Dynamic resource allocation problems in communication networks:

Stochastic approximation, convex optimisation for resource allocation

Alexandre Reiffers-Masson

Equipe Maths&Net, IMT Atlantique, CS department
LabSTICC (UMR CNRS 6285)

July 13, 2023

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Quick reminder in Optimisation, and Differential equations

Problem I: Experimental design

Problem II: Backpropagation for Distributed Resource Allocation

Gradient of vector valued function

- Let $f : \mathbb{R}^n \to \mathbb{R}$. We define the one-sided directional derivative of f in the direction y, to be equal to (provided that the limit exists):

$$f'(x; y) = \lim_{\alpha \downarrow 0} \frac{f(x + \alpha y) - f(x)}{\alpha}.$$

Gradient of vector valued function

- Let $f : \mathbb{R}^n \to \mathbb{R}$. We define the one-sided directional derivative of f in the direction y, to be equal to (provided that the limit exists):

$$f'(x; y) = \lim_{\alpha \downarrow 0} \frac{f(x + \alpha y) - f(x)}{\alpha}.$$

- If all directional derivatives of f at a point x exist and $f'(x; y)$ is a linear function of y, we say that f is differentiable at x.

Gradient of vector valued function

- Let $f : \mathbb{R}^n \to \mathbb{R}$. We define the one-sided directional derivative of f in the direction y, to be equal to (provided that the limit exists):

$$f'(x; y) = \lim_{\alpha \downarrow 0} \frac{f(x + \alpha y) - f(x)}{\alpha}.$$

- If all directional derivatives of f at a point x exist and $f'(x; y)$ is a linear function of y, we say that f is differentiable at x.
- If f is differentiable at x then $f'(x; y) = y^T \nabla f(x)$ for all x.

Jacobian and the chain rule

- A vector valued function $f : \mathbb{R}^n \to \mathbb{R}^m$ is a differentiable vector valued function if each component f_i of f is differentiable.

Jacobian and the chain rule

- A vector valued function $f : \mathbb{R}^n \to \mathbb{R}^m$ is a differentiable vector valued function if each component f_i of f is differentiable.
- Let $\nabla f(x)$ be the matrix of dimensions $n \times m$ whose i-th column is the gradient $\nabla f_i(x)$ of f_i. Therefore:

$$\nabla f(x) = [\nabla f_1(x) \ldots \nabla f_m(x)]$$

Jacobian and the chain rule

- A vector valued function $f : \mathbb{R}^n \to \mathbb{R}^m$ is a differentiable vector valued function if each component f_i of f is differentiable.
- Let $\nabla f(x)$ be the matrix of dimensions $n \times m$ whose i-th column is the gradient $\nabla f_i(x)$ of f_i. Therefore:

$$\nabla f(x) = [\nabla f_1(x) \ldots \nabla f_m(x)]$$

- The transpose of ∇f is called the Jacobian.

Jacobian and the chain rule

- A vector valued function $f : \mathbb{R}^n \to \mathbb{R}^m$ is a differentiable vector valued function if each component f_i of f is differentiable.
- Let $\nabla f(x)$ be the matrix of dimensions $n \times m$ whose i-th column is the gradient $\nabla f_i(x)$ of f_i. Therefore:

$$\nabla f(x) = [\nabla f_1(x) \ldots \nabla f_m(x)]$$

- The transpose of ∇f is called the Jacobian.
- The chain rule differentiation: Let $f : \mathbb{R}^k \to \mathbb{R}^m$ and $g : \mathbb{R}^m \to \mathbb{R}^n$ be continuously differentiable functions and let $h(x) = g(f(x))$. Then the chain rule for differentiation states that:

Jacobian and the chain rule

- A vector valued function $f : \mathbb{R}^n \to \mathbb{R}^m$ is a differentiable vector valued function if each component f_i of f is differentiable.
- Let $\nabla f(x)$ be the matrix of dimensions $n \times m$ whose i-th column is the gradient $\nabla f_i(x)$ of f_i. Therefore:

$$\nabla f(x) = [\nabla f_1(x) \ldots \nabla f_m(x)]$$

- The transpose of ∇f is called the Jacobian.
- The chain rule differentiation: Let $f : \mathbb{R}^k \to \mathbb{R}^m$ and $g : \mathbb{R}^m \to \mathbb{R}^n$ be continuously differentiable functions and let $h(x) = g(f(x))$. Then the chain rule for differentiation states that:

$$\nabla h(x) = \nabla f(x) \nabla g(f(x)). \tag{1}$$

Second order Taylor Series

Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable.

Second order Taylor Series

Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable.

- For any $x \in \mathbb{R}^n$, there exists a function $h : \mathbb{R}^n \to \mathbb{R}$ satisfying $\lim_{\|y\| \to 0} \frac{h(y)}{\|y\|^2} = 0$, and such that

Second order Taylor Series

Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable.

- For any $x \in \mathbb{R}^n$, there exists a function $h : \mathbb{R}^n \to \mathbb{R}$ satisfying $\lim_{\|y\| \to 0} \frac{h(y)}{\|y\|^2} = 0$, and such that

$$f(x+y) = f(x) + y^T \nabla f(x) + \frac{1}{2} y^T \nabla^2 f(x) y + h(y), \ \forall y \in \mathbb{R}^n,$$

Second order Taylor Series

Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable.

