Opening up the Black Box: Fast Non-Smooth and Big-Data Optimization

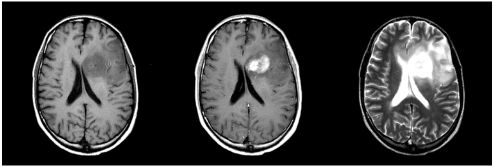
Mark Schmidt

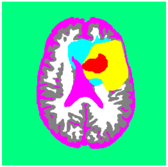
Natural Language Laboratory
School of Computing Science
Simon Fraser University

February 2014

Motivation: Automatic Brain Tumor Segmentation

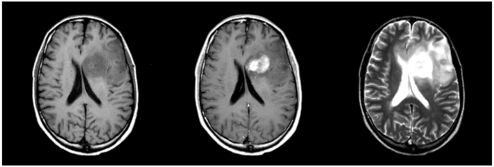
- Task: Segmentation of Multi-Modality MRI Data

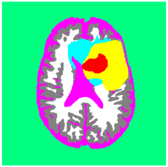




Motivation: Automatic Brain Tumor Segmentation

- Task: Segmentation of Multi-Modality MRI Data

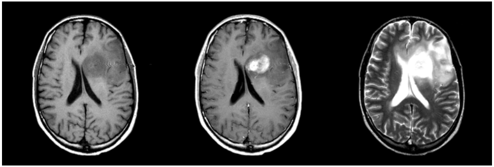


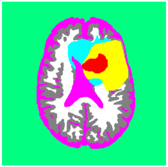


- Various applications:
  - radiation therapy target planning.
  - quantifying growth or treatment response.
  - image-guided surgery.

Motivation: Automatic Brain Tumor Segmentation

- Task: Segmentation of Multi-Modality MRI Data

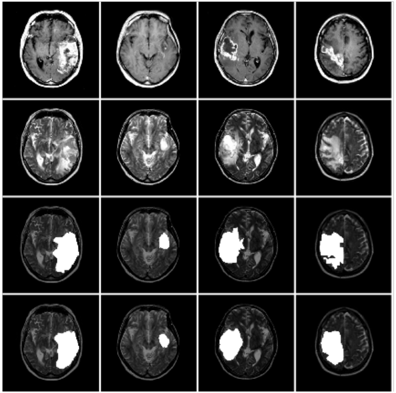




- Various applications:
  - radiation therapy target planning.
  - quantifying growth or treatment response.
  - image-guided surgery.
- Challenges:
  - image noise and intensity inhomogeneity.
  - similarity between tumor and normal tissue.

Motivation: Automatic Brain Tumor Segmentation

- Solution strategy:
  - Explicit correction of image inhomogeneities.
  - Spatial alignment with template.
  - Image and template-based features.
  - Pixel-level classifier.



Motivation: Automatic Brain Tumor Segmentation

- Best performance with logistic regression:

$$\min_x \sum_{i=1}^{N} f_i(x).$$

Motivation: Automatic Brain Tumor Segmentation

- Best performance with logistic regression:

$$\min_x \sum_{i=1}^{N} f_i(x).$$

- Problem 1: Estimating x is slow:
  - 8 million voxels per volume.
  - Last part of talk: Big-N problems.

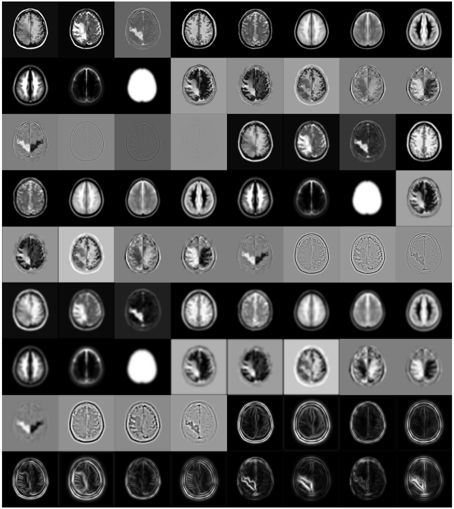
Motivation: Automatic Brain Tumor Segmentation

- Best performance with logistic regression:

$$\min_x \sum_{i=1}^{N} f_i(x).$$

- Problem 1: Estimating x is slow:
  - 8 million voxels per volume.
  - Last part of talk: Big-N problems.
- Problem 2: Designing features.
  - Lots of possible candidate features.
  - Using all features leads to over-fitting.
  - First part of talk: Feature Selection.

Motivation: Automatic Brain Tumor Segmentation



- Training time is too slow for automatic feature selection:
  - forced to use manual feature selection

Optimizing with ℓ_1-Regularzation

- Last day of Master's: try **all features** with ℓ_2-Regularization:

$$\min_x f(x) + \lambda \|x\|^2.$$

  - Reduces over-fitting.
  - As good as best selected features.
  - But, very slow to segment new image.

Optimizing with ℓ_1-Regularzation

- Last day of Master's: try **all features** with ℓ_2-Regularization:

$$\min_x f(x) + \lambda \|x\|^2.$$

  - Reduces over-fitting.
  - As good as best selected features.
  - But, very slow to segment new image.

- Reading on way to Ph.D.: **all features** with ℓ_1-Regularization:

$$\min_x f(x) + \lambda \|x\|_1.$$

  - Still reduces over-fitting.
  - But, solution x is SPARSE (some $x_j = 0$).
  - Feature selection by only training once.

Optimizing with ℓ_1-Regularzation

- Last day of Master's: try **all features** with ℓ_2-Regularization:

$$\min_x f(x) + \lambda \|x\|^2.$$

  - Reduces over-fitting.
  - As good as best selected features.
  - But, very slow to segment new image.

- Reading on way to Ph.D.: **all features** with ℓ_1-Regularization:

$$\min_x f(x) + \lambda \|x\|_1.$$

  - Still reduces over-fitting.
  - But, solution x is SPARSE (some $x_j = 0$).
  - Feature selection by only training once.

- Amazing! But non-smooth, how do we solve this problem?

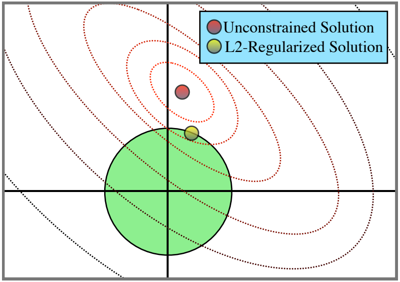
Where does the sparsity come from?

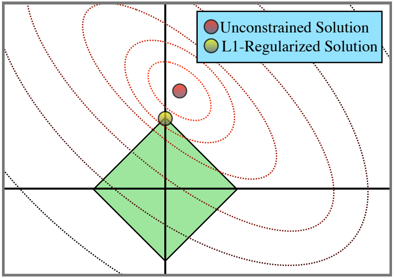
- We can re-write the regularized problem

$$\min_x f(x) + \lambda\|x\|_p$$

as a constrained problem

$$\min_{\|x\|_p \leq \tau} f(x).$$





Where does the sparsity come from?

- Consider our problem

$$\min_x F(x) = f(x) + r(x).$$

Where does the sparsity come from?

- Consider our problem

$$\min_x F(x) = f(x) + r(x).$$

- When F is convex and smooth,
  its minimizer x^* has gradient $F'(x^*) = 0$.

Where does the sparsity come from?

- Consider our problem

$$\min_x F(x) = f(x) + r(x).$$

- When F is convex and smooth,

  its minimizer x^* has gradient $F'(x^*) = 0$.

  When $r(x) = \lambda\|x\|^2$:

  - We need $f'(x) = -\lambda x$.

Where does the sparsity come from?

- Consider our problem

$$\min_x F(x) = f(x) + r(x).$$

- When F is convex and smooth,

  its minimizer x^* has gradient $F'(x^*) = 0$.

  When $r(x) = \lambda\|x\|^2$:

  - We need $f'(x) = -\lambda x$.

- When F is convex ~~and smooth~~,

  its minimizer x^* has a subgradient $d = 0$.

Where does the sparsity come from?

- Consider our problem

$$\min_x F(x) = f(x) + r(x).$$

- When F is convex and smooth,

  its minimizer x^* has gradient $F'(x^*) = 0$.

  When $r(x) = \lambda\|x\|^2$:

  - We need $f'(x) = -\lambda x$.

- When F is convex ~~and smooth~~,

  its minimizer x^* has a subgradient $d = 0$.

  When $r(x) = \lambda\|x\|_1$:

  - We need $f'(x) = -\lambda d$, for some sub-gradient d of $\|x\|_1$.

Where does the sparsity come from?

- Consider our problem

$$\min_x F(x) = f(x) + r(x).$$

- When F is convex and smooth,

  its minimizer x^* has gradient $F'(x^*) = 0$.

  When $r(x) = \lambda\|x\|^2$:

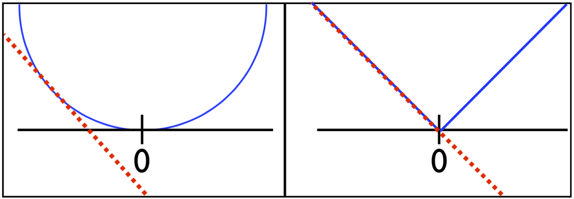
  - We need $f'(x) = -\lambda x$.

- When F is convex ~~and smooth~~,

  its minimizer x^* has a subgradient $d = 0$.

  When $r(x) = \lambda\|x\|_1$:

  - We need $f'(x) = -\lambda d$, for some sub-gradient d of $\|x\|_1$.



Where does the sparsity come from?

- Consider our problem

$$\min_x F(x) = f(x) + r(x).$$

- When F is convex and smooth,

  its minimizer x^* has gradient $F'(x^*) = 0$.

  When $r(x) = \lambda\|x\|^2$:

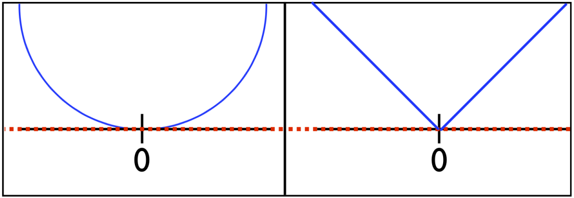
  - We need $f'(x) = -\lambda x$.

- When F is convex ~~and smooth~~,

  its minimizer x^* has a subgradient $d = 0$.

  When $r(x) = \lambda\|x\|_1$:

  - We need $f'(x) = -\lambda d$, for some sub-gradient d of $\|x\|_1$.



Where does the sparsity come from?

- Consider our problem

$$\min_x F(x) = f(x) + r(x).$$

- When F is convex and smooth,

  its minimizer x^* has gradient $F'(x^*) = 0$.

