First-order algorithms for large-scale convex optimization

Javier Peña
Carnegie Mellon University

Midwest Optimization Meeting & Workshop on Large Scale Optimization and Applications

University of Toronto & Fields Institute
Toronto, October 2011

Some applications of convex optimization

Nash equilibria computation

$$\min_{x \in Q_1} \max_{y \in Q_2} \langle x, Ay \rangle$$

Regularized linear regression

$$\min_{\beta} \left(\frac{1}{2} \|X\beta - y\|^2 + \Omega(\beta) \right)$$

$\Omega(\beta)$: regularization term, e.g., $\lambda \|\beta\|_1$ in lasso regression.

Compressed sensing

$$\begin{aligned} \min_{x} \quad & \|x\|_1 \\ & Ax = b \end{aligned}$$

In these applications:

- The convex optimization model has nice structure.
- Interesting practical instances lead to immense problems.
- In some cases the relevant data of the problem is not available in explicit form at once.
- These pose interesting computational challenges.

Outline

1. First-order schemes for convex optimization
2. Nash equilibria computation for large sequential games
3. Elementary algorithms for linear programming
4. A sparsity-preserving stochastic gradient algorithm

1. First-order schemes for convex optimization

Classical approaches for convex minimization:

- Subgradient methods (first-order non-smooth)
- **Gradient-descent methods (first-order smooth)**
- Newton's method (second-order smooth)

These are iterative schemes to solve

$$\begin{array}{ll} \min & f(x) \\ & x \in Q \end{array}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $Q \subseteq \mathbb{R}^n$ are convex.

Notation: $f^* := \min_x f(x)$.

Gradient methods

Consider

$$\min \quad f(x)$$
$$\quad\quad x \in Q,$$

where f is convex, smooth, and ∇f is L-Lipschitz on Q.

Gradient-descent scheme

- pick $x_0 \in Q$
- for $k = 0, 1, \ldots$

  $$x_{k+1} := \underset{y \in Q}{\operatorname{argmin}} \left\{ f(x_k) + \langle \nabla f(x_k), y - x_k \rangle + \frac{L}{2} \|y - x_k\|^2 \right\}$$

  end for

Gradient methods

Properties of gradient-descent scheme

- When $Q = \mathbb{R}^n$ we get the familiar

$$x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k).$$

- Convergence rate: after k iterations

$$f(x_k) - f^* = \mathcal{O}(L/k).$$

  Equivalently, we can find ϵ-solution in $\mathcal{O}(L/\epsilon)$ iterations.

- Each iteration involves a projection of the form

$$\min_{u \in Q} \left\{ \langle g, u \rangle + \frac{1}{2}\|u\|^2 \right\}.$$

Accelerated gradient-descent methods

Nesterov's accelerated scheme

- pick $x_0 = y_0 \in Q$
- for $k = 0, 1, \ldots$

$$x_{k+1} := \operatorname*{argmin}_{y \in Q} \left\{ f(y_k) + \langle \nabla f(y_k), y - y_k \rangle + \frac{L}{2}\|y - y_k\|^2 \right\}$$

$$y_{k+1} := x_{k+1} + \frac{k}{k+3}(x_{k+1} - x_k)$$

end for

Theorem (Nesterov 1983)

Above method has rate of convergence: $f(x_k) - f^* = \mathcal{O}(L/k^2)$. *Equivalently, we can find* ϵ*-solution in* $\mathcal{O}(\sqrt{L/\epsilon})$ *iterations.*

Remark

The $\mathcal{O}(1/\sqrt{\epsilon})$ complexity is optimal (Nemirovskii and Yudin 1983).

Nesterov's smoothing technique

Consider the saddle-point problem

$$\min_{x \in Q_1} \max_{y \in Q_2} \langle Ax, y \rangle.$$

This problem can be rewritten as

$$\begin{array}{ll} \min & f(x) \\ & x \in Q_1, \end{array}$$

where

$$f(x) = \max_{y \in Q_2} \langle Ax, y \rangle.$$

Typically f is non-smooth.

Nesterov's smoothing technique

Suppose we can easily compute projections

$$\min_{u \in Q_i} \left\{ \langle g, u \rangle + \frac{1}{2} \|u\|^2 \right\}$$

for $i = 1, 2$.