- For any $x \in \mathbb{R}^n$, there exists a function $h : \mathbb{R}^n \to \mathbb{R}$ satisfying $\lim_{\|y\| \to 0} \frac{h(y)}{\|y\|^2} = 0$, and such that

$$f(x+y) = f(x) + y^T \nabla f(x) + \frac{1}{2} y^T \nabla^2 f(x) y + h(y), \ \forall y \in \mathbb{R}^n,$$

- For any $x, y \in \mathbb{R}^n$, there exists some $\alpha \in [0,1]$ such that:

Second order Taylor Series

Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable.

- For any $x \in \mathbb{R}^n$, there exists a function $h : \mathbb{R}^n \to \mathbb{R}$ satisfying $\lim_{\|y\|\to 0} \frac{h(y)}{\|y\|^2} = 0$, and such that

$$f(x+y) = f(x) + y^T \nabla f(x) + \frac{1}{2} y^T \nabla^2 f(x) y + h(y), \ \forall y \in \mathbb{R}^n,$$

- For any $x, y \in \mathbb{R}^n$, there exists some $\alpha \in [0,1]$ such that:

$$f(x+y) = f(x) + y^T \nabla f(x) + \frac{1}{2} y^T \nabla^2 f(x+\alpha y) y. \qquad (2)$$

Descent Lemma

Proposition: If $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and has the property that $\|\nabla f(x) - \nabla f(y)\|_2 \leq K\|x - y\|_2$ for every $x, y \in \mathbb{R}^n$, then:

Descent Lemma

Proposition: If $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and has the property that $\|\nabla f(x) - \nabla f(y)\|_2 \leq K\|x - y\|_2$ for every $x, y \in \mathbb{R}^n$, then:

$$f(x + y) \leq f(x) + y'\nabla f(x) + \frac{K}{2}\|y\|_2^2.$$

Question: how will you optimise the function f?

Unconstrained Optimisation

We consider algorithms for minimizing a continuously differentiable cost function $f : \mathbb{R}^n \to \mathbb{R}$.

Unconstrained Optimisation

We consider algorithms for minimizing a continuously differentiable cost function $f : \mathbb{R}^n \to \mathbb{R}$.

- Fact: $\nabla f(x^*) = 0$ for every vector x^* that minimizes f.

Unconstrained Optimisation

We consider algorithms for minimizing a continuously differentiable cost function $f : \mathbb{R}^n \to \mathbb{R}$.

- Fact: $\nabla f(x^*) = 0$ for every vector x^* that minimizes f.
- Most algorithms in optimisation are aimed to find a solution of the equation $\nabla f(x) = 0$ without any guarantees that such a solution is a global minimizer of f.

Unconstrained Optimisation

We consider algorithms for minimizing a continuously differentiable cost function $f : \mathbb{R}^n \to \mathbb{R}$.

- Fact: $\nabla f(x^*) = 0$ for every vector x^* that minimizes f.
- Most algorithms in optimisation are aimed to find a solution of the equation $\nabla f(x) = 0$ without any guarantees that such a solution is a global minimizer of f.
- Warning: Existence of a solution of $\nabla f(x) = 0$ is not trivial.

Intuition of Descent Algorithm

- Let $z = x + \gamma y$ such that $y^T \nabla f(x) \leq -K_2 \|y\|_2^2$.

Intuition of Descent Algorithm

- Let $z = x + \gamma y$ such that $y^T \nabla f(x) \leq -K_2 \|y\|_2^2$.
- Let us define $z = x + \gamma y$. Let us also assume that $z^T \nabla F(x) < 0$. Then from the descent lemma we have:

Intuition of Descent Algorithm

- Let $z = x + \gamma y$ such that $y^T \nabla f(x) \leq -K_2 \|y\|_2^2$.
- Let us define $z = x + \gamma y$. Let us also assume that $z^T \nabla F(x) < 0$. Then from the descent lemma we have:

$$
\begin{aligned}
f(z) &= f(x + \gamma y) \\
&\leq f(x) + \gamma y^T \nabla F(x) + \gamma^2 \frac{K}{2} \|y\|_2^2 \\
&\leq f(x) - \gamma \left(K_2 - \frac{K\gamma}{2} \right) \|y\|_2^2
\end{aligned}
$$

Intuition of Descent Algorithm

- Let $z = x + \gamma y$ such that $y^T \nabla f(x) \leq -K_2 \|y\|_2^2$.
- Let us define $z = x + \gamma y$. Let us also assume that $z^T \nabla F(x) < 0$. Then from the descent lemma we have:

$$\begin{aligned} f(z) &= f(x + \gamma y) \\ &\leq f(x) + \gamma y^T \nabla F(x) + \gamma^2 \frac{K}{2} \|y\|_2^2 \\ &\leq f(x) - \gamma \left(K_2 - \frac{K\gamma}{2} \right) \|y\|_2^2 \end{aligned}$$

- Note that of $\gamma \in (0, 2K_2/K)$ then $f(z) \leq f(x)$.