  When $r(x) = \lambda\|x\|^2$:

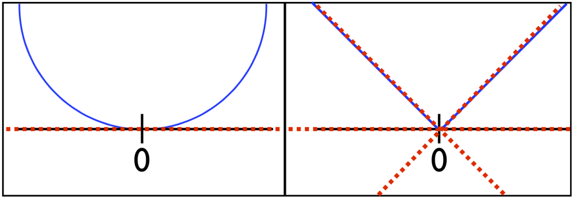
  - We need $f'(x) = -\lambda x$.

- When F is convex ~~and smooth~~,

  its minimizer x^* has a subgradient $d = 0$.

  When $r(x) = \lambda\|x\|_1$:

  - We need $f'(x) = -\lambda d$, for some sub-gradient d of $\|x\|_1$.



Where does the sparsity come from?

- Consider our problem

$$\min_x F(x) = f(x) + r(x).$$

- When F is convex and smooth,

  its minimizer x^* has gradient $F'(x^*) = 0$.

  When $r(x) = \lambda \|x\|^2$:

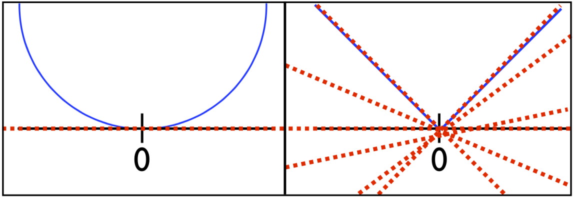
  - We need $f'(x) = -\lambda x$.

- When F is convex ~~and smooth~~,

  its minimizer x^* has a subgradient $d = 0$.

  When $r(x) = \lambda \|x\|_1$:

  - We need $f'(x) = -\lambda d$, for some sub-gradient d of $\|x\|_1$.



Optimization with ℓ_1-Regularization

- We want to optimize a smooth function with ℓ_1-Regularization:

$$\min_x f(x) + \lambda \|x\|_1.$$

Optimization with ℓ_1-Regularization

- We want to optimize a smooth function with ℓ_1-Regularization:

$$\min_x f(x) + \lambda \|x\|_1.$$

- With ℓ_2-Regularization, can use quasi-Newton methods.

  http://www.di.ens.fr/~mschmidt/Software/minFunc.html

Optimization with ℓ_1-Regularization

- We want to optimize a smooth function with ℓ_1-Regularization:

$$\min_x f(x) + \lambda \|x\|_1.$$

- With ℓ_2-Regularization, can use quasi-Newton methods.

  http://www.di.ens.fr/~mschmidt/Software/minFunc.html

- The non-smooth ℓ_1-regularizer breaks these methods.

Optimization with ℓ_1-Regularization

- We want to optimize a smooth function with ℓ_1-Regularization:

$$\min_x f(x) + \lambda \|x\|_1.$$

- With ℓ_2-Regularization, can use quasi-Newton methods.

  `http://www.di.ens.fr/~mschmidt/Software/minFunc.html`

- The non-smooth ℓ_1-regularizer breaks these methods.
- But the regularizer is separable: $\|x\|_1 = \sum_j |x_j|$.

Optimization with ℓ_1-Regularization

- We want to optimize a smooth function with ℓ_1-Regularization:

$$\min_x f(x) + \lambda \|x\|_1.$$

- With ℓ_2-Regularization, can use quasi-Newton methods.

  http://www.di.ens.fr/~mschmidt/Software/minFunc.html

- The non-smooth ℓ_1-regularizer breaks these methods.
- But the regularizer is separable: $\|x\|_1 = \sum_j |x_j|$.
- Can we extend quasi-Newton methods using this property?

Converting to a Bound-Constrained Problem

- Consider splitting each variable into a positive and negative part:

$$x = x^+ - x^-, \text{ with } x^+ \geq 0, x^- \geq 0.$$

Converting to a Bound-Constrained Problem

- Consider splitting each variable into a positive and negative part:

$$x = x^+ - x^-, \text{ with } x^+ \geq 0, x^- \geq 0.$$

- We can re-write the non-smooth objective

$$\min_x f(x) + \lambda \|x\|_1,$$

as a smooth objective with non-negative constraints:

$$\min_{x^+ \geq 0, x^- \geq 0} F(x) = f(x^+ - x^-) + \lambda \sum_j [x_i^+ + x_i^-]$$

Converting to a Bound-Constrained Problem

- Consider splitting each variable into a positive and negative part:

$$x = x^+ - x^-, \text{ with } x^+ \geq 0, x^- \geq 0.$$

- We can re-write the non-smooth objective

$$\min_x f(x) + \lambda \|x\|_1,$$

as a smooth objective with non-negative constraints:

$$\min_{x^+ \geq 0, x^- \geq 0} F(x) = f(x^+ - x^-) + \lambda \sum_j [x_i^+ + x_i^-]$$

- Use methods for smooth bound-constrained optimization.

Gradient Projection

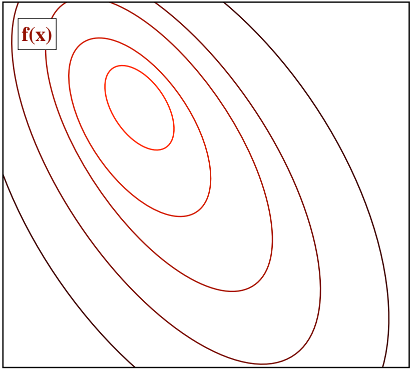
- Classic bound-constrained optimizer is gradient projection:

$$x^{k+1} \leftarrow [x^k - \alpha F'(x^k)]^+.$$

Gradient Projection

- Classic bound-constrained optimizer is gradient projection:

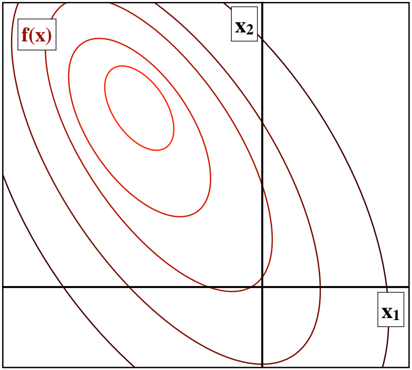
$$x^{k+1} \leftarrow [x^k - \alpha F'(x^k)]^+.$$



Gradient Projection

- Classic bound-constrained optimizer is gradient projection:

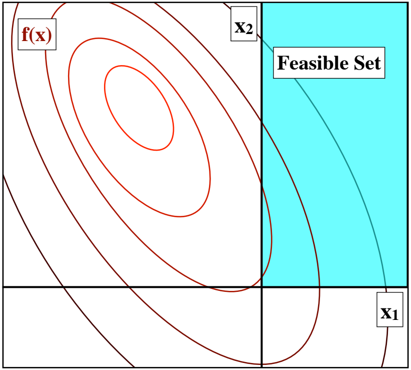
$$x^{k+1} \leftarrow [x^k - \alpha F'(x^k)]^+.$$



Gradient Projection

- Classic bound-constrained optimizer is gradient projection:

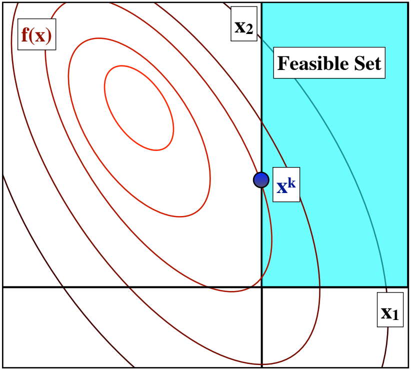
$$x^{k+1} \leftarrow [x^k - \alpha F'(x^k)]^+.$$



Gradient Projection

- Classic bound-constrained optimizer is gradient projection:

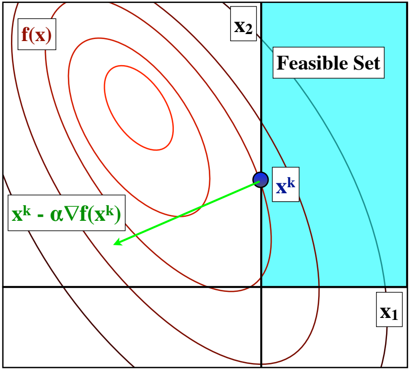
$$x^{k+1} \leftarrow [x^k - \alpha F'(x^k)]^+.$$



Gradient Projection

- Classic bound-constrained optimizer is gradient projection:

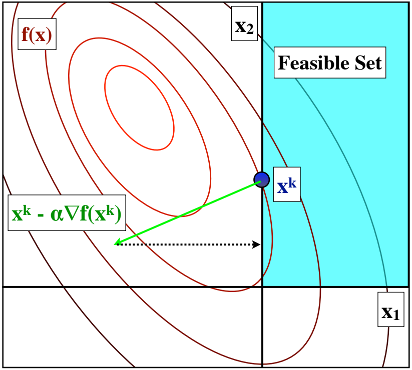
$$x^{k+1} \leftarrow [x^k - \alpha F'(x^k)]^+.$$



Gradient Projection

- Classic bound-constrained optimizer is gradient projection:

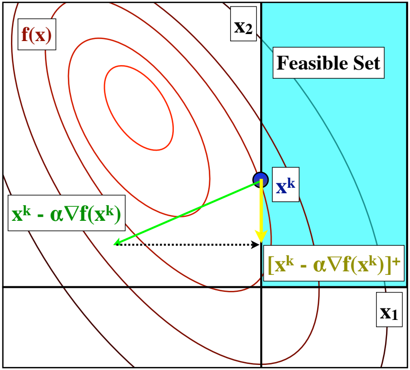
$$x^{k+1} \leftarrow [x^k - \alpha F'(x^k)]^+.$$



Gradient Projection

- Classic bound-constrained optimizer is gradient projection:

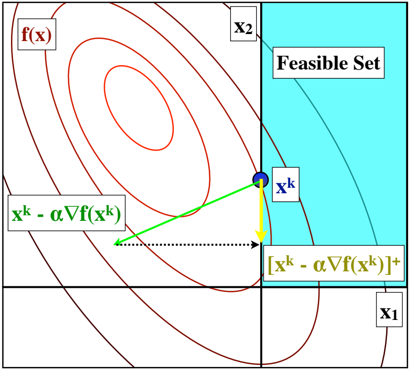
$$x^{k+1} \leftarrow [x^k - \alpha F'(x^k)]^+.$$



Gradient Projection

- Classic bound-constrained optimizer is gradient projection:

$$x^{k+1} \leftarrow [x^k - \alpha F'(x^k)]^+.$$



- Convergence properties similar to gradient method.

