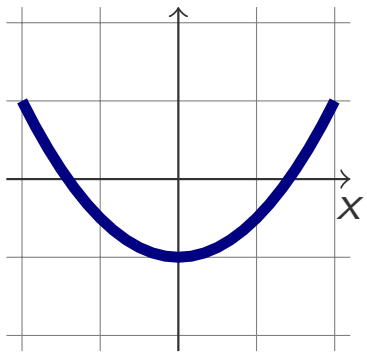
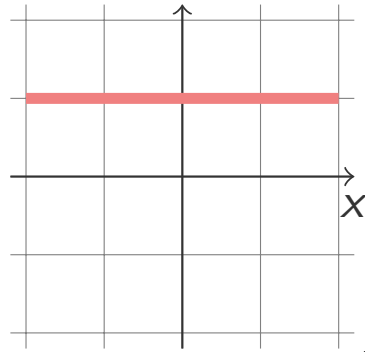
Asynchronous distributed optimization using coordinate descent

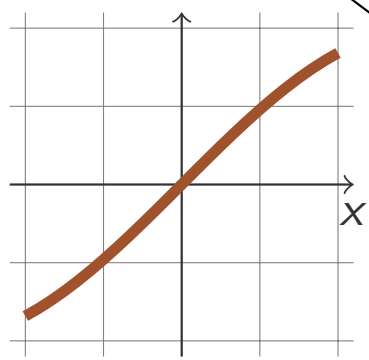
P. Bianchi

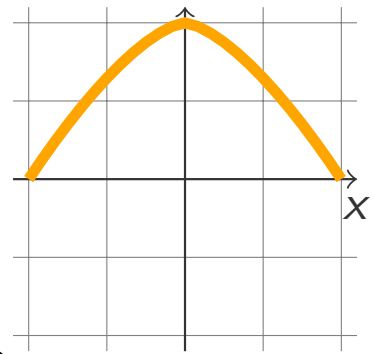
Joint work with
O. Fercoq, W. Hachem, F. Iutzeler