Algorithms

Gradient Algorithm:

$$x_{n+1} = x_n - \gamma \nabla f(x_n), \tag{3}$$

Newton's Algorithm:

$$x_{n+1} = x_n - \gamma (\nabla^2 f(x_n))^{-1} \nabla f(x_n), \tag{4}$$

Scaled Gradient Algorithm:

$$x_{n+1} = x_n - \gamma (D_n)^{-1} \nabla f(x_n). \tag{5}$$

Convergence of Descent Algorithm

Theorem: Consider the sequence x_n generated by an algorithm of the form:

$$x_{n+1} = x_n + \gamma s_n$$

where we suppose that for all n:

- $\|s_n\|_2 \geq K_1 \|\nabla F(x_n)\|_2$,
- $s_n^T \nabla F(x_n) \leq -K_2 \|s_n\|_2^2$,

if $\gamma \in (0, 2K_2/K)$, then:

$$\lim_{n \to +\infty} \nabla F(x_n) = 0. \tag{6}$$

Differential equations and Invariant set

We will focus on the asymptotic property of the following dynamical system:

$$\dot{x}(t) = f(x(t)), x(0) = x_0. \tag{7}$$

We will first recall the following definitions:

- A set A is *positively (resp. negatively) invariant* for (7) if for all $x_0 \in A$ implies that the corresponding $x(t) \in A$ for all $t > 0$ (resp. $t < 0$). A is *invariant* if it is both positively and negatively invariant.
- A compact (more generally, closed) invariant set M will be called an *attractor* if it has an open neighbourhood O such that every trajectory in O remains in O and converges to M .
- A point $x^* \in \mathbb{R}^n$ is an *equilibrium point* of the system if $f(x^*) = 0$. Note that $M = \{x^*\}$ is an invariant set.

Lyapunov stability

- A compact invariant set M will be said to be *Liapunov stable* if for any $\epsilon > 0$, there exists a $\delta > 0$ such that every trajectory initiated in the δ-neighbourhood of M remains in its ϵ-neighbourhood. *(Wikipedia: Lyapunov stability of an equilibrium means that solutions starting "close enough" to the equilibrium remain "close enough" forever.)*
- A compact invariant set M is said to be *asymptotically stable* if it is both Liapunov stable and an attractor. *(Wikipedia: Asymptotic stability means that solutions that start close enough not only remain close enough but also eventually converge to the equilibrium.)*

Illustration

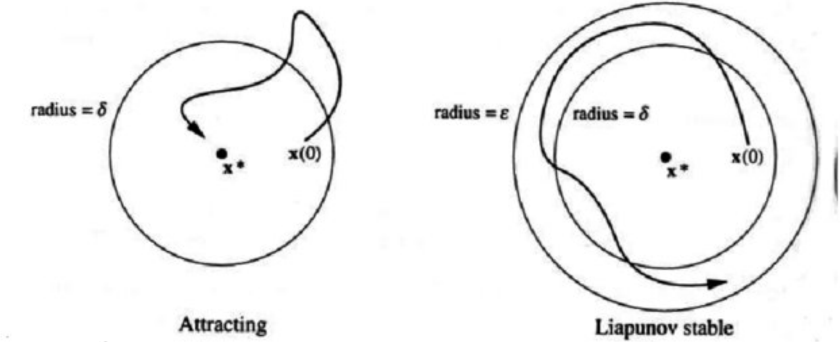

Figure: Attracting and Lyapunov stable equilibrium points, from Strogatz 1998

LaSalle Invariance Principle

Theorem: If one has a continuously differentiable $V : \mathbb{R}^d \to \mathbb{R}$ with $V(x) \to \infty$ as $\|x\| \to \infty$ and for all $x \in \mathbb{R}^d$, $\nabla V(x)^T f(x) \leq 0$ for all x, then any trajectory $x(t)$ must converge to the largest invariant set contained in $M = \{x : \nabla V(x)^T f(x) = 0\}$.

Example: Gradient descent

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function that we wish to minimize. The gradient algorithm in the continuous time is given by:

$$\dot{x} = -\nabla f(x),\ x(0) = x_0.$$

Exercise: show that $f(x)$ is a Lyapunov function for the above defined differential equation. What can you conclude from the LaSalle Invariance Principle.

Initial Model

- Let us assume that when we use the *measurement strategy* $k \in \{1, \ldots, n\}$, we get a multidimensional output $y^{(k)} \in \mathbb{R}^d$ which has the following form:

Initial Model

- Let us assume that when we use the *measurement strategy* $k \in \{1, \ldots, n\}$, we get a multidimensional output $y^{(k)} \in \mathbb{R}^d$ which has the following form:

$$y^{(k)} = A^{(k)}x + \epsilon^{(k)},$$

where $A^{(k)} \in \mathbb{R}^{d \times l}$, $\epsilon^{(k)}$ is some zero-mean noise and finally $x \in \mathbb{R}^l$ is the **hidden signal** to estimate.

Initial Model

- Let us assume that when we use the *measurement strategy* $k \in \{1, \ldots, n\}$, we get a multidimensional output $y^{(k)} \in \mathbb{R}^d$ which has the following form:

$$y^{(k)} = A^{(k)}x + \epsilon^{(k)},$$

where $A^{(k)} \in \mathbb{R}^{d \times l}$, $\epsilon^{(k)}$ is some zero-mean noise and finally $x \in \mathbb{R}^l$ is the **hidden signal** to estimate.

- Let us assume that we can only chose ϕ measurement strategy.

Initial Model

- Let us assume that when we use the *measurement strategy* $k \in \{1, \dots, n\}$, we get a multidimensional output $y^{(k)} \in \mathbb{R}^d$ which has the following form:

$$y^{(k)} = A^{(k)}x + \epsilon^{(k)},$$

  where $A^{(k)} \in \mathbb{R}^{d \times l}$, $\epsilon^{(k)}$ is some zero-mean noise and finally $x \in \mathbb{R}^l$ is the **hidden signal** to estimate.
- Let us assume that we can only chose ϕ measurement strategy.

What should be the measurements selected knowing that we want to build an *accurate* estimate of X?