Naive Projected Newton Method

- Can we use a [quasi-]Newton step?

$$x^{k+1} \leftarrow [x^k - \alpha H_k^{-1} F'(x^k)]^+,$$

Naive Projected Newton Method

- Can we use a [quasi-]Newton step?

$$x^{k+1} \leftarrow [x^k - \alpha H_k^{-1} F'(x^k)]^+,$$

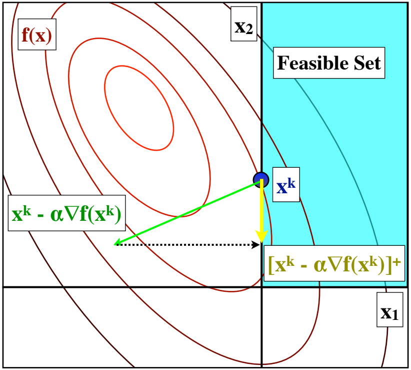
No, this does not work!

Naive Projected Newton Method

- Can we use a [quasi-]Newton step?

$$x^{k+1} \leftarrow [x^k - \alpha H_k^{-1} F'(x^k)]^+,$$

No, this does not work!

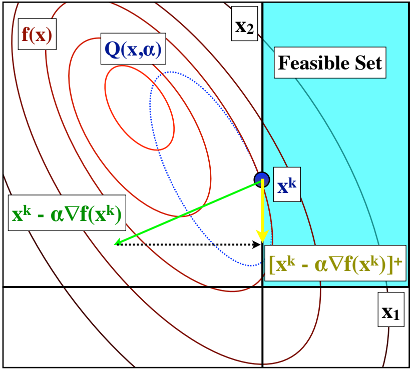


Naive Projected Newton Method

- Can we use a [quasi-]Newton step?

$$x^{k+1} \leftarrow [x^k - \alpha H_k^{-1} F'(x^k)]^+,$$

No, this does not work!

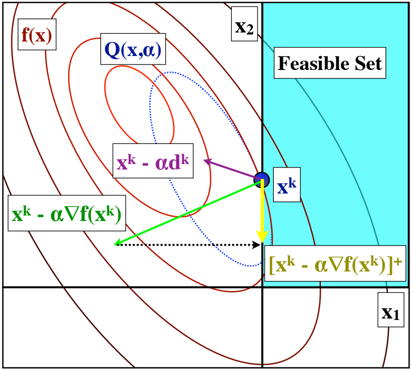


Naive Projected Newton Method

- Can we use a [quasi-]Newton step?

$$x^{k+1} \leftarrow [x^k - \alpha H_k^{-1} F'(x^k)]^+,$$

No, this does not work!

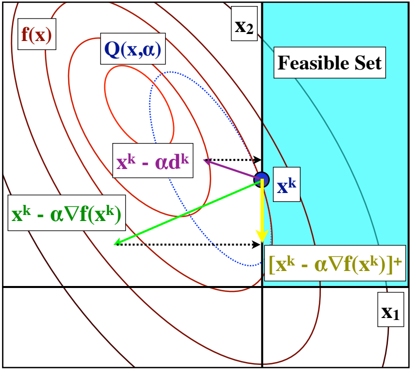


Naive Projected Newton Method

- Can we use a [quasi-]Newton step?

$$x^{k+1} \leftarrow [x^k - \alpha H_k^{-1} F'(x^k)]^+,$$

No, this does not work!

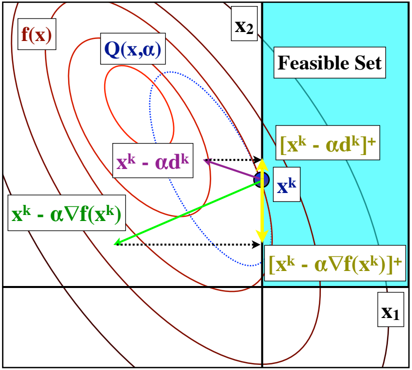


Naive Projected Newton Method

- Can we use a [quasi-]Newton step?

$$x^{k+1} \leftarrow [x^k - \alpha H_k^{-1} F'(x^k)]^+,$$

No, this does not work!



Partially Diagonal Two-Metric Projection

- For separable problems we can fix this by restricting H_k.

Partially Diagonal Two-Metric Projection

- For separable problems we can fix this by restricting H_k.
- Use a diagonal matrix D_k:

$$x^{k+1} \leftarrow [x^k - \alpha[D_k]^{-1} F'(x^k)]^+$$

[Birgin et al., 2000, Figueiredo et al., 2007]

Partially Diagonal Two-Metric Projection

- For separable problems we can fix this by restricting H_k.
- Use a diagonal matrix D_k:

$$x^{k+1} \leftarrow [x^k - \alpha [D_k]^{-1} F'(x^k)]^+$$

[Birgin et al., 2000, Figueiredo et al., 2007]

- But is this too restrictive?

Partially Diagonal Two-Metric Projection

- For separable problems we can fix this by restricting H_k.
- Use a diagonal matrix D_k:

$$x^{k+1} \leftarrow [x^k - \alpha[D_k]^{-1}F'(x^k)]^+$$

[Birgin et al., 2000, Figueiredo et al., 2007]

- But is this too restrictive?
- Only need H_k diagonal with respect to:

$$\mathcal{A} \triangleq \{i | x_i^k \leq \epsilon \text{ and } F_i'(x^k) > 0\}$$

[Gafni & Bertsekas, 1984]

Partially Diagonal Two-Metric Projection

- For separable problems we can fix this by restricting H_k.
- Use a diagonal matrix D_k:

$$x^{k+1} \leftarrow [x^k - \alpha[D_k]^{-1}F'(x^k)]^+$$

[Birgin et al., 2000, Figueiredo et al., 2007]

- But is this too restrictive?
- Only need H_k diagonal with respect to:

$$\mathcal{A} \triangleq \{i | x_i^k \leq \epsilon \text{ and } F_i'(x^k) > 0\}$$

[Gafni & Bertsekas, 1984]

- Re-arranging, we need

$$H_k = \begin{bmatrix} D_k & \mathbf{0} \\ \mathbf{0} & \bar{H}_k \end{bmatrix}$$

Partially Diagonal Two-Metric Projection

- For separable problems we can fix this by restricting H_k.
- Use a diagonal matrix D_k:

$$x^{k+1} \leftarrow [x^k - \alpha[D_k]^{-1} F'(x^k)]^+$$

[Birgin et al., 2000, Figueiredo et al., 2007]

- But is this too restrictive?
- Only need H_k diagonal with respect to:

$$\mathcal{A} \triangleq \{i | x_i^k \leq \epsilon \text{ and } F_i'(x^k) > 0\}$$

[Gafni & Bertsekas, 1984]

- Re-arranging, we need

$$H_k = \begin{bmatrix} D_k & \mathbf{0} \\ \mathbf{0} & \bar{H}_k \end{bmatrix}$$

- $\bar{H}_k$ can be quasi-Newton approximation of $F''(x^k)$.

Discussion of Two-Metric Projection

- Outperforms 11 other methods in Schmidt et al. [2007]:
  - Iterations only require linear time and space.
  - Many variables can be made zero/non-zero at once.
  - Allows warm-starting.
  - Eventually becomes quasi-Newton on the non-zeroes.

Discussion of Two-Metric Projection

- Outperforms 11 other methods in Schmidt et al. [2007]:
  - Iterations only require linear time and space.
  - Many variables can be made zero/non-zero at once.
  - Allows warm-starting.
  - Eventually becomes quasi-Newton on the non-zeroes.
- But should we convert to a bound-constrained problem?
  - The number of variables is doubled.
  - The transformed problem might be harder.

Discussion of Two-Metric Projection

- Outperforms 11 other methods in Schmidt et al. [2007]:
  - Iterations only require linear time and space.
  - Many variables can be made zero/non-zero at once.
  - Allows warm-starting.
  - Eventually becomes quasi-Newton on the non-zeroes.
- But should we convert to a bound-constrained problem?
  - The number of variables is doubled.
  - The transformed problem might be harder.
- Can we use the same tricks on the original problem?

Non-Smooth Steepest Descent

- The original problem:

$$\min_x F(x) = f(x) + \lambda \|x\|_1.$$

Non-Smooth Steepest Descent

- The original problem:

$$\min_x F(x) = f(x) + \lambda \|x\|_1.$$

- If f is smooth, F has directional derivatives everywhere.

Non-Smooth Steepest Descent

- The original problem:

$$\min_x F(x) = f(x) + \lambda\|x\|_1.$$

- If f is smooth, F has directional derivatives everywhere.
- We could use the steepest descent direction $-z^k$.

Non-Smooth Steepest Descent

- The original problem:

$$\min_x F(x) = f(x) + \lambda \|x\|_1.$$

- If f is smooth, F has directional derivatives everywhere.
- We could use the steepest descent direction $-z^k$.
- For convex problems, z^k is the minimum-norm sub-gradient:

$$z^k = \arg\min_{z \in \partial F(x^k)} ||z||$$

Non-Smooth Steepest Descent

- The steepest descent direction for ℓ_1-Regularization problems,

$$\min_x F(x) = f(x) + \lambda\|x\|_1,$$

can be computed coordinate-wise because $\|x\|_1$ is separable:

Non-Smooth Steepest Descent

- The steepest descent direction for ℓ_1-Regularization problems,

$$\min_x F(x) = f(x) + \lambda \|x\|_1,$$

can be computed coordinate-wise because $\|x\|_1$ is separable:

$$z_i = \begin{cases} F_i'(x) = f_i'(x) + \lambda \operatorname{sign}(x_i), & |x_i| > 0 \\ 0, & x_i = 0, |f_i'(x)| \leq \lambda \\ f_i'(x) - \lambda \operatorname{sign}(f_i'(x)), & x_i = 0, |f_i'(x)| > \lambda \end{cases}$$

Non-Smooth Steepest Descent

- The steepest descent direction for ℓ_1-Regularization problems,

$$\min_x F(x) = f(x) + \lambda \|x\|_1,$$

can be computed coordinate-wise because $\|x\|_1$ is separable:

$$z_i = \begin{cases} F_i'(x) = f_i'(x) + \lambda \operatorname{sign}(x_i), & |x_i| > 0 \\ 0, & x_i = 0, |f_i'(x)| \leq \lambda \\ f_i'(x) - \lambda \operatorname{sign}(f_i'(x)), & x_i = 0, |f_i'(x)| > \lambda \end{cases}$$

- We can even try a Newton-like version:

$$x^{k+1} = x^k - \alpha [H_k]^{-1} z^k$$

Non-Smooth Steepest Descent

- The steepest descent direction for ℓ_1-Regularization problems,

$$\min_x F(x) = f(x) + \lambda \|x\|_1,$$

can be computed coordinate-wise because $\|x\|_1$ is separable:

$$z_i = \begin{cases} F_i'(x) = f_i'(x) + \lambda\, \mathrm{sign}(x_i), & |x_i| > 0 \\ 0, & x_i = 0, |f_i'(x)| \leq \lambda \\ f_i'(x) - \lambda\, \mathrm{sign}(f_i'(x)), & x_i = 0, |f_i'(x)| > \lambda \end{cases}$$

- We can even try a Newton-like version:

$$x^{k+1} = x^k - \alpha [H_k]^{-1} z^k$$

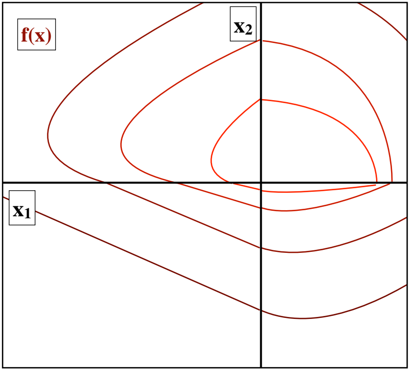
- However, there are two problems with this step:
  1. It may not decrease the objective.
  2. The iterations are not sparse.

