, with the subscript "$d$" now used to indicate that these parameters are related to the distributed implementation.

Assumptions

Assumption 6.1 (Conditions on aggregate and individual costs). It is assumed that the individual cost functions, $J_k(w)$, are each twice-differentiable and convex, with at least one of them being ν_d-strongly convex. Moreover, the aggregate cost function, $J^{\text{glob}}(w)$, is also twice-differentiable and satisfies

$$0 < \frac{\nu_d}{h} I_{hM} \;\leq\; \nabla_w^2 \, J^{\text{glob}}(w) \;\leq\; \frac{\delta_d}{h} I_{hM} \tag{6.13}$$

for some positive parameters $\nu_d \leq \delta_d$.

Assumptions

Under these conditions, the cost $J^{\text{glob}}(w)$ will have a unique minimizer, which we continue to denote by w^o. Note that we are not requiring the individual costs $J_k(w)$ to be strongly convex. As mentioned earlier, it is sufficient to assume that at least one of these costs is ν_d−strongly convex while the remaining costs are simply convex; this condition ensures that $J^{\text{glob}}(w)$ will be strongly convex.

Assumptions

The individual costs $\{J_k(w)\}$ 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 (6.12) would correspond to the case in which the agents work together to optimize the same cost function. Moreover, when they exist, the minimizers of the individual costs, $\{J_k(w)\}$, need not coincide with each other or with w^o; we shall write w_k^o to refer to a minimizer of $J_k(w)$. There are important

Assumptions

situations in practice where all minimizers $\{w_k^o\}$ happen to 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., [56, 208, 214, 246]). This scenario is also common in machine learning problems [4, 37, 85, 192, 233, 239] 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 #6.1

Example 6.1 (Common minimizer). Consider the same setting of Example 3.4 except that we now have N agents observing streaming data $\{\boldsymbol{d}_k(i), \boldsymbol{u}_{k,i}\}$ that satisfy the regression model (3.119) with regression covariance matrices $R_{u,k} = \mathbb{E}\,\boldsymbol{u}_{k,i}^*\boldsymbol{u}_{k,i} > 0$ and with the same unknown w^o, i.e.,

$$\boldsymbol{d}_k(i) \;=\; \boldsymbol{u}_{k,i}w^o + \boldsymbol{v}_k(i) \tag{6.14}$$

where the noise process, $\boldsymbol{v}_k(i)$, is independent of the regression data, $\boldsymbol{u}_{k,i}$. The individual mean-square-error costs are defined by

$$J_k(w) = \mathbb{E}\,|\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}w|^2 \tag{6.15}$$

Example #6.1

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

$$w_k^o = R_{u,k}^{-1} r_{du,k}, \quad k = 1, 2, \ldots, N \tag{6.16}$$

If we multiply both sides of (6.14) by $\boldsymbol{u}_{k,i}^*$ from the left, and take expectations, we find that w^o satisfies

$$r_{du,k} = R_{u,k} w^o \tag{6.17}$$

This relation shows that the unknown w^o from (6.14) satisfies the same expression as w_k^o in (6.16), for any $k = 1, 2, \ldots, N$, so that we must have

$$w^o = w_k^o, \qquad k = 1, 2, \ldots, N \tag{6.18}$$

Example #6.1

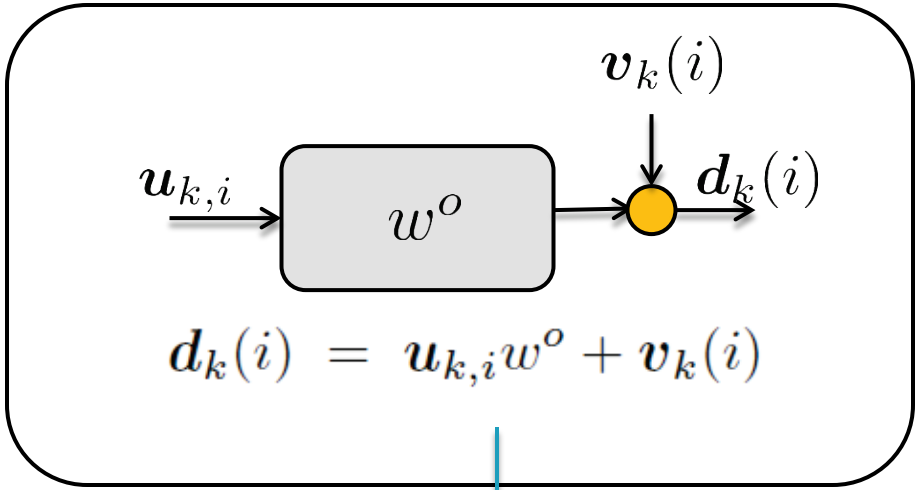
Therefore, this example amounts to a situation where all costs $\{J_k(w)\}$ attain their minima at the same location, w^o, even though the moments $\{r_{du,k}, R_{u,k}\}$ and, therefore, the individual costs $\{J_k(w)\}$, may be different from each other. This example highlights one convenience of working with mean-square-error (MSE) costs: under linear regression models of the form (6.14), the MSE formulation (6.15) allows each agent to recover w^o *exactly*.