Definition of the general observations

- $w_k = 1$ if the measurement strategy k is chosen and 0 otherwise.

Definition of the general observations

- $w_k = 1$ if the measurement strategy k is chosen and 0 otherwise.
- When one is using a particular set of strategies $w = [w_k]_{1 \leq k \leq n}$ we will have the following observations:

Definition of the general observations

- $w_k = 1$ if the measurement strategy k is chosen and 0 otherwise.
- When one is using a particular set of strategies $w = [w_k]_{1 \leq k \leq n}$ we will have the following observations:

$$y^T = A(w)^T x + \epsilon,$$

Definition of the general observations

- $w_k = 1$ if the measurement strategy k is chosen and 0 otherwise.
- When one is using a particular set of strategies $w = [w_k]_{1 \leq k \leq n}$ we will have the following observations:

$$y^T = A(w)^T x + \epsilon,$$

where $y := [y^{(k)T}]_{k|w_k=1}$ and $A(w) := [(A^{(k)})^T]_{k|w_k=1}$ are simply the concatenations of the observed outputs and the associated matrices.

Gauss Markov Theorem

- If we assume that $\Sigma = \mathbb{E}[\epsilon\epsilon^T]$ is a diagonal matrix then the **Gauss Markov Theorem** is telling us that the best linear unbiased estimator is given by:

$$\hat{x} = \left(A(w)\Sigma^{-1}A(w)^T\right)^{-1} A(w)\Sigma^{-1}y^T$$

Gauss Markov Theorem

- If we assume that $\Sigma = \mathbb{E}[\epsilon\epsilon^T]$ is a diagonal matrix then the **Gauss Markov Theorem** is telling us that the best linear unbiased estimator is given by:

$$\hat{x} = \left(A(w)\Sigma^{-1}A(w)^T\right)^{-1} A(w)\Sigma^{-1}y^T$$

and its associated variance is equal to:

$$\boxed{\text{Var}[\hat{x}] = \left(A(w)\Sigma^{-1}A(w)^T\right)^{-1}}$$

Gauss Markov Theorem

- If we assume that ϵ are independent one from another (i.e. $\Sigma = \mathbb{E}[\epsilon\epsilon^T]$ is a diagonal matrix) then the **Gauss Markov Theorem** is telling us that the best linear unbiased estimator is given by:

$$\hat{x} = \left(A(w)\Sigma^{-1}A(w)^T\right)^{-1} A(w)\Sigma^{-1}y^T$$

and its associated variance is equal to:

$$\boxed{\mathsf{Var}[\hat{x}] = \left(A(w)A(w)^T\right)^{-1}}$$

Gauss Markov Theorem

- If we assume that ϵ are independent one from another (i.e. $\Sigma = \mathbb{E}[\epsilon\epsilon^T]$ is a diagonal matrix) then the **Gauss Markov Theorem** is telling us that the best *linear unbiased estimator* is given by:

$$\hat{x} = \left(A(w)\Sigma^{-1}A(w)^T\right)^{-1} A(w)\Sigma^{-1}y^T$$

  and its associated variance is equal to:

$$\boxed{\text{Var}[\hat{x}] = M(w)^{-1},\ M(w) = \sum_{k=0}^{n} w_k (A^{(k)})^T A^{(k)}}$$

Gauss Markov Theorem

- If we assume that ϵ are independent one from another (i.e. $\Sigma = \mathbb{E}[\epsilon\epsilon^T]$ is a diagonal matrix) then the **Gauss Markov Theorem** is telling us that the best *linear unbiased estimator* is given by:

$$\hat{x} = \left(A(w)\Sigma^{-1}A(w)^T\right)^{-1} A(w)\Sigma^{-1}y^T$$

and its associated variance is equal to:

$$\boxed{\text{Var}[\hat{x}] = M(w)^{-1},\ M(w) = \sum_{k=0}^{n} w_k (A^{(k)})^T A^{(k)}}$$

Challenge: How to minimize variance $M(w)^{-1}$?

Optimisation problem

To minimise the variance we will solve the following optimisation problem:

$$\max_{w \in \{0,1\}^n} g\left(\sum_{k=0}^{n} w_k (A^{(k)})^T A^{(k)}\right), \text{ s.t. } \sum_{k=1}^{n} w_k = \phi. \tag{8}$$

Optimisation problem

To minimise the variance we will solve the following optimisation problem:

$$\max_{w\in\{0,1\}^n} g\Big(\sum_{k=0}^{n} w_k (A^{(k)})^T A^{(k)}\Big), \text{ s.t. } \sum_{k=1}^{n} w_k = \phi. \tag{8}$$

But what is $g(\cdot)$?

Optimisation problem

To minimise the variance we will solve the following optimisation problem:

$$\max_{w\in\{0,1\}^n} g\Big(\sum_{k=0}^{n} w_k (A^{(k)})^T A^{(k)}\Big), \text{ s.t. } \sum_{k=1}^{n} w_k = \phi. \qquad (8)$$

But what is $g(\cdot)$? Usually $g(\cdot)$ is a scalar information function of the matrix $M(w)$ which satisfies:

- positive homogeneity,
- monotonicity with respect to Loewner ordering,
- concavity.

Typically $g(M) = \lambda_{min}(M)$ but also $g(M) = \text{trace}(M^p)^{1/p}$

Properties of the optimisation problem

- **Complexity:** NP-Hard.