Orthant Projection

- Use orthant projection to get sparse iterates:

$$x^{k+1} \leftarrow \mathcal{P}_{\mathcal{O}(x^k)}[x^k - \alpha[H_k]^{-1} z^k],$$

[Osborne et al., 2000, Andrew & Gao, 2007]

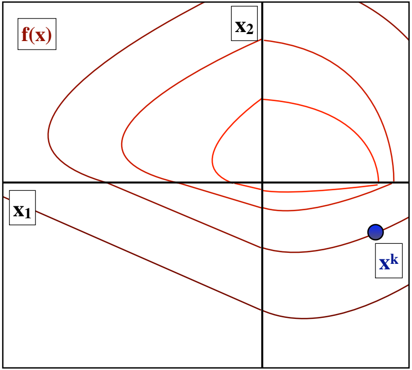


Orthant Projection

- Use orthant projection to get sparse iterates:

$$x^{k+1} \leftarrow \mathcal{P}_{\mathcal{O}(x^k)}[x^k - \alpha[H_k]^{-1}z^k],$$

[Osborne et al., 2000, Andrew & Gao, 2007]

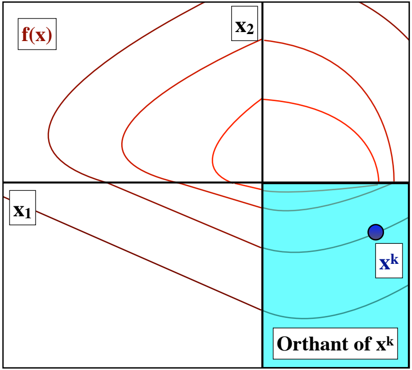


Orthant Projection

- Use orthant projection to get sparse iterates:

$$x^{k+1} \leftarrow \mathcal{P}_{\mathcal{O}(x^k)}[x^k - \alpha[H_k]^{-1} z^k],$$

[Osborne et al., 2000, Andrew & Gao, 2007]

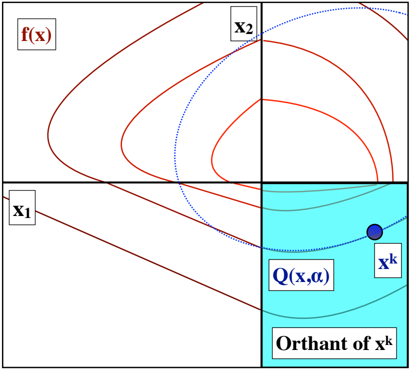


Orthant Projection

- Use orthant projection to get sparse iterates:

$$x^{k+1} \leftarrow \mathcal{P}_{\mathcal{O}(x^k)}[x^k - \alpha[H_k]^{-1}z^k],$$

[Osborne et al., 2000, Andrew & Gao, 2007]

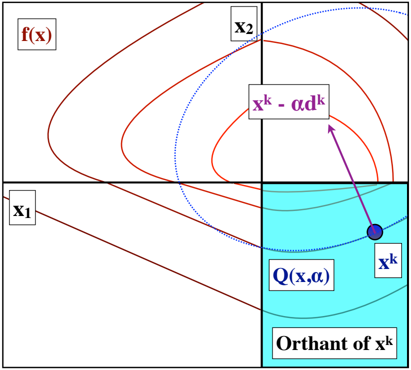


Orthant Projection

- Use orthant projection to get sparse iterates:

$$x^{k+1} \leftarrow \mathcal{P}_{\mathcal{O}(x^k)}[x^k - \alpha[H_k]^{-1} z^k],$$

[Osborne et al., 2000, Andrew & Gao, 2007]

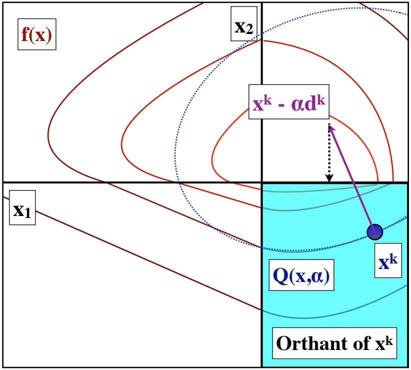


Orthant Projection

- Use orthant projection to get sparse iterates:

$$x^{k+1} \leftarrow \mathcal{P}_{\mathcal{O}(x^k)}[x^k - \alpha[H_k]^{-1} z^k],$$

[Osborne et al., 2000, Andrew & Gao, 2007]

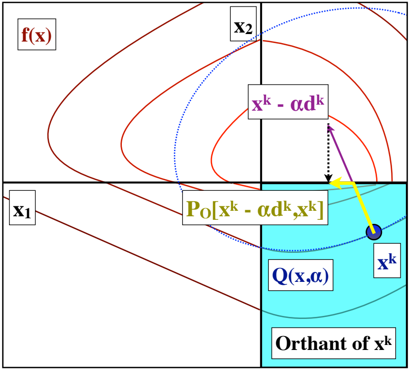


Orthant Projection

- Use orthant projection to get sparse iterates:

$$x^{k+1} \leftarrow \mathcal{P}_{\mathcal{O}(x^k)}[x^k - \alpha[H_k]^{-1} z^k],$$

[Osborne et al., 2000, Andrew & Gao, 2007]

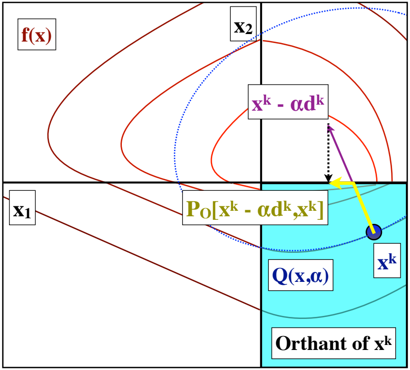


Orthant Projection

- Use orthant projection to get sparse iterates:

$$x^{k+1} \leftarrow \mathcal{P}_{\mathcal{O}(x^k)}[x^k - \alpha[H_k]^{-1}z^k],$$

[Osborne et al., 2000, Andrew & Gao, 2007]



- Variables that change sign become exactly zero.

Two-Metric Sub-Gradient Projection

- We can guarantee descent using diagonal scaling:

$$x^{k+1} \leftarrow \mathcal{P}_{\mathcal{O}(x^k)}[x^k - \alpha[D_k]^{-1}z^k].$$

Two-Metric Sub-Gradient Projection

- We can guarantee descent using diagonal scaling:

$$x^{k+1} \leftarrow \mathcal{P}_{\mathcal{O}(x^k)}[x^k - \alpha[D_k]^{-1}z^k].$$

- Less restrictive: diagonal with respect to variables near zero:

$$\mathcal{A} = \{i || x_i^k| \leq \epsilon\}, \quad \mathcal{F} = \{i || x_i^k| > \epsilon\}$$

Two-Metric Sub-Gradient Projection

- We can guarantee descent using diagonal scaling:

$$x^{k+1} \leftarrow \mathcal{P}_{\mathcal{O}(x^k)}[x^k - \alpha[D_k]^{-1} z^k].$$

- Less restrictive: diagonal with respect to variables near zero:

$$\mathcal{A} = \{i | |x_i^k| \leq \epsilon\}, \quad \mathcal{F} = \{i | |x_i^k| > \epsilon\}$$

- Two-metric sub-gradient projection:

$$x_{\mathcal{F}}^{k+1} \leftarrow \mathcal{P}_{\mathcal{O}(x_{\mathcal{F}}^k)}[x_{\mathcal{F}}^k - \alpha[H_k]^{-1} F'_{\mathcal{F}}(x^k)].$$

$$x_{\mathcal{A}}^{k+1} \leftarrow \mathcal{P}_{\mathcal{O}(x_{\mathcal{A}}^k)}[x_{\mathcal{A}}^k - \alpha[D_k]^{-1} z_{\mathcal{A}}^k],$$

Two-Metric Sub-Gradient Projection

- We can guarantee descent using diagonal scaling:

$$x^{k+1} \leftarrow \mathcal{P}_{\mathcal{O}(x^k)}[x^k - \alpha[D_k]^{-1} z^k].$$

- Less restrictive: diagonal with respect to variables near zero:

$$\mathcal{A} = \{i \| x_i^k| \leq \epsilon\}, \quad \mathcal{F} = \{i \| x_i^k| > \epsilon\}$$

- Two-metric sub-gradient projection:

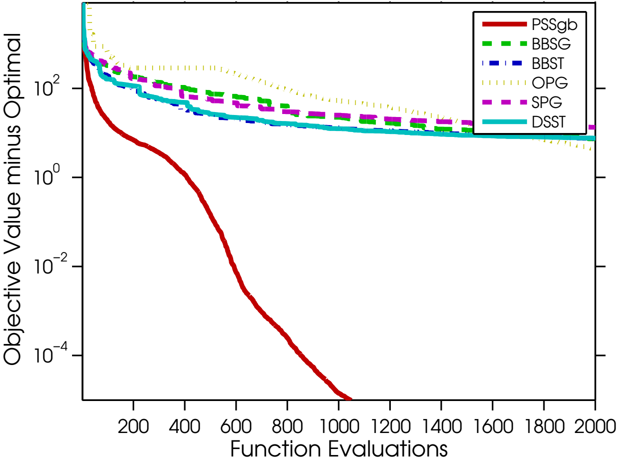
$$x_{\mathcal{F}}^{k+1} \leftarrow \mathcal{P}_{\mathcal{O}(x_{\mathcal{F}}^k)}[x_{\mathcal{F}}^k - \alpha[H_k]^{-1} F'_{\mathcal{F}}(x^k)].$$

$$x_{\mathcal{A}}^{k+1} \leftarrow \mathcal{P}_{\mathcal{O}(x_{\mathcal{A}}^k)}[x_{\mathcal{A}}^k - \alpha[D_k]^{-1} z_{\mathcal{A}}^k],$$

- Quasi-Newton method with separable non-smooth regularization.