The Problem

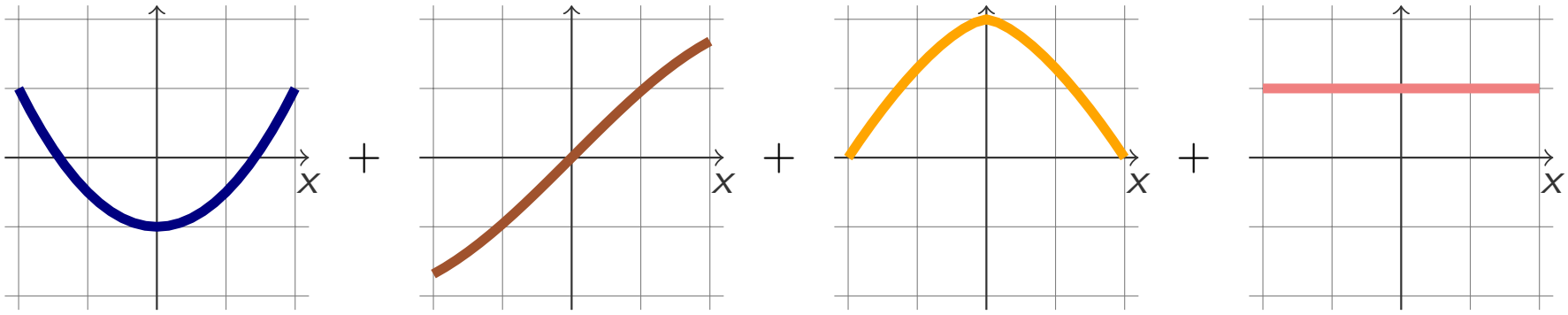




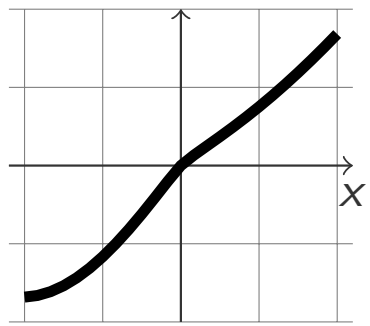




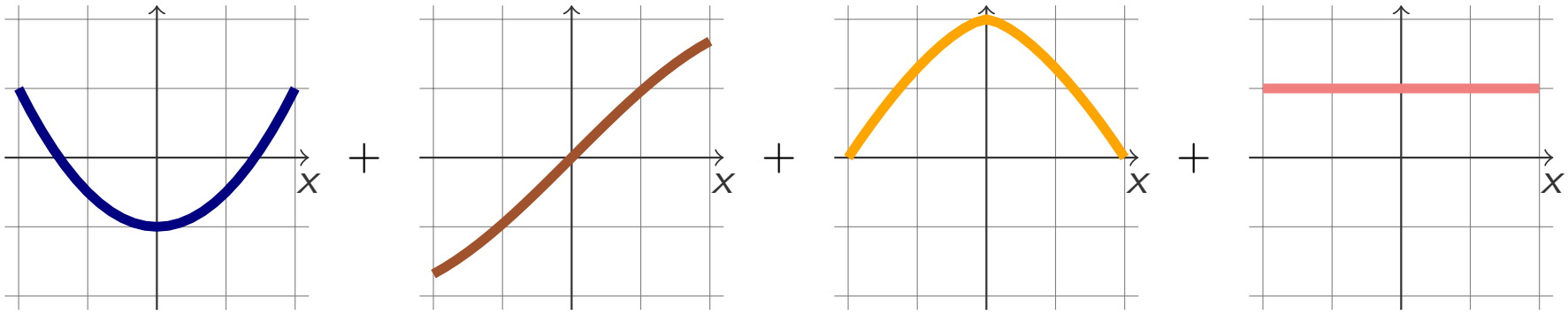
The Problem



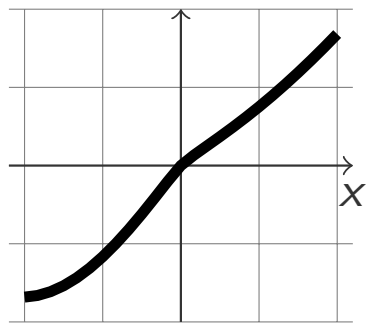
=



The Problem



=



No single agent knows the target function to optimize

Formally

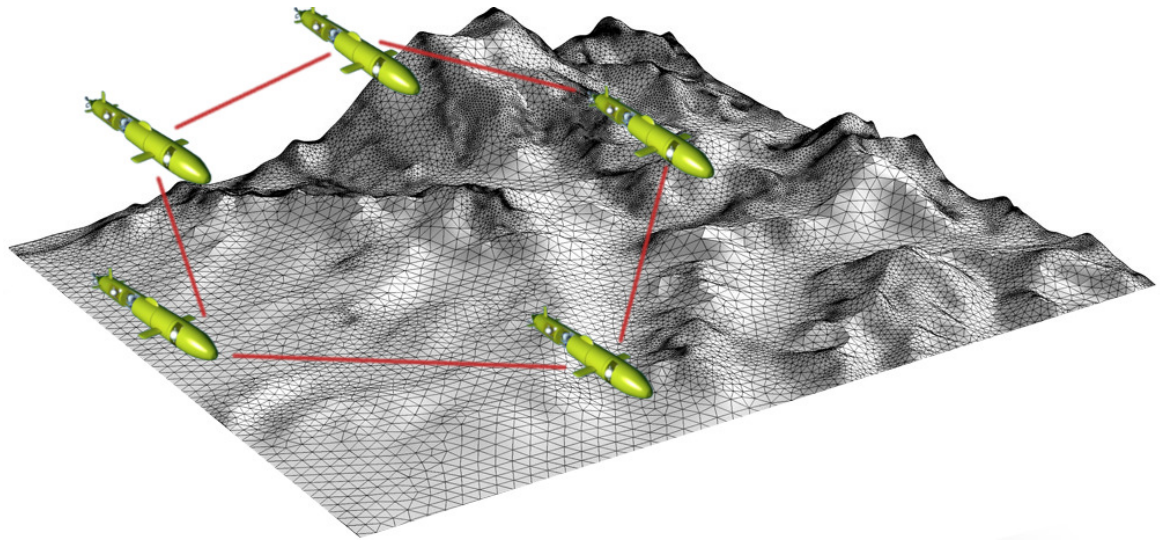
$$\min_{x \in \mathcal{X}} \sum_{n=1}^{N} f_n(x)$$

- $\mathcal{X} = \mathbb{R}^d$
- f_n is the cost function of agent n
- Two agents n and m can exchange messages if $n \sim m$

Numerous works on that problem
Early work: Tsitsiklis '84

Example #1: wireless sensor networks

Y_n = random observation of sensor n
x = unknown parameter to be estimated



$$p(Y_1, \cdots, Y_N; x) = p_1(Y_1; x) \cdots p_N(Y_N; x)$$

The maximum likelihood estimate writes

$$\hat{x} = \arg\max_x \sum_n \ln p_n(Y_n; x)$$

[Schizas'08, Moura'11]

Example #2: machine learning (1/2)