■

Example #6.1

$$J_k(w) = \mathbb{E}\,|\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}w|^2$$

$$\longrightarrow \quad w_k^o \;=\; R_{u,k}^{-1} r_{du,k}$$



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

$$w^o = w_k^o, \qquad k = 1, 2, \ldots, N$$

Example #6.2

Example 6.2 (Linear regression models). Linear data models of the form (6.14) are common in practice. We provide two examples from [208]. Consider first a situation in which agents are spread over a geographical region and observe realizations of an auto-regressive (AR) random process $\{\boldsymbol{d}_k(i)\}$ of order M. The AR process observed by agent k satisfies the model:

$$\boldsymbol{d}_k(i) = \sum_{m=1}^{M} \beta_m \boldsymbol{d}_k(i-m) + \boldsymbol{v}_k(i), \quad k = 1, 2, \ldots, N \tag{6.19}$$

where i is the time index, the scalars $\{\beta_m\}$ represent the model parameters that the agents wish to identify, and $\boldsymbol{v}_k(i)$ represents the additive noise process. If we collect the $\{\beta_m\}$ into an $M \times 1$ column vector:

Example #6.2

$$w^o \triangleq \operatorname{col}\{\beta_1, \beta_2, \ldots, \beta_M\} \tag{6.20}$$

and the past data into a $1 \times M$ regression vector:

$$\boldsymbol{u}_{k,i} \triangleq \left[\begin{array}{cccc} \boldsymbol{d}_k(i-1) & \boldsymbol{d}_k(i-2) & \ldots & \boldsymbol{d}_k(i-M) \end{array}\right] \tag{6.21}$$

then we can rewrite the measurement equation (6.19) in the form (6.14) for each time instant i.

Example #6.2

Consider a second example where the agents are now interested in estimating the taps of a communications channel or the parameters of some physical model of interest. Assume the agents are able to independently probe the unknown model and observe its response to excitations in the presence of additive noise. Each agent k probes the model with an input sequence $\{\boldsymbol{u}_k(i)\}$ and measures the response sequence, $\{\boldsymbol{d}_k(i)\}$, in the presence of additive noise. The system dynamics for each agent k is assumed to be described by a moving-average (MA) model of the form:

Example #6.2

$$\boldsymbol{d}_k(i) = \sum_{m=0}^{M-1} \beta_m \boldsymbol{u}_k(i-m) + \boldsymbol{v}_k(i) \tag{6.22}$$

If we again collect the parameters $\{\beta_m\}$ into an $M \times 1$ column vector w^o, and the input data into a $1 \times M$ regression vector:

$$\boldsymbol{u}_{k,i} = \left[\begin{array}{cccc} \boldsymbol{u}_k(i) & \boldsymbol{u}_k(i-1) & \ldots & \boldsymbol{u}_k(i-M+1) \end{array} \right] \tag{6.23}$$