Nesterov's smoothing technique

1. Fix $y_0 \in Q_2$. For $\mu > 0$ define

$$f_\mu(x) := \max_{y \in Q_2} \left\{ \langle Ax, y \rangle - \frac{\mu}{2} \|y - y_0\|^2 \right\}.$$

2. Apply optimal gradient scheme to $\min_{x \in Q_1} f_\mu(x)$.

Nesterov's smoothing technique

Assume Q_1, Q_2 bounded and let

$$D_1 = \max_{x \in Q_1} \frac{1}{2}\|x - x_0\|^2, \quad D_2 = \max_{y \in Q_2} \frac{1}{2}\|y - y_0\|^2.$$

Theorem (Nesterov 2005)

By setting $\mu := \frac{2\|A\|}{\epsilon}\sqrt{\frac{D_1}{D_2}}$, the above smoothing scheme finds an ϵ-solution to the saddle point problem

$$\min_{x \in Q_1} \max_{y \in Q_2} \langle Ax, y \rangle$$

in

$$\mathcal{O}\left(\frac{\|A\|\sqrt{D_1 D_2}}{\epsilon}\right)$$

first-order iterations.

Nesterov's smoothing technique

Assume Q_1, Q_2 bounded and let

$$D_1 = \max_{x \in Q_1} \frac{1}{2}\|x - x_0\|^2, \quad D_2 = \max_{y \in Q_2} \frac{1}{2}\|y - y_0\|^2.$$

Theorem (Nesterov 2005)

By setting $\mu := \frac{2\|A\|}{\epsilon}\sqrt{\frac{D_1}{D_2}}$, the above smoothing scheme finds an ϵ-solution to the saddle point problem

$$\min_{x \in Q_1} \max_{y \in Q_2} \langle Ax, y \rangle$$

in

$$\mathcal{O}\left(\frac{\|A\|\sqrt{D_1 D_2}}{\epsilon}\right)$$

first-order iterations.

Optimal complexity of a subgradient method is $\mathcal{O}(1/\epsilon^2)$ (Nemirovskii-Yudin 1983).

Nesterov's smoothing technique

Theorem (Gilpin, P, Sandholm 2009)

If Q_1, Q_2 are polyhedral, then an iterated version of Nesterov's smoothing scheme finds an ϵ-solution to the saddle point problem

$$\min_{x \in Q_1} \max_{y \in Q_2} \langle Ax, y \rangle$$

in

$$\mathcal{O}\left(\kappa(A, Q_1, Q_2) \log\left(\frac{\|A\|\sqrt{D_1 D_2}}{\epsilon}\right)\right)$$

first-order iterations.

$\kappa(A, Q_1, Q_2)$: *"condition number" of A, Q_1, Q_2.*

Prox-functions and Bregman projection

Definition

$d : Q \to \mathbb{R}$ is a *prox-function* if

- d is strongly convex in Q, i.e., there exists $\sigma > 0$ such that for all $x, y \in Q$, and $\alpha \in [0, 1]$

$$d(\alpha x+(1-\alpha)y) \leq \alpha d(x)+(1-\alpha)d(y)-\frac{1}{2}\sigma\alpha(1-\alpha)\|x-y\|^2.$$

- $\min_{x \in Q} d(x) = 0$

Bregman distance

$$\xi(y, x) := d(y) - d(x) - \langle \nabla d(x), y - x\rangle \geq \frac{\sigma}{2}\|y - x\|^2.$$

Bregman projection

Use $\min_{y \in Q} \left\{\langle g, y\rangle + \frac{1}{\sigma}\xi(y, x)\right\}$ instead of $\min_{y \in Q} \left\{\langle g, y\rangle + \frac{1}{2}\|y - x\|^2\right\}$.

Other accelerated first-order schemes

- Nemirovskii's mirror-prox method
- Beck and Teboulle's FISTA algorithm
- Other classes of problems, e.g., variational inequalities, composite optimization

2. Nash equilibria computation for large sequential games

(joint work with A. Gilpin, S. Hoda, and T. Sandholm)

Sequential games

Games that involve turn-taking, chance moves, and imperfect information.

2. Nash equilibria computation for large sequential games

(joint work with A. Gilpin, S. Hoda, and T. Sandholm)

Sequential games

Games that involve turn-taking, chance moves, and imperfect information.

Example (r-round poker)

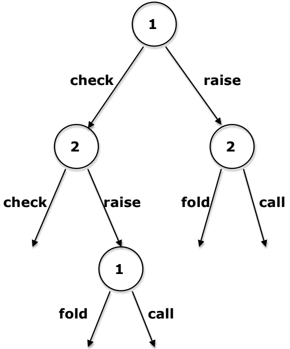
- Deal private cards to the players (face down)
- Betting round
- For $i = 2$ to r
    - Deal public cards (face up)
    - Betting round
- Showdown (if needed)

Example (one-round poker, detailed)

- Initial pot: $1 each
- **Deal:** deck with two Js and two Qs
  Deal one private card to each of two players

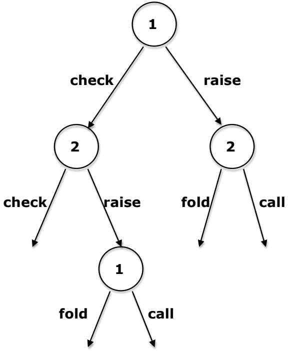
Example (one-round poker, detailed)