Properties of the optimisation problem

- **Complexity:** NP-Hard.
- **Heuristics:** Greedy algorithm is efficient (submodular properties) and convex relaxation can be used.

Properties of the optimisation problem

- **Complexity:** NP-Hard.
- **Heuristics:** Greedy algorithm is efficient (submodular properties) and convex relaxation can be used.
- **Issues of this classical formulation:** linear, A is known, and we need *the Gauss Markov Theorem*.

Properties of the optimisation problem

- **Complexity:** NP-Hard.
- **Heuristics:** Greedy algorithm is efficient (submodular properties) and convex relaxation can be used.
- **Issues of this classical formulation:** linear, A is known, and we need *the Gauss Markov Theorem*.

Quick reminder in Optimisation, and Differential equations

Problem I: Experimental design

Problem II: Backpropagation for Distributed Resource Allocation

Background on Resource Allocation in Networks

Resource Allocation Problem

$$\max_{\mathbf{u}\in\mathcal{U}} \quad \sum_{i=1}^{I} U_i(x_i),$$

subject to $\mathbf{x}$ solution of:

$$x_i = f_i(u_i, x_i, \{y_j\}_{j\in\mathcal{N}_+(i)}),\ \forall i \in \{1,\dots,I\},$$
$$y_i = g_i(x_i),\ \forall i \in \{1,\dots,I\}.$$

Background on Resource Allocation in Networks

Resource Allocation Problem

$$\max_{\mathbf{u}\in\mathcal{U}} \quad \sum_{i=1}^{I} U_i(x_i),$$

$\text{subject to } \mathbf{x}$ solution of:

$$x_i = f_i(u_i, x_i, \{y_j\}_{j\in\mathcal{N}_+(i)}),\ \forall i \in \{1,\ldots,I\},$$
$$y_i = g_i(x_i),\ \forall i \in \{1,\ldots,I\}.$$

- $\mathcal{I}$: the set of nodes;

Background on Resource Allocation in Networks

Resource Allocation Problem

$$\max_{\mathbf{u}\in\mathcal{U}} \quad \sum_{i=1}^{I} U_i(x_i),$$

$$\text{subject to } \mathbf{x} \text{ solution of:}$$

$$x_i = f_i(u_i, x_i, \{y_j\}_{j\in\mathcal{N}_+(i)}),\ \forall i \in \{1,\ldots,I\},$$

$$y_i = g_i(x_i),\ \forall i \in \{1,\ldots,I\}.$$

- $\mathcal{I}$: the set of nodes;
- W: Adjacency matrix of the directed network;

Background on Resource Allocation in Networks

Resource Allocation Problem

$$\max_{\mathbf{u}\in\mathcal{U}} \quad \sum_{i=1}^{I} U_i(x_i),$$

$\text{subject to } \mathbf{x}$ solution of:

$$x_i = f_i(u_i, x_i, \{y_j\}_{j\in\mathcal{N}_+(i)}),\ \forall i \in \{1,\dots,I\},$$
$$y_i = g_i(x_i),\ \forall i \in \{1,\dots,I\}.$$

- $\mathcal{I}$: the set of nodes;
- W: Adjacency matrix of the directed network;
- $\mathcal{N}_+(i)$: set of in-neighbors of node i;

Background on Resource Allocation in Networks

Resource Allocation Problem

$$\max_{\mathbf{u}\in\mathcal{U}} \quad \sum_{i=1}^{I} U_i(x_i),$$

subject to $\mathbf{x}$ solution of:

$$x_i = f_i(u_i, x_i, \{y_j\}_{j\in\mathcal{N}_+(i)}),\ \forall i \in \{1,\ldots,I\},$$
$$y_i = g_i(x_i),\ \forall i \in \{1,\ldots,I\}.$$

- $\mathcal{I}$: the set of nodes;
- W: Adjacency matrix of the directed network;
- $\mathcal{N}_+(i)$: set of in-neighbors of node i;
- $\mathcal{N}_-(i)$: set of out-neighbors of node i.

Applications

Resource Allocation Constraints

$\mathbf{x}$ solution of:

$$x_i = f_i(u_i, x_i, \{y_j\}_{j\in\mathcal{N}_+(i)}),\ \forall i \in \{1,\dots,I\},$$
$$y_i = g_i(x_i),\ \forall i \in \{1,\dots,I\}.$$

- **Opinion Shaping in Social Networks:** $f_i(u_i, x_i, \{y_j\}_{j\in\mathcal{N}_+(i)}) := \alpha_i h(u_i) + \beta_i([W\mathbf{x}]_i) + (1-\alpha_i-\beta_i)x_i$, $g_i(x_i) = x_i$.
- **Distributed ledger:** $f_i(u_i, x_i, \{y_j\}_{j\in\mathcal{N}_+(i)}) = x_i(1-\frac{1}{x})^{k\sum_i u_i}$.
- **Hierarchical Network models in Economy:** $f_i(u_i, x_i, \{y_j\}_{j\in\mathcal{N}_+(i)}) := u_i + \sum_{j=1}^{I} w_{ji} y_j$, $y_i = f_i\left(x_i\right)$.

Today's Challenges

1. How to solve the Resource Allocation Problem using decentralized and asynchronous algorithms?

Today's Challenges

1. How to solve the Resource Allocation Problem using decentralized and asynchronous algorithms?
2. How to extend such algorithms when the adjacency matrix is unknown?

Today's Challenges

1. How to solve the Resource Allocation Problem using decentralized and asynchronous algorithms?
2. How to extend such algorithms when the adjacency matrix is unknown?