Data set formed by T samples (X_i, Y_i) $(i = 1 \dots T)$

- Y_i = variable to be explained
- X_i = explanatory features

Looking for a model.
Linear regression example:

$$\min_x \sum_{i=1}^{T} \| Y_i - x^T X_i \|^2$$

Example #2: machine learning (1/2)

Data set formed by T samples (X_i, Y_i) $(i = 1 \dots T)$

- Y_i = variable to be explained
- X_i = explanatory features

Looking for a model.

$$\min_x \sum_{i=1}^{T} \ell(x^T X_i, Y_i) + r(x)$$

Example #2: machine learning (1/2)

Data set formed by T samples (X_i, Y_i) $(i = 1 \dots T)$

- Y_i = variable to be explained
- X_i = explanatory features

Looking for a model.

$$\min_x \sum_{i=1}^{T} \ell(x^T X_i, Y_i) + r(x)$$

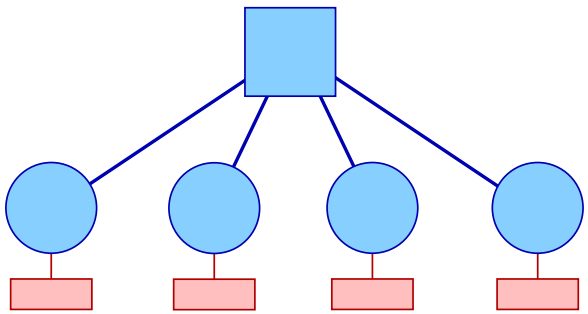
Divide and conquer: Split data into N batches

$$\min_x \sum_{n=1}^{N} \sum_{i} \ell(x^T X_{i,n}, Y_{i,n}) + r(x)$$

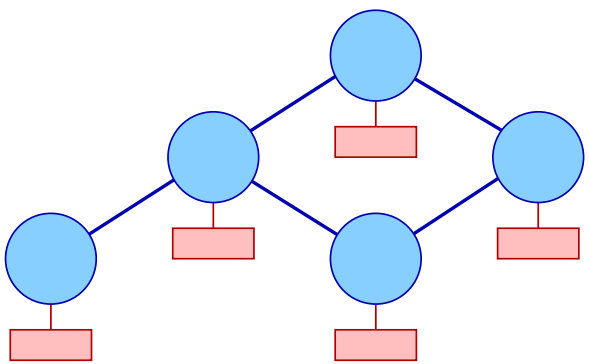
Process each batch individually

Example #2: machine learning (2/2)

- Centralized
  Select a batch n at time k
  Update the estimate x^k using the nth batch



- Parallel
  One agent/worker for one batch
  Master estimates x^k from agents' output

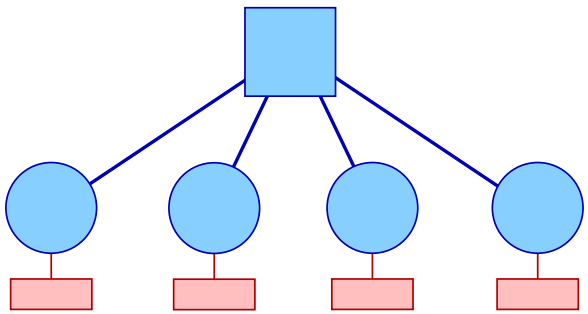


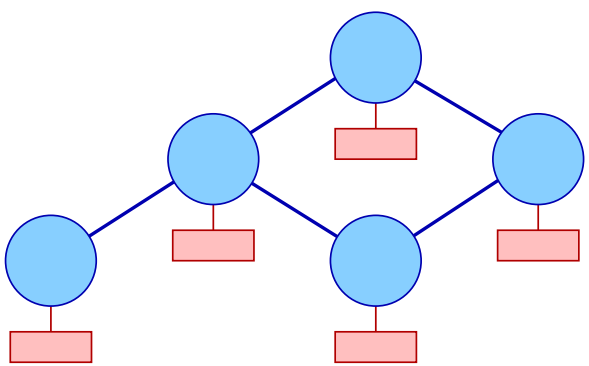
- Distributed
  One agent/worker for one batch
  Each agent communicates with its neighbors

Example #2: machine learning (2/2)

Iterative algorithms can be

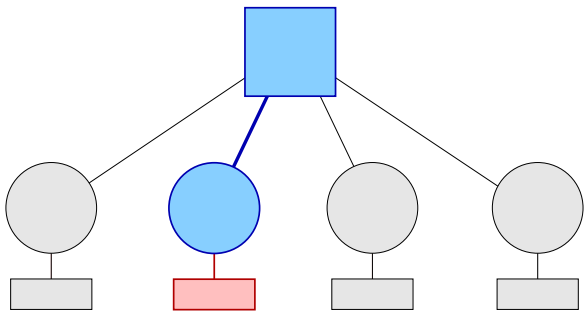
- Synchronous:
  process all batches at each iteration