- Initial pot: \$1 each
- **Deal:** deck with two Js and two Qs
  Deal one private card to each of two players
- **Betting round:**



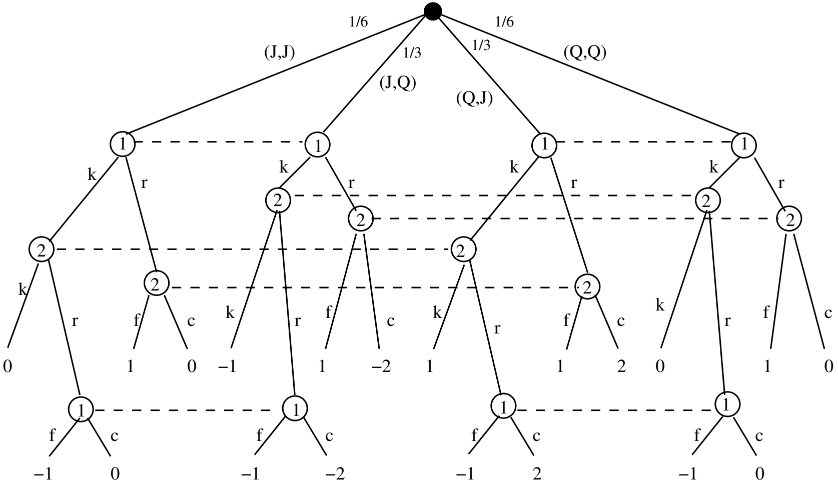
Example (one-round poker, detailed)

- Initial pot: \$1 each
- **Deal:** deck with two Js and two Qs
  Deal one private card to each of two players
- **Betting round:**



- If none of the players folded, player with higher card wins pot.

One-round poker in extensive form (game tree)



Nash equilibrium of sequential games

Simultaneous choice of strategies for all players so that no player has incentive to deviate.

Nash equilibrium of sequential games

Simultaneous choice of strategies for all players so that no player has incentive to deviate.

Formulation via the sequence form for two-person, zero-sum games (Von Stengel, Koller & Megiddo, Romanovskii)

$$\max_{x \in Q_1} \min_{y \in Q_2} \langle x, Ay \rangle = \min_{y \in Q_2} \max_{x \in Q_1} \langle x, Ay \rangle.$$

Nash equilibrium of sequential games

Simultaneous choice of strategies for all players so that no player has incentive to deviate.

Formulation via the sequence form for two-person, zero-sum games (Von Stengel, Koller & Megiddo, Romanovskii)

$$\max_{x \in Q_1} \min_{y \in Q_2} \langle x, Ay \rangle = \min_{y \in Q_2} \max_{x \in Q_1} \langle x, Ay \rangle.$$

- A: Player 1's payoff matrix
- Rows and columns of A indexed by **sequences** of moves of Players 1 and 2 respectively.
- Q_1, Q_2: strategy sets (realization plans) of Players 1 and 2 respectively
- Games in normal form: Q_1, Q_2 are simplexes.
- **Sequential** games in extensive form: Q_1, Q_2 are **treeplexes**.

Treeplexes

Definition

- A simplex is a treeplex.
- If $Q_1, \ldots, Q_k$ treeplexes then

$$\{(u^0, u^1, \ldots, u^k) : u^0 \in \Delta_k, u^i \in u_i^0 \cdot Q_i, i = 1, \ldots, k\}$$

  is a treeplex.
- If $Q_1, \ldots, Q_k$ treeplexes then $Q_1 \times \cdots \times Q_k$ is a treeplex.

Observe

A treeplex can be written in the form

$$\{u \in \mathbb{R}^d | u \geq 0, Eu = e\}$$

where E, e have $\{0, 1\}$ entries.

Computation of Nash equilibrium

Nash equilibrium

$$\max_{x \in Q_1} \min_{y \in Q_2} \langle x, Ay \rangle = \min_{y \in Q_2} \max_{x \in Q_1} \langle x, Ay \rangle.$$

Can formulate as the primal-dual pair of linear programs.
However, interesting games lead to enormous instances.

Computation of Nash equilibrium

Nash equilibrium

$$\max_{x \in Q_1} \min_{y \in Q_2} \langle x, Ay \rangle = \min_{y \in Q_2} \max_{x \in Q_1} \langle x, Ay \rangle.$$

Can formulate as the primal-dual pair of linear programs.
However, interesting games lead to enormous instances.

Poker

- Texas Hold'em (with limits): Game tree has $\sim 10^{18}$ nodes.
- Rhode Island Hold'em: simplification of Texas Hold'em.
  Created for AI research (Shi & Littman 2001).
  Game tree has $\sim 10^9$ nodes.
- These problems are too large for general-purpose linear programming solvers.
- Use Nesterov's smoothing approach.