Comparing to non-L-BFGS methods

Comparing to methods not based on L-BFGS (sido data):



Discussion

- Similar ideas used in many ℓ_1-Regularization solvers.

  [Perkins et al., 2003, Andrew & Gao, 2007, Shi et al., 2007, Kim & Park, 2010, Byrd et al., 2012].

Discussion

- Similar ideas used in many ℓ_1-Regularization solvers.

  [Perkins et al., 2003, Andrew & Gao, 2007, Shi et al., 2007, Kim & Park, 2010, Byrd et al., 2012].

- Recent methods consider two more issues:
  - Sub-Optimization: Identify variables likely to stay zero.

    [El Ghaoui et al., 2010].
  - Continuation: Start with a large λ and slowly decrease it.

    [Xiao and Zhang, 2012]

Discussion

- Similar ideas used in many ℓ_1-Regularization solvers.

  [Perkins et al., 2003, Andrew & Gao, 2007, Shi et al., 2007, Kim & Park, 2010, Byrd et al., 2012].

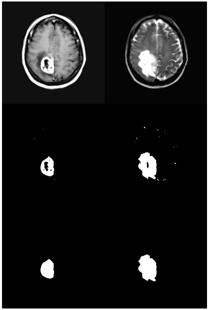
- Recent methods consider two more issues:
  - Sub-Optimization: Identify variables likely to stay zero.

    [El Ghaoui et al., 2010].
  - Continuation: Start with a large λ and slowly decrease it.

    [Xiao and Zhang, 2012]
- Generalizes to separable A.E.-differentiable regularizers.
- Exist two-metric projection for simplex constraints.

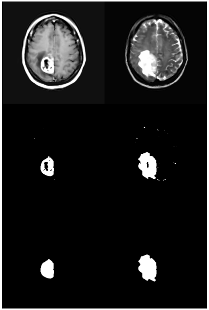
Motivation:Automatic Brain Tumor Segmentation

- Independent pixel classifier ignores correlations.
- Conditional random fields (CRFs) generalize logistic regression to multiple labels.



Motivation:Automatic Brain Tumor Segmentation

- Independent pixel classifier ignores correlations.
- Conditional random fields (CRFs) generalize logistic regression to multiple labels.



- Can use exact same optimizer for ℓ_1-regularized CRFs.

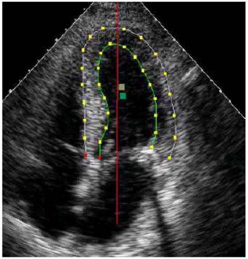
`http://www.di.ens.fr/~mschmidt/Software/L1General.html`

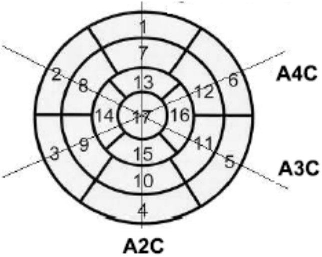
Outline

1. Sparsity
2. Group Sparsity
3. Structured Sparsity
4. Big-N Problems

Motivation: Structure Learning in CRFs

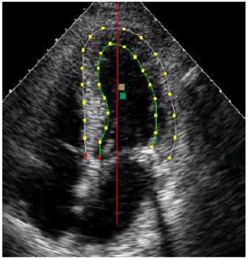
- Task: early detection of coronoary heart disease.

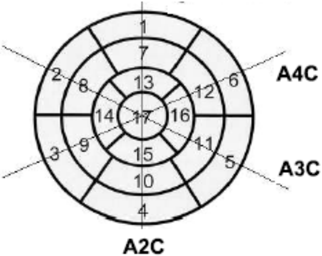




Motivation: Structure Learning in CRFs

- Task: early detection of coronoary heart disease.

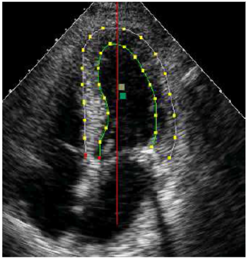


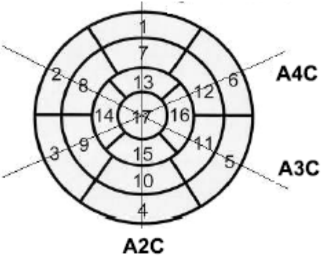


- Assess motion of 16 heart segments using CRF.
- But, do not know the best correlation structure.

Motivation: Structure Learning in CRFs

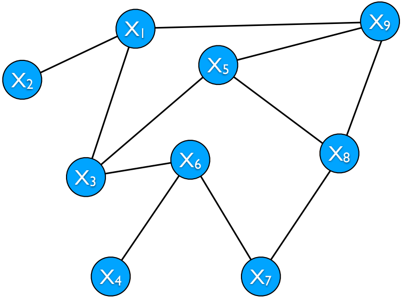
- Task: early detection of coronoary heart disease.





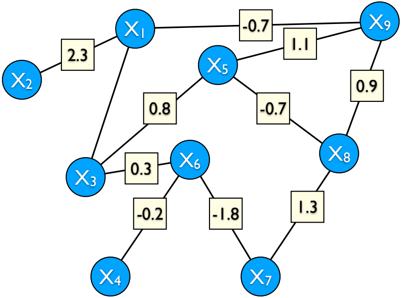
- Assess motion of 16 heart segments using CRF.
- But, do not know the best correlation structure.
- Perform structure learning with ℓ_1-regularization.

Structure Learning with ℓ_1-Regularization



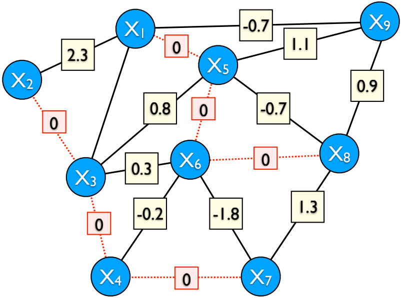
- We want to fit a Markov random field with unknown structure.

Structure Learning with ℓ_1-Regularization



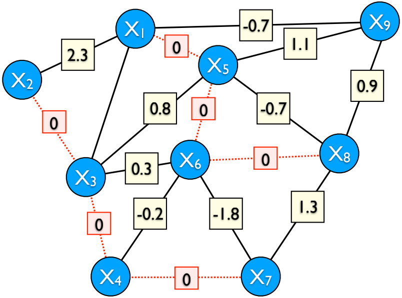
- We want to fit a Markov random field with unknown structure.

Structure Learning with ℓ_1-Regularization



- We want to fit a Markov random field with unknown structure.

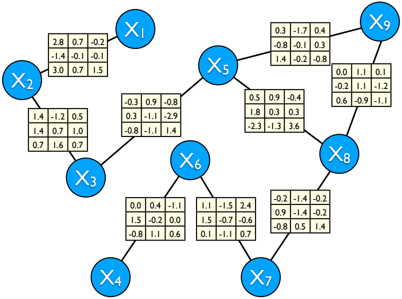
Structure Learning with ℓ_1-Regularization



- We want to fit a Markov random field with unknown structure.
- Learn a sparse structure by ℓ_1-regularization of edge weights.

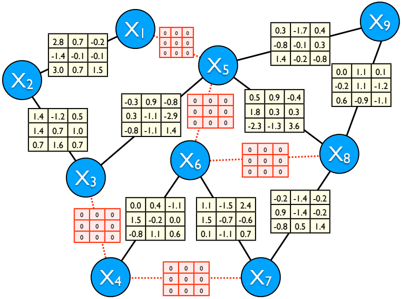
[Lee et al. 2006, Wainwright et al. 2006]

Structure Learning with Group ℓ_1-Regularization



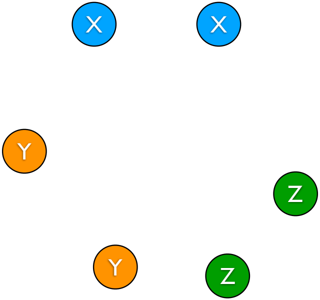
- In some cases, we want sparsity in groups of parameters:
  1. Multi-class variables [Lee et al., 2006].

Structure Learning with Group ℓ_1-Regularization



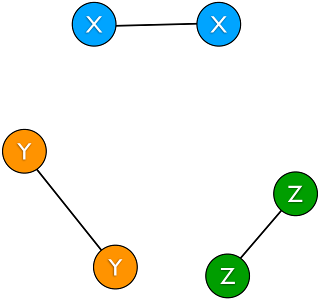
- In some cases, we want sparsity in groups of parameters:
  1. Multi-class variables [Lee et al., 2006].

Structure Learning with Group ℓ_1-Regularization



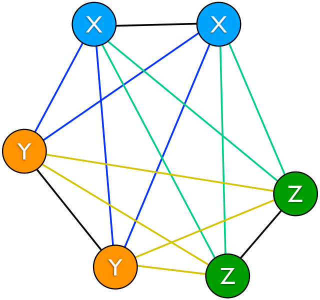
- In some cases, we want sparsity in groups of parameters:
  1. Multi-class variables [Lee et al., 2006].
  2. Blockwise-sparsity [Duchi et al., 2008].

Structure Learning with Group ℓ_1-Regularization



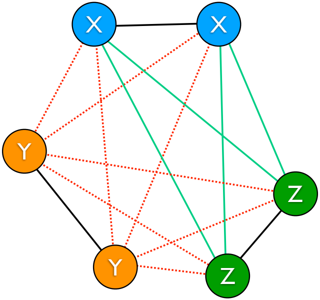
- In some cases, we want sparsity in groups of parameters:
  1. Multi-class variables [Lee et al., 2006].
  2. Blockwise-sparsity [Duchi et al., 2008].

Structure Learning with Group ℓ_1-Regularization



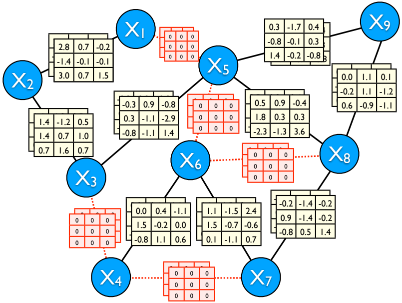
- In some cases, we want sparsity in groups of parameters:
  1. Multi-class variables [Lee et al., 2006].
  2. Blockwise-sparsity [Duchi et al., 2008].

Structure Learning with Group ℓ_1-Regularization



- In some cases, we want sparsity in groups of parameters:
  1. Multi-class variables [Lee et al., 2006].
  2. Blockwise-sparsity [Duchi et al., 2008].