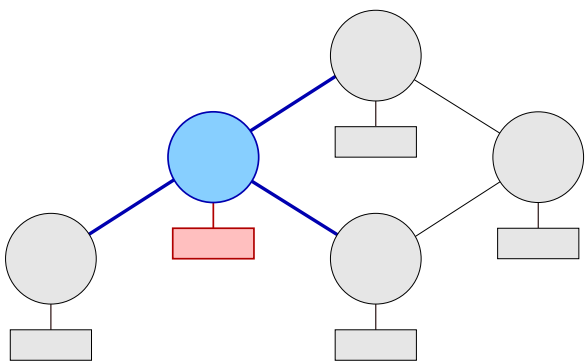


Example #2: machine learning (2/2)

Iterative algorithms can be



- Synchronous:
  process all batches at each iteration
- Asynchronous:
  Randomly select one batch/worker
  process one at a time



Main focus: distributed and asynchronous

Outline

Background

Coordinate descent

A primal dual algorithm

Application to distributed optimization

Outline

Background

Coordinate descent

A primal dual algorithm

Application to distributed optimization

Fixed point algorithms

Let $R : \mathcal{X} \to \mathcal{X}$ be non-expansive:

$$\|Rx - Ry\| \leq \|x - y\|$$

Goal: Find a fixed point $x^* \in \mathrm{fix}(R)$ (assumed $\neq \emptyset$)

First try: $x^{k+1} = R(x^k) \quad \to$ generally non convergent

Damped iterations: Set $0 < \alpha < 1$. The following iterations converge:

$$\begin{aligned} x^{k+1} &= (1-\alpha)x^k + \alpha R(x^k) \\ &= Tx^k \end{aligned}$$

where $T = (1-\alpha)I + \alpha R$. Clearly, $\mathrm{fix}\, T = \mathrm{fix}\, R$.

T is said α-averaged

Example 1: gradient descent

Let f convex differentiable s.t. ∇f is L-Lipschitz continuous

$$T(x) = x - \gamma \nabla f(x)$$

Assume $\gamma < 2/L$. Then

- $\operatorname{fix} T = \arg\min f$
- T is $(\gamma L/2)$-averaged

The iterations $x^{k+1} = x^k - \gamma \nabla f(x^k)$ converge to a minimizer of f.

Example 2: proximal point algorithm

Let g convex lower semicontinuous on $\mathcal{X}$ and $\gamma > 0$

$$\operatorname{prox}_{\gamma g}(x) = \arg\min_y g(y) + \frac{\|y - x\|^2}{2\gamma}$$

- $\operatorname{fix} \operatorname{prox}_{\gamma g} = \arg\min g$
- $\operatorname{prox}_{\gamma g}$ is $(1/2)$-averaged

The iterations $x^{k+1} = \operatorname{prox}_{\gamma g}(x^k)$ converge to a minimizer of g.

Example 3: proximal gradient algorithm

Fact: The composition of two averaged operators is an averaged operator

- $T = \mathrm{prox}_{\gamma g} \circ (I - \gamma \nabla f)$ is α-averaged
- $\mathrm{fix}(T) = \arg\min(f + g)$

The sequence

$$x^{k+1} = \mathrm{prox}_{\gamma g}(x^k - \gamma \nabla f(x^k))$$

converges to a minimizer of $f + g$

More examples later: Vũ-Condat algorithm, ADMM,...

Outline

Background

Coordinate descent

A primal dual algorithm

Application to distributed optimization

Coordinate descent (CD)

Assume $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_N$. Set

$$\begin{aligned} x &= (x_1, \ldots, x_N) \\ T(x) &= (T_1(x), \ldots, T_N(x)) \end{aligned}$$

For any $\ell \in \{1, \ldots, N\}$ and any $x = (x_1, \ldots, x_N)$, define

$$\hat{T}^{(\ell)}(x) = (x_1, \ldots, x_{\ell-1}, T_\ell(x), x_{\ell+1}, \ldots, x_N)$$

Algorithm: Let (ξ_k) be a sequence on $\{1, \ldots, N\}$

$$x^{k+1} = \hat{T}^{(\xi_{k+1})}(x^k)$$

Warga'63, Bertsekas'83, Luo'92: coordinate-wise minimization of a function f
Tseng'09, Nesterov'10: coordinate descent with first-order black box
Richtarik'12, Fercoq'13: proximal gradient, acceleration, parallelization
Iutzeler'13, Combettes'14, Bianchi'14: α-averaged operators, primal-dual algorithms