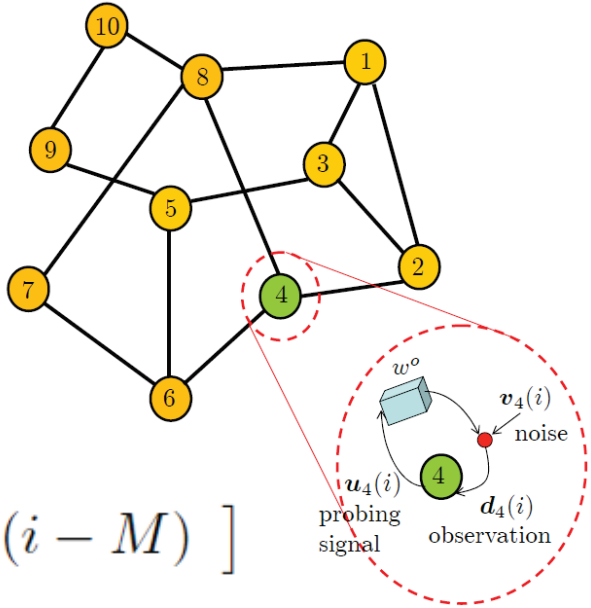
then we arrive again at the same linear model (6.14).

■

Example #6.2

AR (auto-regressive) model:

$$\begin{cases} \boldsymbol{d}_k(i) = \displaystyle\sum_{m=1}^{M} \beta_m \boldsymbol{d}_k(i-m) + \boldsymbol{v}_k(i) \\ w^o \triangleq \operatorname{col}\{\beta_1, \beta_2, \ldots, \beta_M\} \\ \boldsymbol{u}_{k,i} \triangleq \begin{bmatrix} \boldsymbol{d}_k(i-1) & \boldsymbol{d}_k(i-2) & \ldots & \boldsymbol{d}_k(i-M) \end{bmatrix} \\ \boldsymbol{d}_k(i) = \boldsymbol{u}_{k,i} w^o + \boldsymbol{v}_k(i) \end{cases}$$

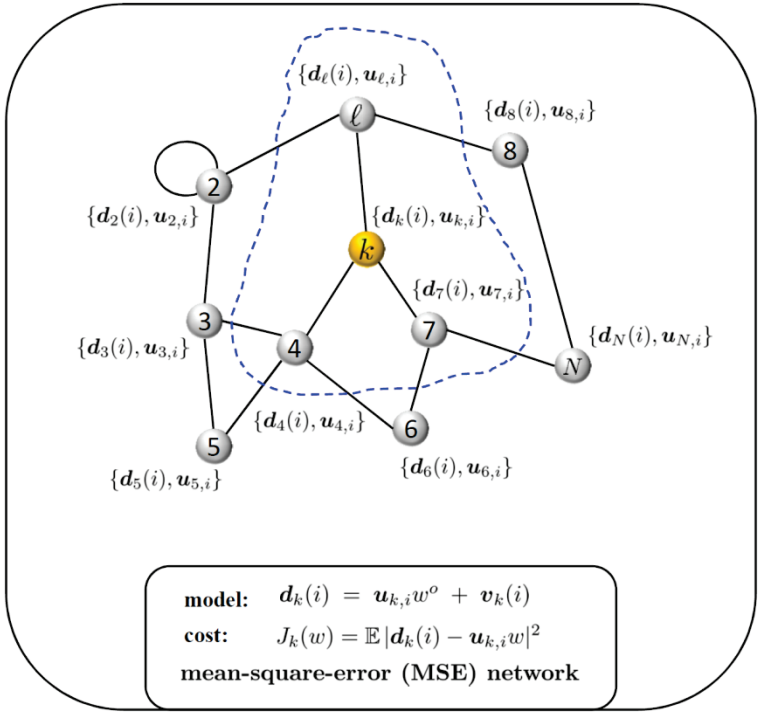


Example #6.3 (MSE Networks)

Example 6.3 (Mean-square-error (MSE) networks). The data model introduced in Example 6.1 will be called upon frequently in our presentation to illustrate various concepts and results. We shall refer to strongly-connected networks with agents receiving data according to model (6.14) and seeking to estimate w^o by adopting the mean-square-error costs $J_k(w)$ defined by (6.15), as mean-square-error (MSE) networks.

We find it useful to collect in this example the details of the model for ease of reference whenever necessary. Thus, refer to Figure 6.5. The plot shows a

Example #6.3 (MSE Networks)



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

$$J_k(w) = \mathbb{E}\,|\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}w|^2$$

Figure 6.5: Illustration of mean-square-error (MSE) networks. The plot shows a strongly-connected network where each agent is subjected to streaming data $\{\boldsymbol{d}_k(i), \boldsymbol{u}_{k,i}\}$ that satisfy the linear regression model (6.24). The cost associated with each agent is the mean-square-error cost defined by (6.25).