Structure Learning with Group ℓ_1-Regularization



- In some cases, we want sparsity in groups of parameters:
  1. Multi-class variables [Lee et al., 2006].
  2. Blockwise-sparsity [Duchi et al., 2008].
  3. Conditional random fields [Schmidt et al., 2008]

Structure Learning with Group ℓ_1-Regularization

- Encourage group sparsity using group ℓ_1-regularization:

$$\min_x f(x) + \lambda \|x\|_{1,p},$$

where

$$\|x\|_{1,p} = \sum_g \|x_g\|_p.$$

Structure Learning with Group ℓ_1-Regularization

- Encourage group sparsity using group ℓ_1-regularization:

$$\min_x f(x) + \lambda \|x\|_{1,p},$$

where

$$\|x\|_{1,p} = \sum_g \|x_g\|_p.$$

- This is ℓ_1-regularization of group norms.
- Typically $p = 2$, but other norms give other properties.

Structure Learning with Group ℓ_1-Regularization

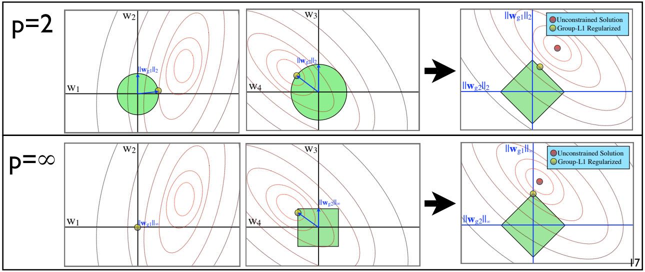
- Encourage group sparsity using group ℓ_1-regularization:

$$\min_x f(x) + \lambda \|x\|_{1,p},$$

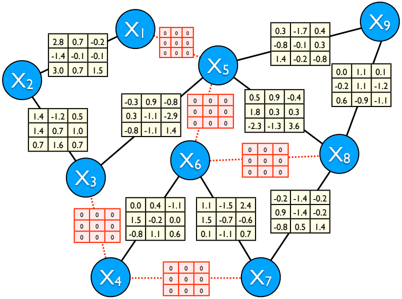
where

$$\|x\|_{1,p} = \sum_g \|x_g\|_p.$$

- This is ℓ_1-regularization of group norms.
- Typically $p = 2$, but other norms give other properties.

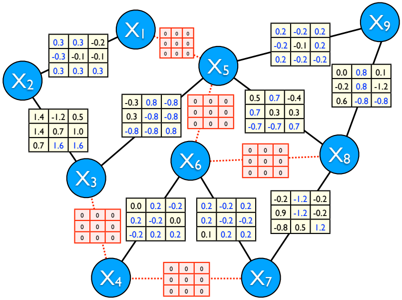


Effect of Different Group Norms



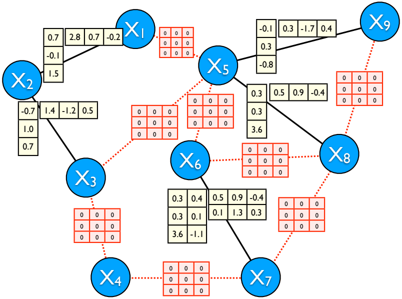
- Group ℓ_1-Regularization with the ℓ_2 group norm.
- Encourages group sparsity.

Effect of Different Group Norms



- Group ℓ_1-Regularization with the ℓ_∞ group norm.
- Encourages group sparsity and parameter tieing.

Effect of Different Group Norms



- Group ℓ_1-Regularization with the nuclear group norm.
- Encourages group sparsity and low-rank.

Optimization with Group ℓ_1-Regularization

- We'll focus on the group ℓ_1-regularized optimization:

$$\min_x f(x) + \lambda \|x\|_{1,2},$$

where f is the CRF (expensive) objective.

Optimization with Group ℓ_1-Regularization

- We'll focus on the group ℓ_1-regularized optimization:

$$\min_x f(x) + \lambda \|x\|_{1,2},$$

  where f is the CRF (expensive) objective.
- The regularizer is non-separable.

Optimization with Group ℓ_1-Regularization

- We'll focus on the group ℓ_1-regularized optimization:

$$\min_x f(x) + \lambda \|x\|_{1,2},$$

where f is the CRF (expensive) objective.

- The regularizer is non-separable.
- But the regularizer is simple.

Optimization with Group ℓ_1-Regularization

- We'll focus on the group ℓ_1-regularized optimization:

$$\min_x f(x) + \lambda \|x\|_{1,2},$$

  where f is the CRF (expensive) objective.

- The regularizer is non-separable.
- But the regularizer is simple.
- Can we extend quasi-Newton methods using this property?

Converting to a Constrained Problem

- We can re-write the non-smooth objective

$$\min_x f(x) + \lambda \sum_g \|x\|,$$

as a smooth objective with norm-cone constraints:

$$\min_{\|x_g\| \leq t_g} F(x) = f(x) + \lambda \sum_g t_g.$$

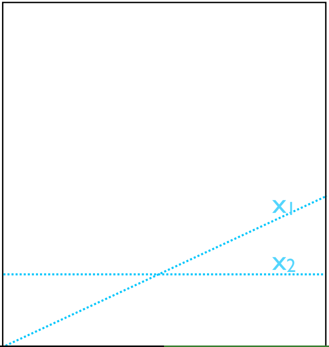
Converting to a Constrained Problem

- We can re-write the non-smooth objective

$$\min_x f(x) + \lambda \sum_g \|x\|,$$

as a smooth objective with norm-cone constraints:

$$\min_{\|x_g\| \leq t_g} F(x) = f(x) + \lambda \sum_g t_g.$$



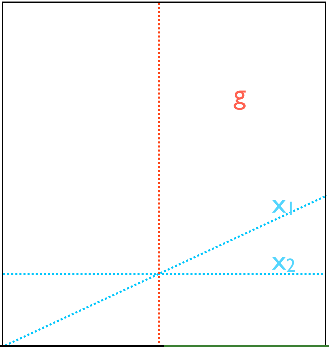
Converting to a Constrained Problem

- We can re-write the non-smooth objective

$$\min_x f(x) + \lambda \sum_g \|x\|,$$

as a smooth objective with norm-cone constraints:

$$\min_{\|x_g\| \leq t_g} F(x) = f(x) + \lambda \sum_g t_g.$$



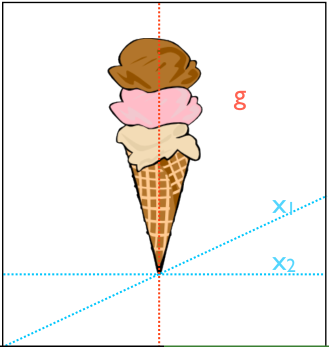
Converting to a Constrained Problem

- We can re-write the non-smooth objective

$$\min_x f(x) + \lambda \sum_g \|x\|,$$

as a smooth objective with norm-cone constraints:

$$\min_{\|x_g\| \leq t_g} F(x) = f(x) + \lambda \sum_g t_g.$$



Converting to a Constrained Problem

- We can re-write the non-smooth objective

$$\min_x f(x) + \lambda \sum_g \|x\|,$$

as a smooth objective with norm-cone constraints:

$$\min_{\|x\|_p \leq t_g} F(x) = f(x) + \lambda \sum_g t_g.$$

- Properties of this problem:
  1. the number of parameters is large.
  2. evaluating $F(x)$ is expensive.
  3. we have constraints.

Converting to a Constrained Problem

- We can re-write the non-smooth objective

$$\min_x f(x) + \lambda \sum_g \|x\|,$$

as a smooth objective with norm-cone constraints:

$$\min_{\|x\|_p \leq t_g} F(x) = f(x) + \lambda \sum_g t_g.$$

- Properties of this problem:
  1. the number of parameters is large.
  2. evaluating $F(x)$ is expensive.
  3. we have constraints.
- But the constraints are simple:
  - We can compute the projection in linear time.

Converting to a Constrained Problem

- We can re-write the non-smooth objective

$$\min_x f(x) + \lambda \sum_g \|x\|,$$

as a smooth objective with norm-cone constraints:

$$\min_{\|x\|_p \leq t_g} F(x) = f(x) + \lambda \sum_g t_g.$$

- Properties of this problem:
  1. the number of parameters is large.
  2. evaluating $F(x)$ is expensive.
  3. we have constraints.
- But the constraints are simple:
  - We can compute the projection in linear time.
- We want to optimize costly objectives with simple constraints.

Projected Gradient over General Convex Sets

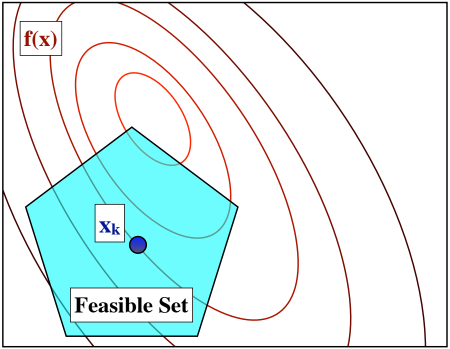
A general form of projected gradient:

$$x^{k+1} \leftarrow \operatorname*{arg\,min}_{x \in \mathcal{C}} ||x - (x^k - \alpha F'(x^k))||$$

Projected Gradient over General Convex Sets

A general form of projected gradient:

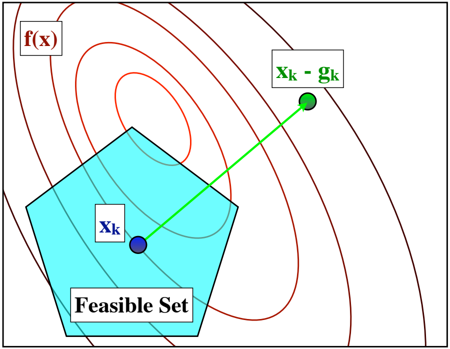
$$x^{k+1} \leftarrow \operatorname*{arg\,min}_{x \in \mathcal{C}} ||x - (x^k - \alpha F'(x^k))||$$



Projected Gradient over General Convex Sets

A general form of projected gradient:

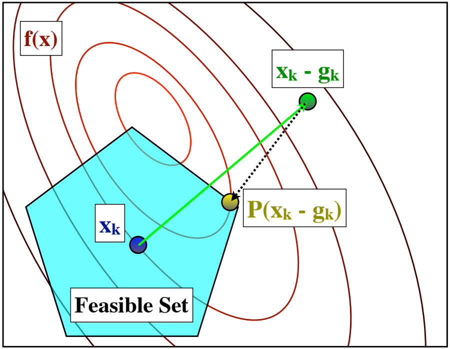
$$x^{k+1} \leftarrow \operatorname*{arg\,min}_{x \in \mathcal{C}} ||x - (x^k - \alpha F'(x^k))||$$



Projected Gradient over General Convex Sets

A general form of projected gradient:

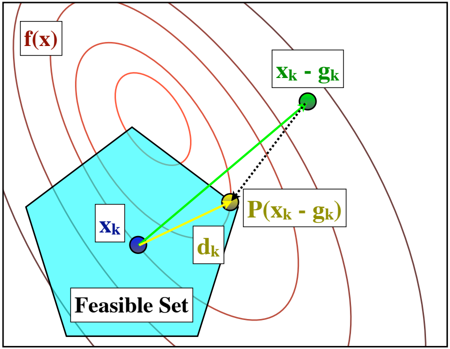
$$x^{k+1} \leftarrow \underset{x \in \mathcal{C}}{\arg\min} ||x - (x^k - \alpha F'(x^k))||$$



Projected Gradient over General Convex Sets

A general form of projected gradient:

$$x^{k+1} \leftarrow \operatorname*{arg\,min}_{x \in \mathcal{C}} ||x - (x^k - \alpha F'(x^k))||$$



Projected Newton

- We can consider a Newton-like step:

$$x^{k+1} \leftarrow \operatorname*{arg\,min}_{x \in \mathcal{C}} ||x - (x^k - \alpha [H_k]^{-1} F'(x^k))||,$$

but as we saw this doesn't work.