A convergence result

$$x^{k+1} = \hat{T}^{(\xi_{k+1})}(x^k)$$

Assumptions

- (ξ_k) is an i.i.d. sequence on $\{1, \dots, N\}$
- $\mathbb{P}(\xi_1 = \ell) > 0$ for all ℓ
- T is α-averaged $(0 < \alpha < 1)$
- $\operatorname{fix} T \neq \emptyset$

Result (Iutzeler'13). Almost surely, there exists $x^* \in \operatorname{fix} T$ such that

$$\lim_{k \to \infty} x^k = x^*$$

Generalization: update coordinates by *blocks* (rather than individually)

Two applications

Splitting accross variables

- Generate iterates $x^k = (x_1^k, \ldots, x_N^k)$ to solve:

$$\min_{x_1,\ldots,x_N} \quad f(x_1, \ldots, x_N)$$

- Update one coordinate at each iteration
- More in Olivier's talk

Splitting accross functions (this talk)

- Generate iterates x^k to solve:

$$\min_{x} \quad \sum_{n=1}^{N} f_n(x)$$

- Activate one function f_n at each iteration
- (Le Roux'12, Mairal'13)

Splitting accross functions

$\min_u \sum_n f_n(u)$ is equivalent to

$$\min_{x_1 \cdots x_N} \sum_n f_n(x_n) \quad \text{s.t. } x_1 = \cdots = x_N$$

Splitting accross functions

$\min_u \sum_n f_n(u)$ is equivalent to

$$\min_{x_1 \cdots x_N} \sum_n f_n(x_n) \quad \text{s.t. } x_1 = \cdots = x_N$$

which rewrites

$$\min_{x_1 \cdots x_N} \sum_n f_n(x_n) + \iota_C(x_1, \ldots, x_N)$$

where $C = \{(u, \ldots, u) : u \in \mathcal{X}\}$ and

$$\iota_C(x) = \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{otherwise} \end{cases}$$

Splitting accross functions

$\min_u \sum_n f_n(u)$ is equivalent to

$$\min_{x_1 \cdots x_N} \sum_n f_n(x_n) \quad \text{s.t. } x_1 = \cdots = x_N$$

which rewrites

$$\min_{x_1 \cdots x_N} \underbrace{\sum_n f_n(x_n)}_{f(x)} + \underbrace{\iota_C(x_1, \ldots, x_N)}_{g(x)}$$

where $C = \{(u, \ldots, u) : u \in \mathcal{X}\}$ and

$$\iota_C(x) = \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{otherwise} \end{cases}$$

Proximal gradient: $x^{k+1} = \text{prox}_{\gamma g} \circ (I - \gamma \nabla f)(x^k)$

$\rightarrow$ Apply coordinate descent on $(I - \gamma \nabla f) \circ \text{prox}_{\gamma g}$

MISO as a CD algorithm

$$\min \sum_n f_n(x) \tag{1}$$

MISO algorithm (Mairal'13): Choose $n \in \{1 \dots N\}$ at random at time k

$$\begin{aligned} x^k &= \tfrac{1}{N} \textstyle\sum_j y_j^k \\ y_n^{k+1} &= x^k - \gamma \nabla f_n(x^k) \end{aligned}$$

and $y_j^{k+1} = y_j^k$ for all $j \neq n$

Result: Almost surely, x^k tends to a minimizer of (1)

MISO as a CD algorithm

$$\min \sum_n f_n(x) + \underbrace{r(x)}_{\textit{regularization}} \tag{1}$$

MISO algorithm (Mairal'13): Choose $n \in \{1 \dots N\}$ at random at time k

$$\begin{aligned} x^k &= \operatorname{prox}_{\frac{\gamma}{N} r}\left(\tfrac{1}{N} \textstyle\sum_j y_j^k\right) \\ y_n^{k+1} &= x^k - \gamma \nabla f_n(x^k) \end{aligned}$$

and $y_j^{k+1} = y_j^k$ for all $j \neq n$

Result: Almost surely, x^k tends to a minimizer of (1)

Outline

Background

Coordinate descent

A primal dual algorithm

Application to distributed optimization

The problem

$$\min_x \; f(x) + g(x) + h(Mx)$$

- $M : \mathcal{X} \to \mathcal{Y}$ linear operator
- f, g, h convex - ∇f is L-Lipschitz continuous
- First order black-box on f
- prox_g and prox_h easy to compute

Boils down to finding a saddle point of the Lagrangian

$$\mathcal{L}(x, \lambda) = f(x) + g(x) + \langle \lambda, Mx \rangle - h^*(\lambda)$$

where h^* is the Legendre transform of h

The Vũ-Condat Algorithm

$$\begin{aligned} x^{k+1} &= \operatorname{prox}_{\gamma g}(x^k - \gamma \nabla f(x^k) - \gamma M^* \lambda^k) \\ \lambda^{k+1} &= \operatorname{prox}_{\rho^{-1} h^*}(\lambda^k + \rho^{-1} M(2x^{k+1} - x^k)) \end{aligned}$$