Computation of Nash equilibrium

Theorem (Gilpin, Hoda, P, Sandholm 2007)

Assume $Q \subseteq \mathbb{R}^n$ is a treeplex. We can construct a prox-function $d : Q \to \mathbb{R}$ so that the projection $\min\{\langle g, u\rangle + d(u) : u \in Q\}$ is easily computable.

Theorem (Gilpin, Hoda, P, Sandholm 2007)

The new prox-functions yield a first-order smoothing algorithm that finds $(\bar{x}, \bar{y}) \in Q_1 \times Q_2$ such that

$$0 \leq \max_{x \in Q_1} \langle x, A\bar{y}\rangle - \min_{y \in Q_2} \langle \bar{x}, Ay\rangle \leq \epsilon$$

in

$$\left\lfloor 4 n_1 n_2 \frac{\|A\|}{\epsilon} \right\rfloor$$

first-order iterations.

n_i: number of sequences of Player i for $i = 1, 2$

Application to poker

Poker

- Central problem in artificial intelligence
- Unlike chess or checkers, it is a game of imperfect information
- Bluffing and other deceptive strategies are necessary to be a good player.
- Developing automatic poker players is an important milestone in artificial intelligence.

Game-theoretic approach to designing poker players

Limit Texas Hold'em

- Main version of poker used in academic research
- Game tree has about 10^{18} nodes.
- Use a sophisticated *abstraction* technique to create smaller games that approximate the original game
    - Compute approximate Nash equilibria for the abstractions
    - Recover approximate Nash equilibria for the original game
- Main current limitation of this approach: size of the abstractions that can be handled

Computational experience

Instances

- Lossy and lossless abstraction of Rhode Island Hold'em
- Lossy abstractions of Texas Hold'em

Problem sizes (when formulated as LPs)

Name	Rows	Columns	Nonzeros
10k	14,590	14,590	536,502
160k	226,074	226,074	9,238,993
RI	1,237,238	1,237,238	50,428,638
Texas	18,536,842	18,536,852	61,498,656,400
GS4	299,477,082	299,477,102	4,105,365,178,571

Implementation

Main work per iteration

- (Most expensive) matrix-vector products $x \mapsto A^{\mathsf{T}}x, \; y \mapsto Ay$
- Projections $\min_{u \in Q_i} \{\langle g, u\rangle + d(u)\}$.

Peculiar structure in poker instances

- Payoff matrix in poker games admits a concise representation. For example, for a three-round game

$$A = \begin{bmatrix} F_1 \otimes B_1 & & \\ & F_2 \otimes B_2 & \\ & & F_3 \otimes B_3 + S \otimes W \end{bmatrix}$$

- Do not need to form A explicitly.
- Instead have subroutines that compute $x \mapsto A^{\mathsf{T}}x, \; y \mapsto Ay$.

Do we get useful strategies?

- Annual AAAI Computer poker Competition (since 2006).
- About 15 teams competed Texas Hold'em with limits and with no limits.
- Players based on our algorithm are quite competitive (ended between first and fourth place).
- Unlike other players, these players do not use poker-specific expert knowledge.

3. Elementary algorithms for linear programming

(joint work with N. Soheili)

Assume

$A = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix} \in \mathbb{R}^{m \times n}$, where $\|a_j\| = 1,\ j = 1, \ldots, n.$

The perceptron algorithm solves

$$A^\mathsf{T} y > 0.$$

Perceptron Algorithm (Rosenblatt, 1958)

- $y_0 := 0$
- for $k = 0, 1, \ldots$
  $a_j^\mathsf{T} y_k := \min_i a_i^\mathsf{T} y_k$
  $y_{k+1} := y_k + a_j$
  end for

Normalized Perceptron Algorithm

Observe

$$a_j^{\mathsf{T}} y := \min_i a_i^{\mathsf{T}} y \Leftrightarrow a_j = Ax(y),\; x(y) = \operatorname*{argmin}_{x \in \Delta_n} \langle A^{\mathsf{T}} y, x \rangle.$$

Hence in the perceptron algorithm $y_k = Ax_k$ where $x_k \geq 0, \;\; \|x_k\|_1 = k$.

Normalized Perceptron Algorithm

- $y_0 := 0$
- for $k = 0, 1, \ldots$
  $\theta_k := \frac{1}{k+1}$
  $y_{k+1} := (1 - \theta_k) y_k + \theta_k A x(y_k)$

  end for

In this algorithm $y_k = Ax_k$ for $x_k \in \Delta_n$.