Example #6.3 (MSE Networks)

strongly-connected network where each agent is subjected to streaming data $\{\boldsymbol{d}_k(i), \boldsymbol{u}_{k,i}\}$ that are assumed to satisfy the linear regression model:

$$\boldsymbol{d}_k(i) \;=\; \boldsymbol{u}_{k,i} w^o \;+\; \boldsymbol{v}_k(i), \;\; i \geq 0, \qquad k = 1, 2, \ldots, N \tag{6.24}$$

for some unknown $M \times 1$ vector w^o. A mean-square-error cost is associated with each agent k, namely,

$$J_k(w) \;=\; \mathbb{E}\,|\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i} w|^2, \qquad k = 1, 2, \ldots, N \tag{6.25}$$

Example #6.3 (MSE Networks)

The processes $\{\boldsymbol{d}_k(i), \boldsymbol{u}_{k,i}, \boldsymbol{v}_k(i)\}$ that appear in (6.24) are assumed to represent zero-mean jointly wide-sense stationary random processes that satisfy the following three conditions (these conditions help facilitate the analysis of such networks):

Example #6.3 (MSE Networks)

(a) The regression data $\{\boldsymbol{u}_{k,i}\}$ are temporally white and independent over space with

$$\mathbb{E}\,\boldsymbol{u}_{k,i}^{*}\boldsymbol{u}_{\ell,j} \;\triangleq\; R_{u,k}\,\delta_{k,\ell}\,\delta_{i,j} \tag{6.26}$$

where $R_{u,k} > 0$ and the symbol $\delta_{m,n}$ denotes the Kronecker delta sequence: its value is equal to one when $m = n$ and its value is equal to zero otherwise.

(b) The noise process $\{\boldsymbol{v}_k(i)\}$ is temporally white and independent over space with variance

$$\mathbb{E}\,\boldsymbol{v}_k(i)\boldsymbol{v}_\ell^{*}(j) \;\triangleq\; \sigma_{v,k}^2\,\delta_{k,\ell}\,\delta_{i,j} \tag{6.27}$$

(c) The regression and noise processes $\{\boldsymbol{u}_{\ell,j}, \boldsymbol{v}_k(i)\}$ are independent of each other for all k, ℓ, i, j.

■

Why Cooperate?

One natural question that arises in the case of a common minimizer is to inquire why agents should cooperate to determine w^o when each one of them is capable of determining w^o on its own through (6.16)? There are at least two good reasons to justify cooperation even in this case. First, agents will rarely have access to the full information they need to determine w^o independently. For example, in many situations, agents may not know fully their own costs $J_k(w)$. For instance, agents may not know beforehand the statistical moments $\{r_{du,k}, R_{u,k}\}$ of the data that they are sensing; this is the situation we encountered earlier in Examples 3.1 and 3.4 when we developed recursive adaptation schemes to address this lack of information.

Why Cooperate?

Agents would need to replace the unavailable moments $\{r_{du,k}, R_{u,k}\}$ by some approximations before attempting (6.16). Moreover, different agents will generally be subject to different noise conditions and the quality of their moment approximations will therefore vary. In that case, their estimates for w^o will be as good as the quality of their data, as we already remarked earlier following result (5.10). Through cooperation with each other, not only agents with "bad" noise conditions will benefit, but also agents with "good" noise conditions can benefit and improve the accuracy of their estimation (see, e.g., Chapter 12 and also [208, 214]).

Why Cooperate?

A second reason to motivate cooperation among the agents is that even when they know the moments $\{r_{du,k}, R_{u,k}\}$, the individual costs need not be strongly convex and the agents may not be able to recover w^o on their own due to ambiguities or ill-conditioning. For example, if some of the covariance matrices $\{R_{u,k}\}$ in Example 6.1 are singular, then the corresponding cost functions $\{J_k(w)\}$ will not be strongly convex and the individual agents will not be able to determine w^o uniquely. In that case, cooperation among agents would help them resolve the ambiguity about w^o.

Key Points

Proc. IEEE, vol. 102, no. 4, pp. 460-497, April 2014.
Foundations and Trends in Machine Learning, vol. 7, no. 4-5, pp. 311-801, July 2014.