Theorem (Condat'13): Let $\frac{1}{\gamma} - \frac{\|M\|^2}{\rho} > \frac{L}{2}$. Then

$$(x^k, \lambda^k) \longrightarrow \text{a saddle point of } \mathcal{L}$$

Proof: Write the algorithm as

$$(x^{k+1}, \lambda^{k+1}) = T(x^k, \lambda^k)$$

for some α-averaged mapping T

Case $f = 0$: Alternating Direction Method of Multipliers (ADMM)

Set $f = 0$. Rewrite the problem as

$$\min_{x \in \mathcal{X}} \; g(x) + h(Mx) \;=\; \min_{y \in \mathcal{Y}} \underbrace{\left[\inf_{x : y = Mx} g(x) \right]}_{\tilde{g}(y)} + h(y)$$

Apply the Vũ-Condat algorithm on $\min \tilde{g} + h$ with $\gamma = \rho$

We obtain the standard ADMM

$$z^{k+1} = \arg\min_{w \in \mathcal{Y}} \left[h(w) + \frac{\|w - (Mx^k + \rho\lambda^k)\|^2}{2\rho} \right]$$

$$\lambda^{k+1} = \lambda^k + \rho^{-1}(Mx^k - z^{k+1})$$

$$x^{k+1} = \arg\min_{w \in \mathcal{X}} \left[g(w) + \frac{\|Mw - z^{k+1} + \rho\lambda^{k+1}\|^2}{2\rho} \right]$$

ADMM is an instance of the Vũ-Condat algorithm

ADMM+

Now assume $f \neq 0$ and M injective.

Applying a similar change of variable, the Vũ-Condat algorithm yields

$$
\begin{aligned}
z^{k+1} &= \arg\min_{w \in \mathcal{Y}} \left[h(w) + \frac{\|w - (Mx^k + \rho\lambda^k)\|^2}{2\rho} \right] \\
\lambda^{k+1} &= \lambda^k + \rho^{-1}(Mx^k - z^{k+1}) \\
u^{k+1} &= (1 - \tau\rho^{-1})Mx^k + \tau\rho^{-1}z^{k+1} \\
x^{k+1} &= \arg\min_{w \in \mathcal{X}} \left[g(w) + \langle \nabla f(x^k), w \rangle + \frac{\|Mw - u^{k+1} + \tau\lambda^{k+1}\|^2}{2\tau} \right]
\end{aligned}
$$

Outline

Background

Coordinate descent

A primal dual algorithm

Application to distributed optimization

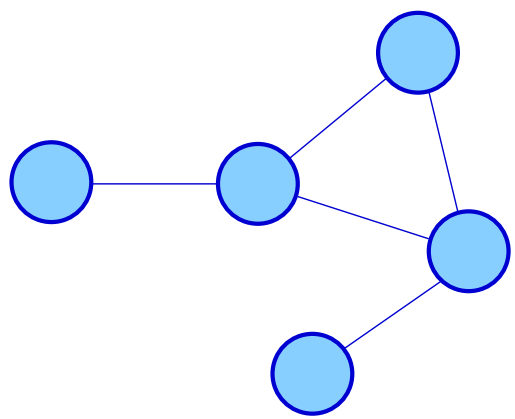
The problem

$f_1, \ldots, f_N, g_1, \ldots, g_N$ convex - ∇f_n is L-lipschitz continuous $\forall n$

$$\min_{u \in \mathcal{X}} \sum_{n=1}^{N} (f_n(u) + g_n(u))$$

Context:

- An agent $n \in \{1, \ldots, N\}$ knows only f_n, g_n
- Agents communicate with neighbors to estimate a minimizer



A brief review

Incremental methods (Bertsekas'83, Nedic'01, Rabbat'04, Tsianos'14)

- An estimate x^k travels from node to node
- The active node at time k updates x^k using a local (sub)gradient

Diffusion-adaptation methods (Tsitsiklis'84, Nedic'09, Duchi'12, Bianchi'14)

- Each node performs a local gradient descent on their local estimate...
- ...merge the estimates with neighbors and iterate

Distributed ADMM (Schizas'08, Boyd'11, Ozdaglar'12, Iutzeler'13)

- Special instance of ADMM
- Works only if $\forall n$, $f_n = 0 \rightarrow$ it is *not* a 1st-order method
- So far, synchronous or partially synchronous

Reformulating the problem (1/2)