The Von Neumann Algorithm

Algorithm to solve

$$Ax = 0, \; x \in \Delta_n. \tag{1}$$

Von Neumann Algorithm, 1948

- $x_0 := \frac{1}{n}\mathbf{1}$; $y_0 := Ax_0$
- For $k = 0, 1, \ldots$

  if $v_k := \min_i a_i^T y_k > 0$ then STOP; (1) is infeasible

  $\lambda_k := \frac{\|y_k\|^2 - v_k}{\|y_k\|^2 - 2v_k + 1}$

  $x_{k+1} := (1-\lambda_k)x_k + \lambda_k x(y_k)$

  $y_{k+1} := (1-\lambda_k)y_k + \lambda_k Ax(y_k)$

  end for

The Von Neumann Algorithm

Algorithm to solve

$$Ax = 0,\ x \in \Delta_n. \tag{1}$$

Von Neumann Algorithm, 1948

- $x_0 := \frac{1}{n}\mathbf{1}$; $y_0 := Ax_0$
- For $k = 0, 1, \ldots$

  if $v_k := \min_i a_i^{\mathsf{T}} y_k > 0$ then STOP; (1) is infeasible

  $\lambda_k := \frac{\|y_k\|^2 - v_k}{\|y_k\|^2 - 2v_k + 1}$

  $x_{k+1} := (1-\lambda_k)x_k + \lambda_k x(y_k)$

  $y_{k+1} := (1-\lambda_k)y_k + \lambda_k Ax(y_k)$

  end for

Main loop in the normalized perceptron:

$\theta_k := \frac{1}{k+1}$

$x_{k+1} := (1-\theta_k)x_k + \theta_k x(y_k)$

$y_{k+1} := (1-\theta_k)y_k + \theta_k Ax(y_k)$

Properties of Perceptron and Von Neumann Algorithms

Recall assumption: $A = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix} \in \mathbb{R}^{m \times n}$, $\|a_j\| = 1$, $j = 1, \ldots, n$.

Perceptron and Von Neumann Algorithms

- Simple greedy iterations
- Convergence analysis in terms of the parameter

$$\rho(A) = \max_{||y||=1} \min_i a_i^\mathsf{T} y = \max_{||y||=1} \min_{x \in \Delta_n} \langle A^\mathsf{T} y, x \rangle.$$

Observe:

- $\rho(A) > 0$ if and only if $A^\mathsf{T} y > 0$ feasible
- $\rho(A) \leq 0$ if and only if $Ax = 0$, $x \in \Delta_n$ feasible

Properties of Perceptron and Von Neumann Algorithms

Theorem (Block, Novikoff 1962)

If $\rho(A) > 0$ then the perceptron finds a solution to $A^T y > 0$ in at most

$$\frac{1}{\rho(A)^2}$$

iterations.

Properties of Perceptron and Von Neumann Algorithms

Theorem (Block, Novikoff 1962)

If $\rho(A) > 0$ *then the perceptron finds a solution to* $A^T y > 0$ *in at most*

$$\frac{1}{\rho(A)^2}$$

iterations.

Theorem (Epelman & Freund, 2000)

If $\rho(A) < 0$ *then then the Von Neumann Algorithm finds an* ϵ*-solution to* $Ax = 0,\ x \in \Delta_n$ *in at most*

$$\frac{1}{\rho(A)^2} \cdot \log\left(\frac{1}{\epsilon}\right)$$

iterations.

Smooth Perceptron and Von Neumann Algorithms

Theorem (Soheili & P 2011)

If $\rho(A) > 0$, then a Smooth Perceptron Algorithm finds a solution to $A^T y > 0$ in at most

$$\frac{2\sqrt{\log(n)}}{\rho(A)}$$

iterations while preserving the simplicity of the perceptron.

Theorem (Soheili & P 2011)

If $\rho(A) < 0$, then a Smooth Von Neumann Algorithm finds an ϵ-solution to $Ax = 0,\ x \in \Delta_n$ in at most

$$\frac{2\sqrt{\log(n)}}{|\rho(A)|} \log\left(\frac{1}{\epsilon}\right)$$

iterations while preserving the simplicity of Von Neumann Algorithm.

Perceptron and Von Neumann as first-order algorithms

Observation

The perceptron and Von Neumann algorithms are respectively subgradient and gradient schemes for the saddle-point problems

$$\max_{\|y\|\leq 1} \min_{x\in\Delta_n} \langle y, Ax\rangle = \min_{x\in\Delta_n} \max_{\|y\|\leq 1} \langle y, Ax\rangle.$$

Smooth perceptron and Von Neumann

Use a smooth version of

$$x(y) = \operatorname*{argmin}_{x\in\Delta_n} \langle A^{\mathsf{T}} y, x\rangle,$$

namely,

$$x_\mu(y) := \frac{\exp(-A^{\mathsf{T}} y/\mu)}{\|\exp(-A^{\mathsf{T}} y/\mu)\|_1}$$

for some $\mu > 0$.