Mathematical tools used: Convex optimization, Stochastic Approximation and Markov chains.

Why is it important?

Distributed Systems: multiple computers connected through a network and trying to solve a given problem.

Why is it important?

Distributed Systems: multiple computers connected through a network and trying to solve a given problem.

But why do we need distributed systems? (why not 1 computer to rule them all?)

Why is it important?

Distributed Systems: multiple computers connected through a network and trying to solve a given problem.

But why do we need distributed systems? (why not 1 computer to rule them all?)

Answers: *Not robust to failure, Limited computation/storage/, Physical Location.*

Why is it important?

Distributed Systems: multiple computers connected through a network and trying to solve a given problem.

But why do we need distributed systems? (why not 1 computer to rule them all?)

Answers: *Not robust to failure, Limited computation/storage/, Physical Location.*

A distributed system has the following features:

- No common physical clock
- No Shared memory
- Geographical separation
- Autonomy and heterogeneity

Decentralized and Asynchronous algorithms

Decentralized algorithm: An algorithm is said to be *decentralized* when there is no need for a central controller to execute it.

Decentralized and Asynchronous algorithms

Decentralized algorithm: An algorithm is said to be *decentralized* when there is no need for a central controller to execute it.

Asynchronous algorithm: An algorithm is said to be *asynchronous* when there is no need for a common central clock to execute it.

Related Works

- Primal-dual and saddle point algorithms;
- The push-pull method for network optimization and distributed learning
- Handling chance-constraints and general nonlinear constraints under NUM
- Handling nonlinear constraints

Stochastic approximation

A stochastic approximation scheme has usually the following form:

$$w_{k+1} = \Gamma\left[w_k + \alpha_k(h(w_k) + M_{k+1})\right],\ w_0 = w^0.$$

The classical ingredients are:

- The previous guess;
- a small-update for which its amplitude is controlled by the step-size. The update is composed of:
  1. A deterministic function +
  2. A noise with zero mean.
- $\Gamma[\cdot]$ is a projection function to ensure a good behavior of the iterative scheme.

Assumptions:

- The function h is Lipschitz.
- The stepsizes $\{a_k\}$ are square summable but not summable.
- $\{M_k\}$ is a martingale difference sequence (need to define the correct σ-field).
- The iterates w_k remain bounded a.s.
- There exists a continuously positive differentiable function V such that $\lim_{\|x|\to\infty} V(x) = \infty$, $H := \{x \mid V(x) = 0\} \neq \emptyset$ and $\langle h(x), \nabla V(x)\rangle \leq 0$ with equality iff $x \in H$. (V is a Liapunov function)

Convergence theorem: The O.D.E approach

Theorem

Under the assumptions stated in the previous slides, the behavior of $\lim_{k\to+\infty} w_k$, where

$$\boxed{w_{k+1} = w_k + \alpha_k(h(w_k) + M_{k+1}),\ w_0 = w^0,}$$

is the same as the behavior of $\lim_{t\to+\infty} w(t)$, where $w(t)$ is the solution of the ordinary differential equations:

$$\boxed{\dot{w}(t) = h(w),\ w(0) = w^0.}$$

Why is it true? An intuition

Recall that the standard 'Euler scheme' for numerically approximation a trajectory of the o.d.e

$$\dot{x}(t) = h(x(t))$$

would be

$$x_{n+1} = x_n + ah(x_n),$$

where $a > 0$ is a small step.
The stochastic approximation iteration differs from this in two aspects:

1. Possible replacement of the constant time step a by a time-varying α_k (for instance $\frac{1}{k+1}$).
2. The presence of the noise M_{k+1}.

This is why the stochastic approximation scheme is nothing more than a noisy discretization of the o.d.e.

Why this theorem/ the O.D.E. approach

- There are two approaches to the theoretical analysis of such algorithms:
    1. Probabilistic approach, popular among statisticians,
    2. O.D.E. approach, more popular among engineers.
- The O.D.E. approach can serve as useful recipe for concocting new algorithms: any convergent o.d.e. is a potential source of a stochastic approximation algorithm that converge to the desired one.

Exercise: Asynchronous stochastic approximation

Let us assume that we have I processors. For iteration $k \in \mathbb{N}_+$ and for every processor $i \in \{1, \dots, I\}$, $w_k^i \in \mathbb{R}$ denotes the update of processor i at instant k. We assume that every processor i follows the update rule below:

$$w_{k+1}^i = w_k^i + \frac{1}{n+1}\xi_n^i,$$

where

$$\begin{array}{rcl} \xi_n^i & = & h(w_k^i) \text{ with probability } p_i, \\ & = & 0 \text{ with probability } 1 - p_i. \end{array}$$

- Rewrite down the recursive update as a stochastic approximation. What is the associated O.D.E. ?
- What can you conclude about the convergence of our algorithm?

Differential Equations with fast components

Let us consider a system described by state variables x and y whose evolution is

$$\begin{aligned} \dot{x}(t) &= \epsilon f(x(t), y(t)),\ x(0) = x_0, && (9) \\ \dot{y}(t) &= g(x(t), y(t)),\ y(0) = y_0 && (10) \end{aligned}$$

When ϵ is small enough, the solution of (9)-(10),

- *at the fast-time scale*, is *closed* to the solution of:

$$\begin{aligned} \dot{x}(t) &= 0,\ x(0) = x_0, \\ \dot{y}(t) &= g(x(t), y(t)),\ y(0) = y_0. \end{aligned}$$

- *at the slow-time scale*, is *closed* to the solution of:

$$\begin{aligned} \dot{x}(t) &= \epsilon f(x(t), y(t)),\ x(0) = x_0, \\ 0 &= g(x(t), y(t)). \end{aligned}$$

  In particular, if $g(x, y(x)) = 0$ for every x, then we have $\dot{x}(t) = f(x(t), y(t)),\ x(0) = x_0$.