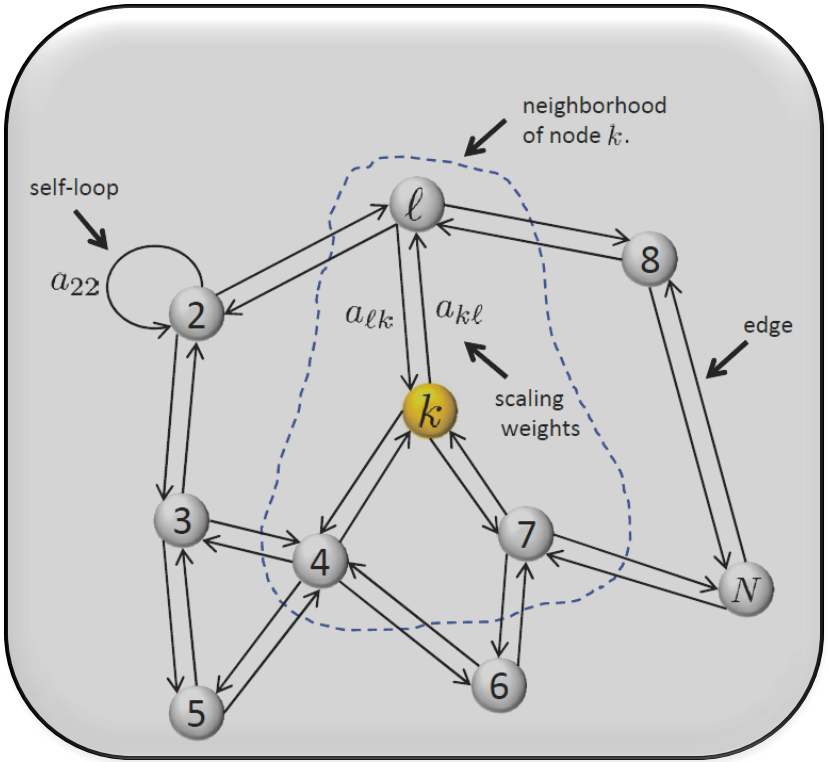
Multi-Agent Network

Network with *N* vertices (agents):

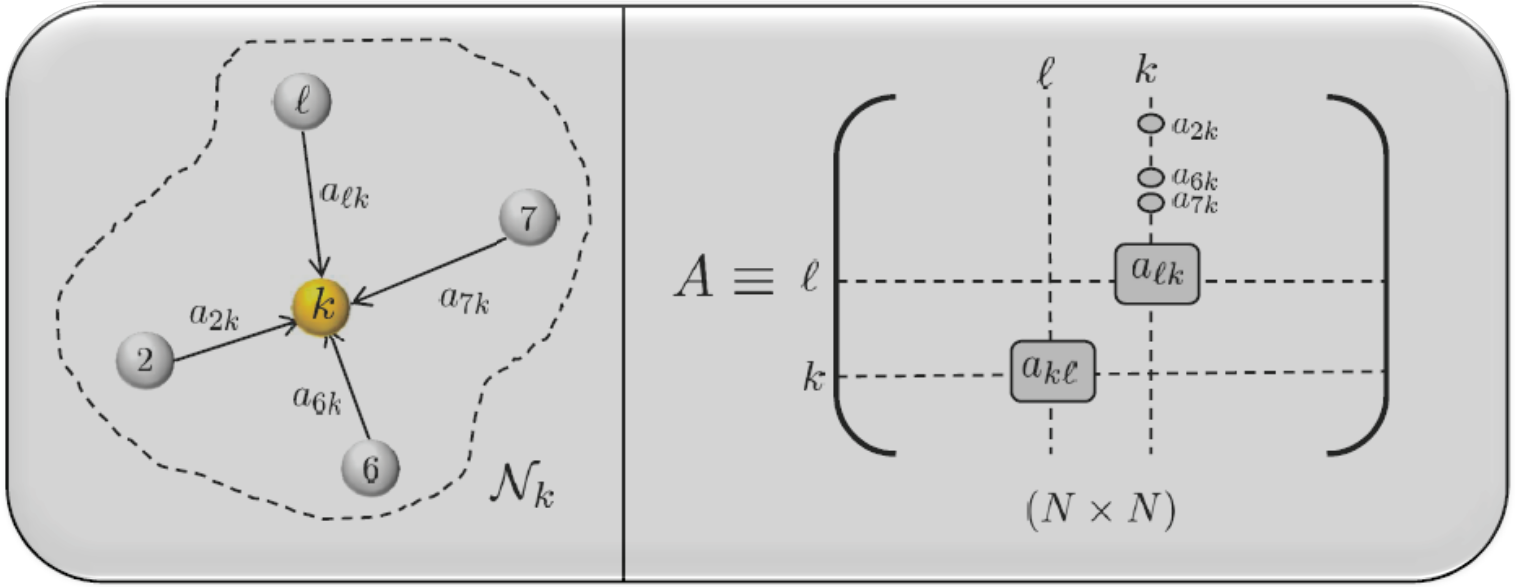
- Edges connecting agents.
- $a_{\ell k}$: scales data from ℓ to k.
- $\mathcal{N}_k$: neighbors of agent k.