Smooth Perceptron Algorithm

Smooth Perceptron Algorithm

- $y_0 := \frac{1}{n}A\mathbf{1}$; $\mu_0 := 1$; $x_0 := x_{\mu_0}(y_0)$
- for $k = 0, 1, \ldots$

  $\theta_k := \frac{2}{k+3}$

  $y_{k+1} := (1-\theta_k)(y_k + \theta_k A x_k) + \theta_k^2 A x_{\mu_k}(y_k)$

  $\mu_{k+1} := (1-\theta_k)\mu_k$

  $x_{k+1} := (1-\theta_k)x_k + \theta_k x_{\mu_{k+1}}(y_{k+1})$

  end for

Smooth Perceptron Algorithm

Smooth Perceptron Algorithm

- $y_0 := \frac{1}{n} A\mathbf{1}$; $\mu_0 := 1$; $x_0 := x_{\mu_0}(y_0)$
- for $k = 0, 1, \ldots$

$$\begin{aligned}
&\theta_k := \tfrac{2}{k+3} \\
&y_{k+1} := (1-\theta_k)(y_k + \theta_k A x_k) + \theta_k^2 A x_{\mu_k}(y_k) \\
&\mu_{k+1} := (1-\theta_k)\mu_k \\
&x_{k+1} := (1-\theta_k)x_k + \theta_k x_{\mu_{k+1}}(y_{k+1})
\end{aligned}$$

  end for

Main loop in the normalized perceptron:

$$\begin{aligned}
&\theta_k := \tfrac{1}{k+1} \\
&x_{k+1} := (1-\theta_k)x_k + \theta_k x(y_k) \\
&y_{k+1} := (1-\theta_k)y_k + \theta_k A x(y_k)
\end{aligned}$$

4. A sparsity-preserving stochastic gradient algorithm

(joint work with Q. Lin and X. Chen)

Consider the convex optimization problem

$$\begin{array}{ll} \min & f(x) + h(x) \\ & x \in Q, \end{array}$$

where f is smooth of the form

$$f(x) = \mathbb{E}[F(x, \omega)],$$

and h is a non-smooth, sparsity enforcing, e.g., $h(x) = \|x\|_1$.

4. A sparsity-preserving stochastic gradient algorithm

(joint work with Q. Lin and X. Chen)

Consider the convex optimization problem

$$\begin{array}{ll} \min & f(x) + h(x) \\ & x \in Q, \end{array}$$

where f is smooth of the form

$$f(x) = \mathbb{E}[F(x, \omega)],$$

and h is a non-smooth, sparsity enforcing, e.g., $h(x) = \|x\|_1$.

For instance, lasso regression

$$\min_{\beta} \left(\frac{1}{2} \|X\beta - y\|^2 + \|\beta\|_1 \right)$$

Beck and Teboulle's FISTA algorithm

Assume f convex, ∇f is L-Lipschitz.

FISTA Algorithm

- pick $x_0 = y_0 \in Q$; $t_0 := 1$
- for $k = 0, 1, \ldots$

$$x_{k+1} := \operatorname*{argmin}_{y \in Q} \left\{ f(y_k) + \langle \nabla f(y_k), y - y_k \rangle + \tfrac{L}{2} \|y - y_k\|^2 + h(y) \right\}$$

$$t_{k+1} := \frac{1 + \sqrt{1 + 4t_k^2}}{2}$$

$$y_{k+1} := x_{k+1} + \frac{t_k - 1}{t_{k+1}} (x_{k+1} - x_k)$$

end for

Theorem (Beck and Teboulle 2008)

After k iterations $(f + h)(x_k) - (f + h)^* = \mathcal{O}(1/k^2)$.

Stochastic gradient methods

Challenge

Gradient $\nabla f(x)$ may be expensive or impossible to compute. We may only have a *stochastic gradient* $\nabla F(x, \omega)$ such that

$$\nabla f(x) = \mathbb{E}[\nabla F(x, \omega)].$$

Stochastic gradient

Estimate $\mathbb{E}[\nabla F(x, \omega)]$: Draw a sample $S = \{\omega_1, \ldots, \omega_K\}$ and compute

$$G(x, S) = \frac{1}{K} \sum_{i=1}^{K} \nabla_x F(x, \omega_i)$$

Assumption

$\mathbb{E} G(x, S) = \nabla f(x)$ and $\mathbb{E}\| G(x, S) - \nabla f(x)\|^2 \leq \sigma^2$.