The initial problem

$$\min_{u \in \mathcal{X}} \sum_{n=1}^{N} f_n(u) + \sum_{n=1}^{N} g_n(u)$$

boils down to

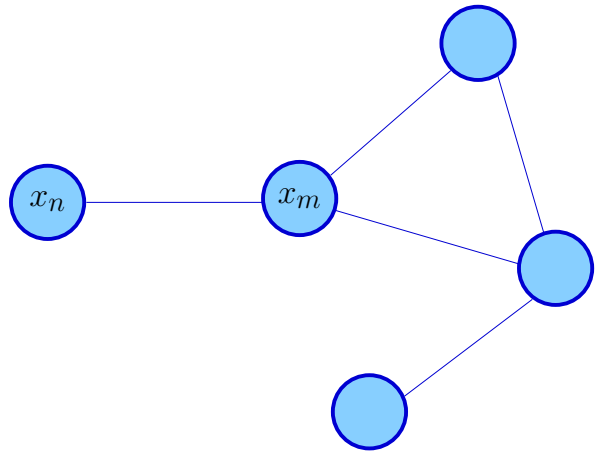
$$\min_{x \in \mathcal{X}^N} \underbrace{\sum_{n=1}^{N} f_n(x_n)}_{f(x)} + \underbrace{\sum_{n=1}^{N} g_n(x_n)}_{g(x)} + \iota_C(x)$$

Sum of 3 functions: ADMM+ can be applied...

...but the resulting algorithm will not be distributed

We need to introduce the communication graph

Reformulating the problem (2/2)



Global consensus is equivalent to consensus on the edges:

$$x_1 = \cdots = x_N \;\Leftrightarrow\; \forall n \sim m,\; x_n = x_m$$

Setting $C_2 = \{(u, u) : u \in \mathcal{X}\}$,

$$\iota_C(x) = \sum_{n \sim m} \iota_{C_2}(x_n, x_m)$$

Finally,

$$\min_{x \in \mathcal{X}^N} \underbrace{\sum_{n=1}^N f_n(x_n)}_{f(x)} + \underbrace{\sum_{n=1}^N g_n(x_n)}_{g(x)} + \underbrace{\sum_{n \sim m} \iota_{C_2}(x_n, x_m)}_{h(Mx)}$$

Applying ADMM+

Variables physically handled by node n : x_n^k and $(\lambda_{\{n,m\}}^k(n))_{m:n\sim m}$

Algorithm: At iteration k, do for each node n

- Obtain x_m^k from neighbors $m \sim n$
- Update dual variables

$$\lambda_{\{n,m\}}^{k+1}(n) = \lambda_{\{n,m\}}^k(n) + \frac{x_n^k - x_m^k}{2\rho}$$

- Update primal variable

$$x_n^{k+1} = \mathrm{prox}_{\tau g_n/d_n}\left[(1-\tau\rho^{-1})x_n^k - \frac{\tau}{d_n}\nabla f_n(x_n^k) + \frac{\tau}{d_n}\sum_{m:n\sim m}(\rho^{-1}x_m^k - \lambda_{\{n,m\}}^k(n))\right]$$

Using coordinate descent

From (Condat'13) we can write (up to a slight change of variable)

$$\begin{pmatrix} x^{k+1} \\ \lambda^{k+1} \end{pmatrix} = T \begin{pmatrix} x^{k} \\ \lambda^{k} \end{pmatrix} \tag{2}$$

where T is α-averaged

Apply coordinate descent:

- At iteration k, select a node n at random
- Update the variables handled by node n according to (2)

$$\zeta_n^{k+1} = (x_n^{k+1}, (\lambda_{\{n,m\}}^{k+1}(n))_{m:n\sim m})$$

- Keep other variables unchanged

$$\zeta_j^{k+1} = \zeta_j^k \quad \forall j \neq n$$

Distributed asynchronous algorithm

At iteration k, one node n is active and does:

- Update the dual variables

$$\lambda_{\{n,m\}}^{k+1}(n) = \frac{\lambda_{\{n,m\}}^{k}(n) - \lambda_{\{n,m\}}^{k}(m)}{2} + \frac{x_n^k - x_m^k}{2\rho},$$

- Update the primal variable

$$x_n^{k+1} = \text{prox}_{\tau g_n / d_n}\left[(1 - \tau\rho^{-1})x_n^k - \frac{\tau}{d_n}\nabla f_n(x_n^k) + \frac{\tau}{d_n}\sum_{m\sim n}\left(\frac{x_m^k}{\rho} + \lambda_{\{n,m\}}^{k}(m)\right)\right]$$