Strongly-connected network:

path with nonzero weights between any two agents plus at least one self-loop $(a_{kk} > 0)$.



Combination Matrix



$$a_{\ell k} \geq 0, \qquad \sum_{\ell \in \mathcal{N}_k} a_{\ell k} = 1, \qquad A^{\mathsf{T}} \mathbb{1} = \mathbb{1}$$ **(left-stochastic)**

Primitive Matrix

Combination matrix A is said to be **primitive** if there exists an integer $n_o > 0$:

$$[A^{n_o}]_{\ell k} > 0$$

Lemma 6.1: The combination matrix of a strongly-connected network is primitive.

Perron-Frobenius Theorem

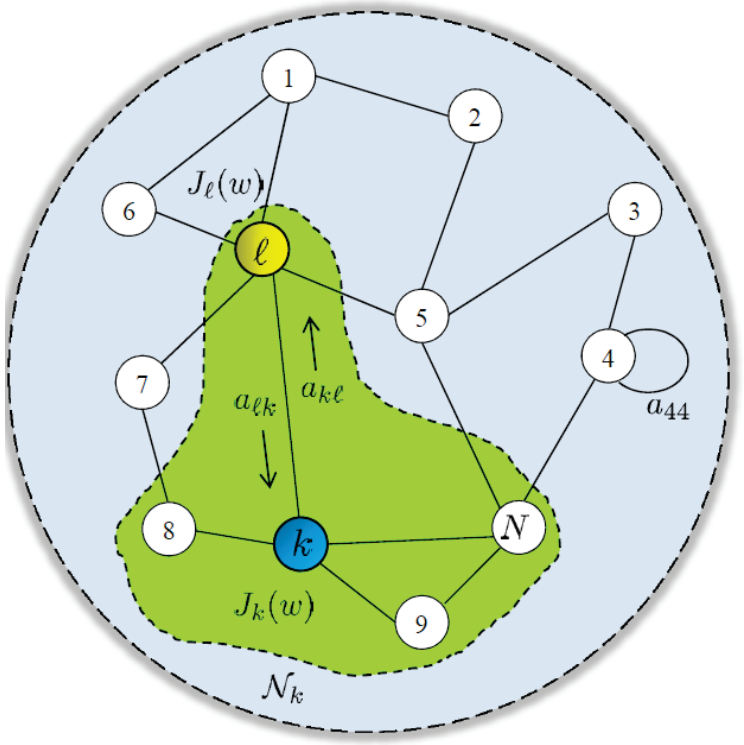
- A has a single eigenvalue at one.
- All other eigenvalues are strictly inside the unit circle.
- **Perron vector:**

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

Aggregate Network Cost

$$J^{\text{glob}}(w) \triangleq \sum_{k=1}^{N} J_k(w)$$

At least one individual cost is strongly-convex.



Why Cooperate?

At least two formidable reasons:

- Agents observe data of different quality.

- Some agents may not have sufficient information to recover the unknown on their own:

  Singular Hessian matrices → ambiguity in data.

End of Lecture

Proc. IEEE, vol. 102, no. 4, pp. 460-497, April 2014.
Foundations and Trends in Machine Learning, vol. 7, no. 4-5, pp. 311-801, July 2014.