Stochastic gradient methods

Algorithms that use $G(x, S)$ instead of $\nabla f(x)$

- Stochastic Approximation:
  Robbins and Morron 1951, Polyak and Juditsky 1992
- Mirror Descent Stochastic Approximation:
  Nemirovski et al. 1983, 2009
- Accelerated Stochastic Approximation (AC-SA):
  Lan 2010, Ghadimi and Lan 2010
- (Accelerated) Regularized Dual Average (RDA):
  Xiao 2010
- Others: Langford et al 2009, Shalev-Shwartz 2009, Hu 2009,
  Duchi 2009, Byrd et al. 2011

Most of these have optimal convergence rate $\mathcal{O}(1/\epsilon^2)$
(Nemirovskii and Yudin 1983).

The issue of sparsity

Solution to

$$\begin{array}{ll} \min & f(x) + h(x) \\ & x \in Q, \end{array}$$

is sparse thanks to $h(x)$.

Observe

- The sparsity of the optimal solution is due to the form of the objective, not the specific algorithm.
- The iterates generated by an algorithm may not be as sparse as the optimal solution, particularly for first-order algorithms that converge slowly.

The issue of sparsity

AC-SA (Lan 2010)

- Choose $x_0 = z_0 \in Q$
- For $t = 1, 2, \ldots$

  $y_t = (1 - \alpha_t) z_{t-1} + \alpha_t x_{t-1}$

  draw a sample S_t

  $x_t = \underset{x \in Q}{\operatorname{argmin}} \{ \langle G(y_t, S_t), x \rangle + \frac{\gamma_t}{2\alpha_t} \|x - x_{t-1}\|^2 + h(x) \}$

  $z_t = (1 - \alpha_t) z_{t-1} + \alpha_t x_t$

  end for
- Output: z_t

In this algorithm

- Iterate x_t is sparse while z_t is not.
- Iterate z_t converges to the optimal solution while x_t does not.
- Similar situation in other stochastic gradient methods.
- We would like to make the sparse x_t converge to optimality.

The issue of sparsity

Sparsity-preserving Stochastic Gradient(SSG)

- Choose $\gamma_0 > 0$ and $x_0 = y_0 \in Q$.
- For $t = 0, 1, 2, \dots$

  Choose $L_t > L,\ \alpha_t \in (0,1)$ s.t. $2L_t\alpha_t^2 \le (1-\alpha_t)\gamma_t =: \gamma_{t+1}$

  $y_t := \frac{\alpha_t\gamma_t z_t + \gamma_{t+1}x_t}{\gamma_t}$

  draw a sample S_t

  $x_{t+1} := \operatorname*{argmin}_{x\in Q}\{\langle G(y_t, S_t), x\rangle + \frac{L_t}{2}\|x - y_t\|^2 + h(x)\}$

  $z_{t+1} := \frac{(1-\alpha_t)\gamma_t z_t - \alpha_t L_t(y_t - x_{t+1})}{\gamma_{t+1}}$
- Output x_t

x_t is sparse and converges to the optimal solution.

Sparse solution

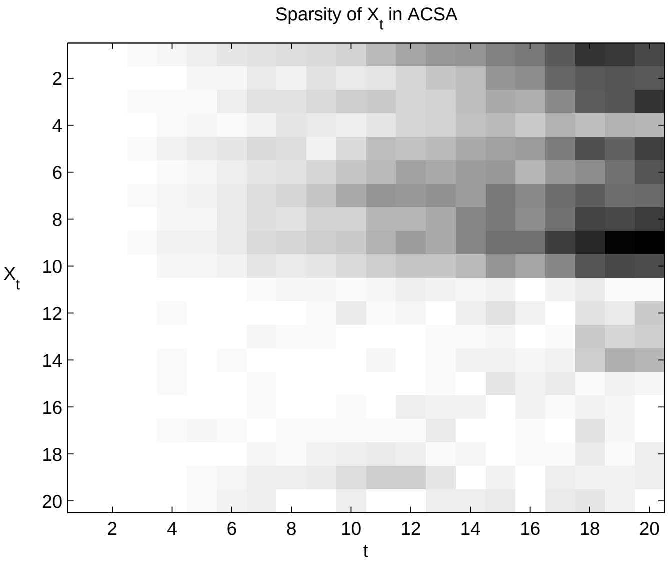
Example

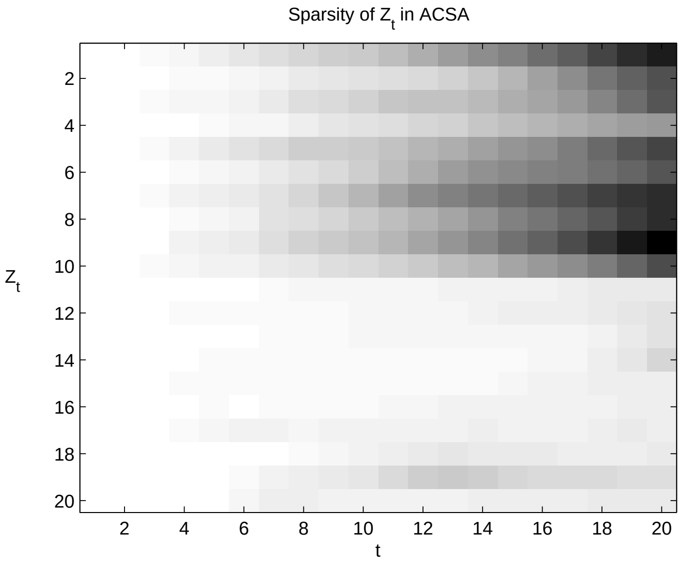
$$\min_x \frac{1}{2}\mathbb{E}(a^T x - b)^2 + \lambda\|x\|_1$$