- Push $\{x_n^{k+1}, \lambda_{\{n,m\}}^{k+1}(n)\}$ to neighbor m

Other nodes $j \neq n$, maintain the variables

$$x_j^{k+1} = x_j^k \quad \text{and} \quad \lambda_{\{j,i\}}^{k+1}(j) = \lambda_{\{j,i\}}^{k}(j)$$

Convergence result

Denote by n^{k+1} the active node at iteration k

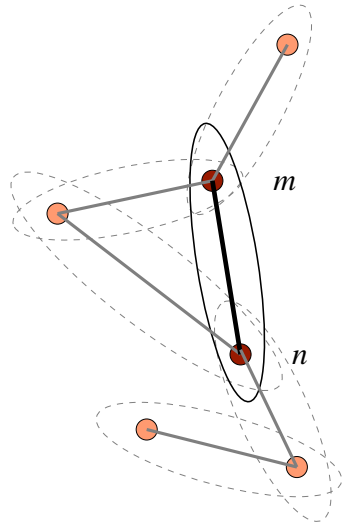
Assume:

- The sequence $n^1, n^2, \dots$ is iid
- $\mathbb{P}(n^1 = j) > 0$ for all $j = 1 \dots N$
- $\tau^{-1} - \rho^{-1} > L/2$

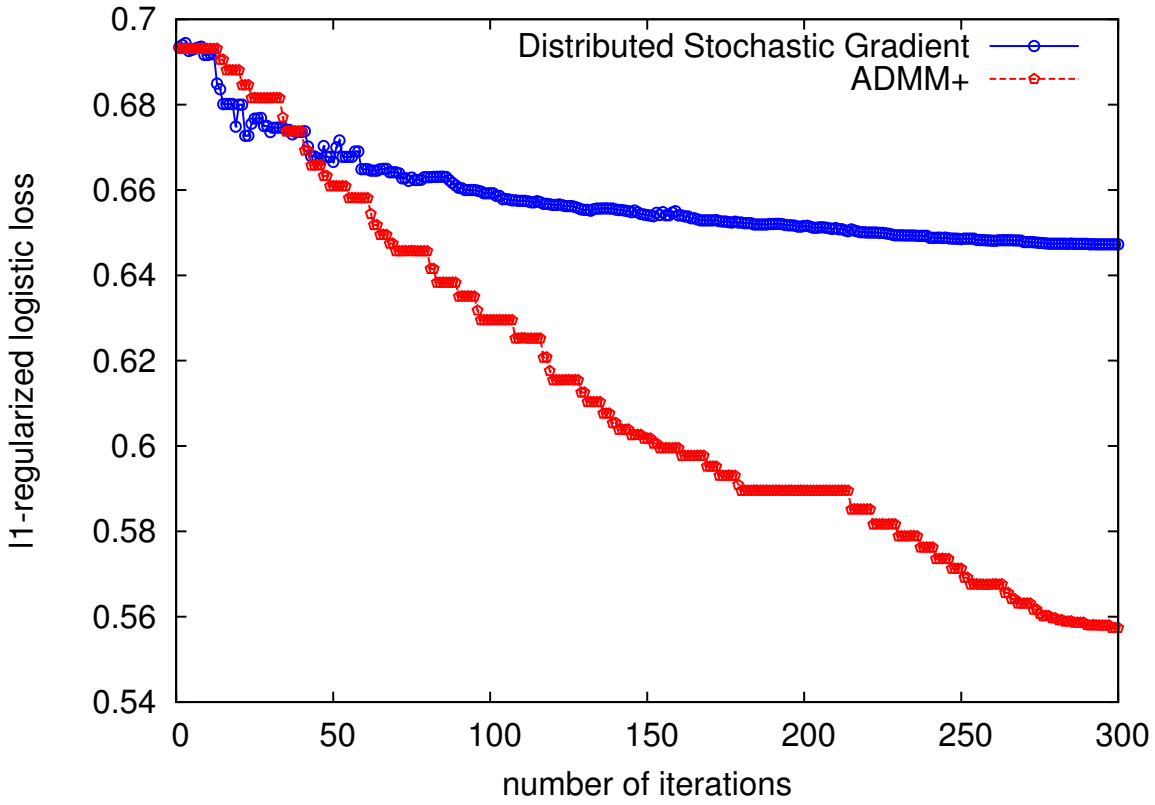
Result (Bianchi'14): Almost surely, there exists $x^* \in \arg\min \sum_n f_n + g_n$ s.t.

$$\forall n, \quad x_n^k \to x^* \quad \text{as} \quad k \to \infty$$

Numerical illustration



Context: Logistic regression + ℓ_1 regularization



Conclusions

- Convergence of a random coordinate descent scheme
- Application to a primal dual method
- Yields an asynchronous distributed optimization algorithm

Further question: convergence rates