Projected Newton

- We can consider a Newton-like step:

$$x^{k+1} \leftarrow \operatorname*{arg\,min}_{x \in \mathcal{C}} ||x - (x^k - \alpha[H_k]^{-1} F'(x^k))||,$$

  but as we saw this doesn't work.

- Projected Newton methods project under the same norm:

$$x^{k+1} \leftarrow \operatorname*{arg\,min}_{x \in \mathcal{C}} ||x - (x^k - \alpha[H_k]^{-1} F'(x_k))||_{H^k},$$

  where $||x||_{H^k} = \sqrt{x^T H^k x}$.

  [Levitin & Polyak, 1966]

Projected Newton

- We can consider a Newton-like step:

$$x^{k+1} \leftarrow \operatorname*{arg\,min}_{x \in \mathcal{C}} ||x - (x^k - \alpha [H_k]^{-1} F'(x^k))||,$$

  but as we saw this doesn't work.

- Projected Newton methods project under the same norm:

$$x^{k+1} \leftarrow \operatorname*{arg\,min}_{x \in \mathcal{C}} ||x - (x^k - \alpha [H_k]^{-1} F'(x_k))||_{H^k},$$

  where $||x||_{H^k} = \sqrt{x^T H^k x}$.

  [Levitin & Polyak, 1966]

- Convergence properties similar to Newton's method.

Inexact Projected Newton

- Projected Newton methods equivalently minimize a constrained quadratic approximation:

$$x^{k+1} \leftarrow \underset{x \in \mathcal{C}}{\arg\min}\, F(x^k) + \langle F'(x^k), x - x^k \rangle + \frac{1}{2\alpha}\|x - x_k\|_{H_k}^2.$$

Inexact Projected Newton

- Projected Newton methods equivalently minimize a constrained quadratic approximation:

$$x^{k+1} \leftarrow \operatorname*{arg\,min}_{x \in \mathcal{C}} F(x^k) + \langle F'(x^k), x - x^k \rangle + \frac{1}{2\alpha}\|x - x_k\|^2_{H_k}.$$

- This is expensive even with simple constraints.

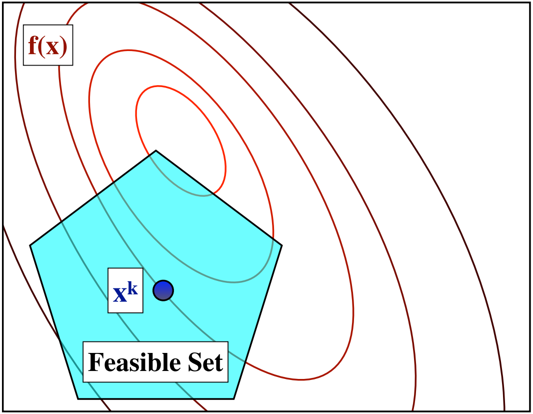
Inexact Projected Newton

- Projected Newton methods equivalently minimize a constrained quadratic approximation:

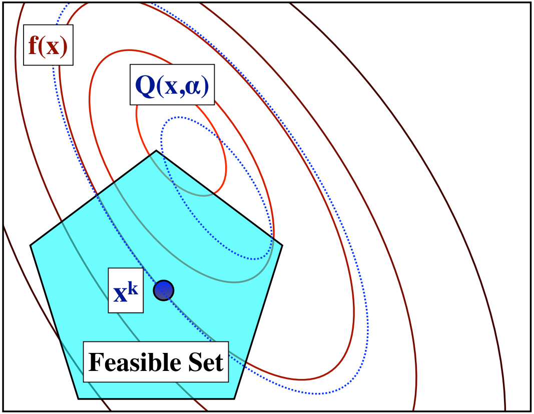
$$x^{k+1} \leftarrow \operatorname*{arg\,min}_{x \in \mathcal{C}} F(x^k) + \langle F'(x^k), x - x^k \rangle + \frac{1}{2\alpha}\|x - x_k\|_{H_k}^2.$$

- This is expensive even with simple constraints.
- Solution: use a cheap approximate solver.

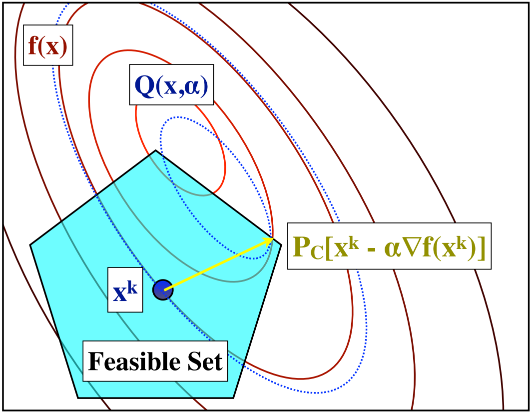
Inexact Projected Newton



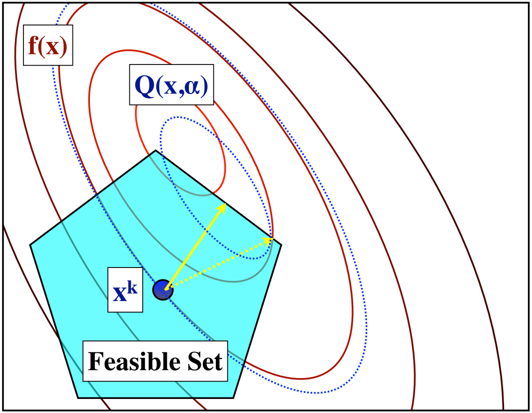
Inexact Projected Newton



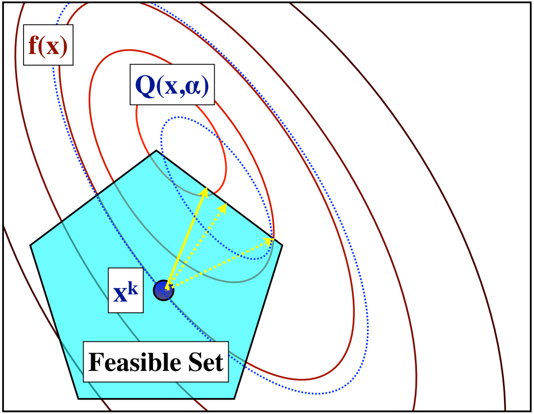
Inexact Projected Newton



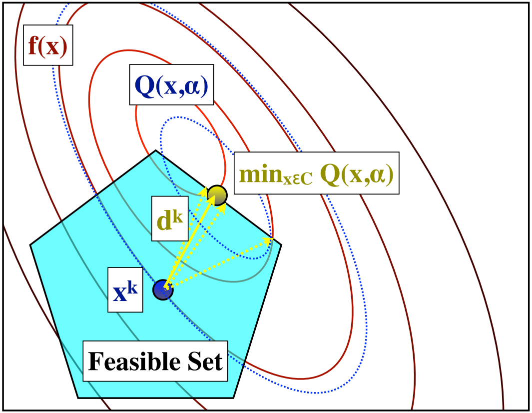
Inexact Projected Newton



Inexact Projected Newton



Inexact Projected Newton



Inexact Projected Newton

- Can we terminate this early?

Inexact Projected Newton

- Can we terminate this early?
  - For small enough α, we just need $Q(x, \alpha)$ less than $f(x^k)$.

Inexact Projected Newton

- Can we terminate this early?
  - For small enough α, we just need $Q(x, \alpha)$ less than $f(x^k)$.
- Can we efficiently get an approximate solution?

Inexact Projected Newton

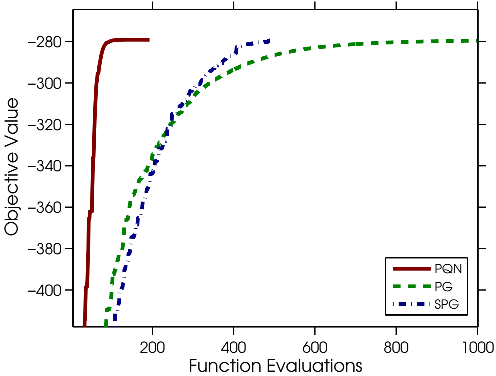
- Can we terminate this early?
  - For small enough α, we just need $Q(x, \alpha)$ less than $f(x^k)$.
- Can we efficiently get an approximate solution?
  - Schmidt et al. [2009]: use a quasi-Newton approximation of H_k and use (spectral) projected-gradient on $Q(x, \alpha)$:
    - Quasi-Newton approximation: linear time/space inner iterations.
    - Simple constraints: inner projection step takes linear time.
    - Efficient for optimizing costly functions with simple constraints.

Inexact Projected Newton

- Can we terminate this early?
  - For small enough α, we just need $Q(x, \alpha)$ less than $f(x^k)$.
- Can we efficiently get an approximate solution?
  - Schmidt et al. [2009]: use a quasi-Newton approximation of H_k and use (spectral) projected-gradient on $Q(x, \alpha)$:
    - Quasi-Newton approximation: linear time/space inner iterations.
    - Simple constraints: inner projection step takes linear time.
    - Efficient for optimizing costly functions with simple constraints.
- The projected quasi-Newton (PQN) approach:
  - Best paper prize at AI/Stats.
  - "The projected quasi-Newton (PQN) algorithm [19, 20] is perhaps the most elegant and logical extension of quasi-Newton methods, but it involves solving a sub-iteration." [Becker and Fadili, 2012].
  - "PQN is an implementation that uses a limited-memory quasi-Newton update and has both excellent empirical performance and theoretical properties." [Lee et al., 2012].
  - `http://www.di.ens.fr/~mschmidt/Software/PQN.html`

Graphical Model Structure Learning with Groups

Comparing PQN to first-order methods on a graphical model structure learning problem. [Gasch et al., 2000, Duchi et al., 2008].