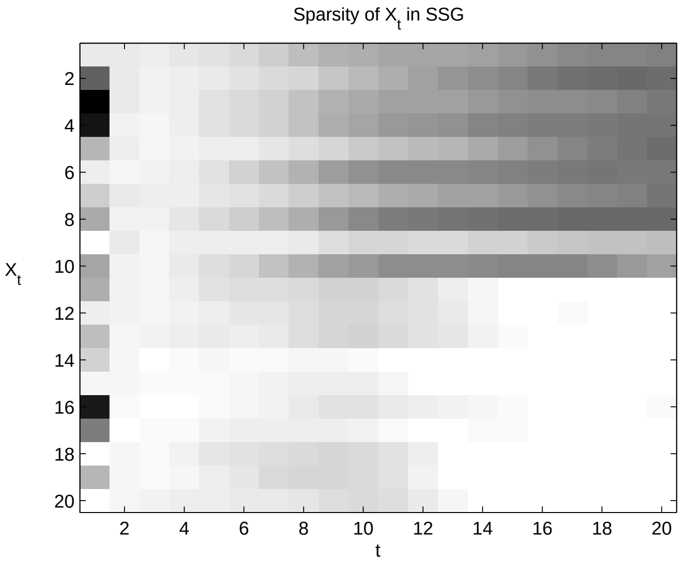
$b = a^T \bar{x} + \epsilon$

$\bar{x} = (1, 1, \ldots, 1, 0, 0, \ldots, 0)^T$

$a_i \leftarrow U(0, 1)$ and $\epsilon \leftarrow N(0, 1)$


Sparsity of X_t in ACSA
X_t
2
4
6
8
10
12
14
16
18
20
t
2
4
6
8
10
12
14
16
18
20


Sparsity of Z_t in ACSA
Z_t
2
4
6
8
10
12
14
16
18
20
t
2
4
6
8
10
12
14
16
18
20


Sparsity of X_t in SSG
X_t
t
2
4
6
8
10
12
14
16
18
20

Properties of SSG Algorithm

Observe

Each iterate x_t is random since each $G(y_t, S_t)$ is random.

Theorem (Qihang-Chen-P 2011)

If $\mathbb{E}\|G(x,S) - \nabla f(x)\|^2 \leq C < \infty$ *then for suitable chosen* γ_t, α_t, L_t *we have*

$$\mathbb{E}\left[\phi(x_{t+1}) - \phi(x^*)\right] \leq \mathcal{O}\left(\frac{C}{\sqrt{t}}\right).$$

Rate of converge $\mathcal{O}\left(\frac{1}{\sqrt{t}}\right)$ is optimal (Nemirovski and Yudin, 1983)

Theorem (Qihang-Chen-P 2011)

If $\mathbb{E}\|G(x,S) - \nabla f(x)\|^4 \leq C$ *and* $\|x^* - x_t\|, \ \|x^* - z_t\| \leq D$ *then*

$$\mathbb{V}\left[\phi(x_{t+1}) - \phi(x^*)\right] \leq \mathcal{O}\left(\frac{CD^2}{t}\right).$$

Concluding remarks

- Numerous interesting applications in computational game theory, signal processing, machine learning can be modeled as convex optimization problems.
- Practical problems are typically immense. This poses major computational challenges.
- Modern algorithmic technology (accelerated gradient methods) can be specialized to deal effectively with these challenges.

References

- S. Hoda, J. A. Gilpin, and J. Peña, and T. Sandholm, "Smoothing techniques for computing Nash equilibria of sequential games," Mathematics of Operations Research 35 (2010) pp. 494–512.
- A. Gilpin, J. Peña, and T. Sandholm, "First-order algorithm with $\mathcal{O}(\log(1/\epsilon))$ convergence for ϵ-equilibrium in two-person zero-sum games," To Appear in Mathematical Programming.
- N. Soheili and J Peña, "A smooth perceptron algorithm," Technical report, Carnegie Mellon University.
- Q. Lin, X. Chen, and J. Peña "A sparsity preserving stochastic gradient method for composite optimization," Technical report, Carnegie Mellon University.