Classical applications

Such technique has been used for reduced order modeling:

- In optimal control:
  - "Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations", Bardi, Martino, and Italo Capuzzo Dolcetta (see chapter 7 section 4).
  - "A Mean-field Approach for Controlling Singularly Perturbed Multi-population SIS Epidemics" Eitan Altman, Rajesh Sundaresan.
- In differential equations stability:
  - "Dynamics of a fishery on two fishing zones with fish stock dependent migrations: aggregation and control", R. Mchich, P.M. Auger, R. Bravo de la Parra, N. Raissi.
  - "Methods of aggregation of variables in population dynamics", Auger, P.M., Bravo de la Parra, R.

Multiple Timescales stochastic approximations

- The classical two timescales stochastic approximation can be rewritten as:

$$\begin{aligned} w_{k+1} &= w_k + \alpha_k(h(w_k, v_k) + M^1_{k+1}),\ w_0 = w^0, \\ v_{k+1} &= v_k + \beta_k(g(w_k, v_k) + M^2_{k+1}),\ v_0 = v^0, \end{aligned}$$

  with $\beta_k/\alpha_k \to 0$.
- To the study the limiting behavior of such dynamics, then we simply need to study the following O.D.E.:

$$\begin{aligned} \dot{w} &= h(w, v),\ w(0) = w^0, \\ \dot{v} &= \epsilon g(w, v),\ v(0) = v^0. \end{aligned}$$

Advantages of this approach

Heavily used to decoupled the estimation from the control.

For instance,

- Estimation of the gradient is performed at the fast timescale.
- A gradient descent algorithm is used at the slow timescale.

Machine learning application

Such technique has been used to design new algorithms.

1. Consensus based optimization and synchronisation.
2. Gradient descent with approximate projections.
3. Reinforcement learning:
   - "A tale of two-timescale reinforcement learning with the tightest finite-time bound", Dalal, Gal, Balazs Szorenyi, and Gugan Thoppe.
   - "An actor-critic algorithm for constrained Markov decision processes", Vivek Borkar.
4. Resource allocation:
   - "Opinion shaping in social networks using reinforcement learning", Vivek Borkar, Alexandre Reiffers-Masson.
   - "A Backpropagation Approach for Distributed Resource Allocation", Alexandre Reiffers-Masson, Nahum Shimkin, Daniel Sadoc Menasche, Eitan Altman.

Model

Hierarchical Network models in Economy

$$\max_{\mathbf{u}\in[\underline{u},\overline{u}]^I} \quad \sum_{i=1}^{I} U_i(x_i) - \sum_{i=1}^{I} C_i(u_i) := U(\mathbf{x}) - C(\mathbf{u}),$$

subject to $\mathbf{x}$ solution of:

$$x_i = u_i + \sum_{j=1}^{I} f_j\left(x_j\right) w_{ji}, \; \forall i \in \{1, \ldots, I\}.$$

Agents are connected through a *directed acyclic graph* (DAG), where relationships are described by an adjacency matrix $\mathbf{W}$.

Optimality conditions

The Karush Kuhn Tucker (KKT) conditions, $\forall i \in \{1, \ldots, I\}$:

$$
\begin{aligned}
x_i &= u_i + \sum_{j=1}^{I} f_j\left(x_j\right) w_{ji}, \\
\lambda_i &= U_i'(x_i) + f_i'(x_i) \sum_{j=1}^{I} w_{ij} \lambda_j, \\
0 &= -C_i'(u_i) + \lambda_i.
\end{aligned}
$$

Backpropagation Algorithm for Distributed Resource Allocation

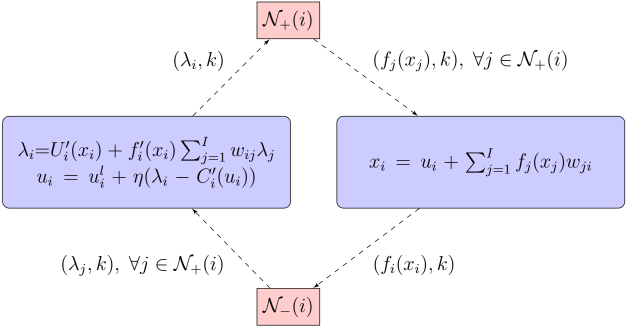

Figure: Block diagram of the synchronous resource allocation distributed algorithm, at the level of node i. Upstream neighbors ($\mathcal{N}_+(i)$) transfer flows at the forward step, and downstream neighbors ($\mathcal{N}_-(i)$) backpropagate gradients at the backward step. The current time slot, k, is included in all messages.

Distributed asynchronous version: Pulling version

When node i is activated: (1) requests the outputs from its neighbors; (2) he performs the following updates:

$$
\begin{aligned}
x_i^{k+1} =& x_i^k + a(\nu(i,k)) \left(u_i^k + \sum_{j \in \mathcal{N}_+(i)} f_j(x_j^{k-\tau_{ij}(k)}) w_{ji} - x_i^k \right), \\
\lambda_i^{k+1} =& \lambda_i^k + b(\nu(i,k)) \; \left(U_i'(x_i(k)) - \lambda_i^k + f_i'(x_i^k) \sum_{j \in \mathcal{N}_-(i)} w_{ij} \lambda_j^{k-\tau_{ij}(k)} \right), \\
u_i^{k+1} =& \left[u_i^k + c(\nu(i,k)) \left(\lambda_i^k - C_i'(u_i^k) \right) \right]_{\underline{u}}^{\overline{u}}.
\end{aligned}
$$

Differences!

$$
\begin{aligned}
x_i^{k+1} =& x_i^k + a(\nu(i,k)) \left(u_i^k + \sum_{j \in \mathcal{N}_+(i)} f_j(x_j^{k-\tau_{ij}(k)}) w_{ji} - x_i^k \right), \\
\lambda_i^{k+1} =& \lambda_i^k + b(\nu(i,k)) \left(U_i'(x_i(k)) - \lambda_i^k + f_i'(x_i^k) \sum_{j \in \mathcal{N}_-(i)} w_{ij} \lambda_j^{k-\tau_{ij}(k)} \right), \\
u_i^{k+1} =& \left[u_i^k + c(\nu(i,k)) \left(\lambda_i^k - C_i'(u_i^k) \right) \right]_{\underline{u}}^{\overline{u}}.
\end{aligned}
$$

This scheme is a distributed, three time scale stochastic approximation. Therefore, using the O.D.E approach we can prove the convergence of this scheme to the desired point.

Quick Proof

We consider the following singularly perturbed ordinary differential equations,

$$\begin{aligned}
\dot{x}_i &= u_i + \sum_{j \in \mathcal{N}_+(i)} w_{ji} f_j(x_j) - x_i, && (11) \\
\dot{\lambda}_i &= \epsilon_1 \left(U_i'(x_i) - \lambda_i + f_i'(x_i) \sum_{j \in \mathcal{N}_-(i)} w_{ij} \lambda_j \right), && (12) \\
\dot{u}_i &= \epsilon_2 (C_i'(u_i) - \lambda_i - v_i(\mathbf{u})), && (13)
\end{aligned}$$

$\forall i \in \{1, \ldots, I\}$, with $0 < \epsilon_1 \downarrow 0$, $0 < \epsilon_2 \downarrow 0$, $\frac{\epsilon_2}{\epsilon_1} \downarrow 0$ and where $\mathbf{v}(\mathbf{u}) = [v_i(\mathbf{u})]_{1 \le i \le I}$ is the minimum adjustment needed to keep $\mathbf{u}$ in $[\underline{u}, \overline{u}]^I$, noting that (13) is a projected dynamical system.

The three timescales

Fast timescale:

$$\dot{x}_i = u_i + \sum_{j \in \mathcal{N}_+(i)} w_{ji} f_j(x_j) - x_i, \quad \dot{\lambda}_i = 0, \quad \dot{u}_i = 0.$$

The three timescales

Fast timescale:

$$\dot{x}_i = u_i + \sum_{j \in \mathcal{N}_+(i)} w_{ji} f_j(x_j) - x_i, \quad \dot{\lambda}_i = 0, \quad \dot{u}_i = 0.$$

Middle timescale:

$$\begin{aligned}
0 &= u_i + \sum_{j \in \mathcal{N}_+(i)} w_{ji} f_j(x_j^*(\mathbf{u})) - x_i^*(\mathbf{u}), \\
\dot{\lambda}_i &= U_i'(x_i^*(\mathbf{u})) - \lambda_i + f_i'(x_i^*(\mathbf{u})) \sum_{j \in \mathcal{N}_-(i)} w_{ij} \lambda_j, \\
\dot{u}_i &= 0,
\end{aligned}$$

The three timescales

Slow timescale:

$$
\begin{aligned}
0 &= u_i + \sum_{j \in \mathcal{N}_+(i)} w_{ji} f_j(x_j^*(\mathbf{u})) - x_i^*(\mathbf{u}), \\
0 &= U_i'(x_i^*(\mathbf{u})) - \lambda_i(\mathbf{x}^*(\mathbf{u})) + f_i'(\mathbf{x}^*(\mathbf{u})) \sum_{j \in \mathcal{N}_-(i)} w_{ij} \lambda_j(\mathbf{x}^*(\mathbf{u})), \\
\dot{u}_i &= C_i'(u_i) - \lambda_i(\mathbf{x}^*(\mathbf{u})) - v_i(\mathbf{u}).
\end{aligned}
$$

Extensions

- Unknown adjacency matrix: Borkar, Vivek S., and Alexandre Reiffers. "Opinion shaping in social networks using reinforcement learning." IEEE Transactions on Control of Network Systems (2021).
- Distributed ledger management: Jay, M., Mollard, A., Sun, Y., Zheng, R., Amigo, I., Reiffers-Masson, A., & Rincón, S. (2021). Utility maximisation in the Coordinator-less IOTA Tangle. UNET 2021 (Best Paper)
- Reiffers-Masson, Alexandre, et al. "A Backpropagation Approach for Distributed Resource Allocation." (2021).

Acknowledgment

- Bertsekas, Dimitri, and John Tsitsiklis. Parallel and distributed computation: numerical methods. Athena Scientific, 2015. (ch. 3)
- Borkar, V. S. (2009). Stochastic approximation: a dynamical systems viewpoint (Vol. 48). Springer.
- Lecture 12: Basic Lyapunov theory (EE363 Stanford, Boyd)