Proximal Operators

- As before, we may not want to introduce constraints:
  - Increases number of variables.
  - Constrained problem may be harder.
- Can we use the same tricks without introducing constraints?

Proximal Operators

- As before, we may not want to introduce constraints:
  - Increases number of variables.
  - Constrained problem may be harder.
- Can we use the same tricks without introducing constraints?
- Yes, with proximal-gradient methods.

Overview of the Basic Gradient Method

- We want to solve a smooth optimization problem,

$$\min_x \; f(x).$$

Overview of the Basic Gradient Method

- We want to solve a smooth optimization problem,

$$\min_x \; f(x).$$

- At iteration x_k we use a quadratic upper bound on f,

$$x_{k+1} = \operatorname*{arg\,min}_x \; f(x_k) + \langle f'(x_k), x - x_k \rangle + \frac{1}{2\alpha}\|x - x_k\|^2.$$

Overview of the Basic Gradient Method

- We want to solve a smooth optimization problem,

$$\min_x \; f(x).$$

- At iteration x_k we use a quadratic upper bound on f,

$$x_{k+1} = \mathop{\arg\min}_x \; f(x_k) + \langle f'(x_k), x - x_k \rangle + \frac{1}{2\alpha}\|x - x_k\|^2.$$

- We can equivalently write this as the quadratic optimization

$$x_{k+1} = \mathop{\arg\min}_x \; \frac{1}{2}\|x - (x_k - \alpha f'(x_k))\|^2.$$

Overview of the Basic Gradient Method

- We want to solve a smooth optimization problem,

$$\min_x \; f(x).$$

- At iteration x_k we use a quadratic upper bound on f,

$$x_{k+1} = \operatorname*{arg\,min}_x \; f(x_k) + \langle f'(x_k), x - x_k \rangle + \frac{1}{2\alpha}\|x - x_k\|^2.$$

- We can equivalently write this as the quadratic optimization

$$x_{k+1} = \operatorname*{arg\,min}_x \; \frac{1}{2}\|x - (x_k - \alpha f'(x_k))\|^2.$$

- The solution is the gradient algorithm:

$$x_{k+1} = x_k - \alpha f'(x_k).$$

Overview of the Basic Proximal-Gradient Method

- We want to solve a smooth optimization problem,

$$\min_x \; f(x).$$

- At iteration x_k we use a quadratic upper bound on f,

$$x_{k+1} = \arg\min_x \; f(x_k) + \langle f'(x_k), x - x_k \rangle + \frac{1}{2\alpha}\|x - x_k\|^2.$$

- We can equivalently write this as the quadratic optimization

$$x_{k+1} = \arg\min_x \; \frac{1}{2}\|x - (x_k - \alpha f'(x_k))\|^2.$$

- The solution is the gradient algorithm:

$$x_{k+1} = x_k - \alpha f'(x_k).$$

Overview of the Basic Proximal-Gradient Method

- We want to solve a composite optimization problem,

$$\min_x \; f(x) + g(x).$$

- At iteration x_k we use a quadratic upper bound on f,

$$x_{k+1} = \operatorname*{arg\,min}_x \; f(x_k) + \langle f'(x_k), x - x_k \rangle + \frac{1}{2\alpha}\|x - x_k\|^2.$$

- We can equivalently write this as the quadratic optimization

$$x_{k+1} = \operatorname*{arg\,min}_x \; \frac{1}{2}\|x - (x_k - \alpha f'(x_k))\|^2.$$

- The solution is the gradient algorithm:

$$x_{k+1} = x_k - \alpha f'(x_k).$$

Overview of the Basic Proximal-Gradient Method

- We want to solve a composite optimization problem,

$$\min_x \; f(x) + g(x).$$

- At iteration x_k we use a quadratic upper bound on f,

$$x_{k+1} = \arg\min_x \; f(x_k) + \langle f'(x_k), x - x_k \rangle + \frac{1}{2\alpha}\|x - x_k\|^2 + g(x).$$

- We can equivalently write this as the quadratic optimization

$$x_{k+1} = \arg\min_x \; \frac{1}{2}\|x - (x_k - \alpha f'(x_k))\|^2.$$

- The solution is the gradient algorithm:

$$x_{k+1} = x_k - \alpha f'(x_k).$$

Overview of the Basic Proximal-Gradient Method

- We want to solve a composite optimization problem,

$$\min_x \; f(x) + g(x).$$

- At iteration x_k we use a quadratic upper bound on f,

$$x_{k+1} = \arg\min_x \; f(x_k) + \langle f'(x_k), x - x_k \rangle + \frac{1}{2\alpha}\|x - x_k\|^2 + g(x).$$

- We can equivalently write this as the proximal optimization

$$x_{k+1} = \arg\min_x \; \frac{1}{2}\|x - (x_k - \alpha f'(x_k))\|^2 + \alpha g(x).$$

- The solution is the gradient algorithm:

$$x_{k+1} = x_k - \alpha f'(x_k).$$

Overview of the Basic Proximal-Gradient Method

- We want to solve a composite optimization problem,

$$\min_x \; f(x) + g(x).$$

- At iteration x_k we use a quadratic upper bound on f,

$$x_{k+1} = \operatorname*{arg\,min}_x \; f(x_k) + \langle f'(x_k), x - x_k \rangle + \frac{1}{2\alpha}\|x - x_k\|^2 + g(x).$$

- We can equivalently write this as the proximal optimization

$$x_{k+1} = \operatorname*{arg\,min}_x \; \frac{1}{2}\|x - (x_k - \alpha f'(x_k))\|^2 + \alpha g(x).$$

- The solution is the proximal-gradient algorithm:

$$x_{k+1} = \mathrm{prox}_{\alpha g}[x_k - \alpha f'(x_k)].$$

Special case of Projected-Gradient Methods

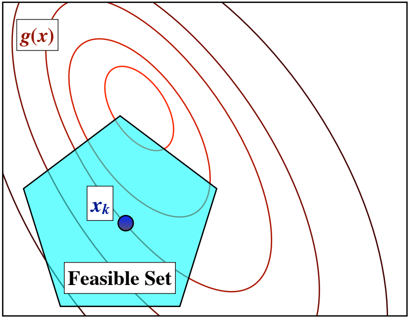
- Projected-gradient methods are a special case:

$$g(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ \infty & \text{if } x \notin \mathcal{C}. \end{cases}$$

Special case of Projected-Gradient Methods

- Projected-gradient methods are a special case:

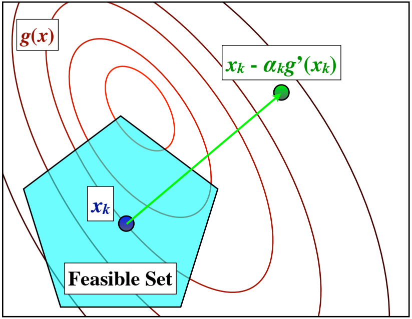
$$g(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ \infty & \text{if } x \notin \mathcal{C}. \end{cases}$$



Special case of Projected-Gradient Methods

- Projected-gradient methods are a special case:

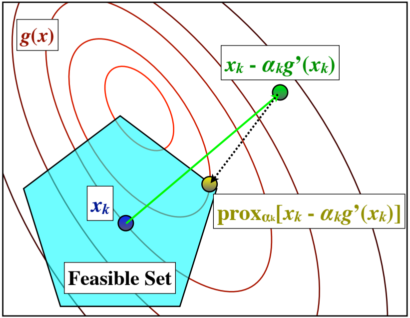
$$g(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ \infty & \text{if } x \notin \mathcal{C}. \end{cases}$$



Special case of Projected-Gradient Methods

- Projected-gradient methods are a special case:

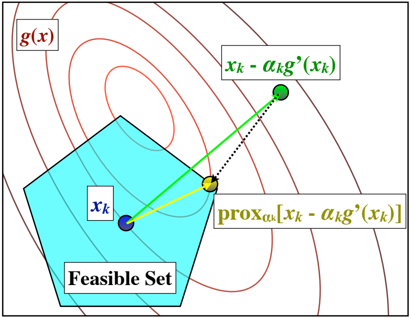
$$g(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ \infty & \text{if } x \notin \mathcal{C}. \end{cases}$$



Special case of Projected-Gradient Methods

- Projected-gradient methods are a special case:

$$g(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ \infty & \text{if } x \notin \mathcal{C}. \end{cases}$$



file:///Users/Mark/Pictures/2011/11Paris/MVI_0605.MOV

Special case of Iterative Soft-Thresholding Methods

- Iterative Soft-Thresholding methods are a special case:

$$g(x) = \lambda \|x\|_1.$$

Special case of Iterative Soft-Thresholding Methods

- Iterative Soft-Thresholding methods are a special case:

$$g(x) = \lambda \|x\|_1.$$

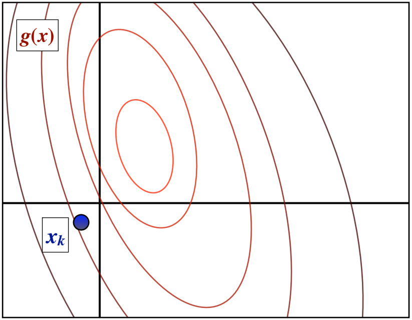
- In this case, proximal operator shrinks $|x_i|$ by up to $\lambda\alpha$.

Special case of Iterative Soft-Thresholding Methods

- Iterative Soft-Thresholding methods are a special case:

$$g(x) = \lambda \|x\|_1.$$

- In this case, proximal operator shrinks $|x_i|$ by up to $\lambda\alpha$.

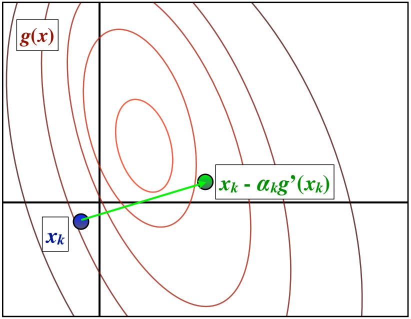


Special case of Iterative Soft-Thresholding Methods

- Iterative Soft-Thresholding methods are a special case:

$$g(x) = \lambda \|x\|_1.$$

- In this case, proximal operator shrinks $|x_i|$ by up to $\lambda\alpha$.

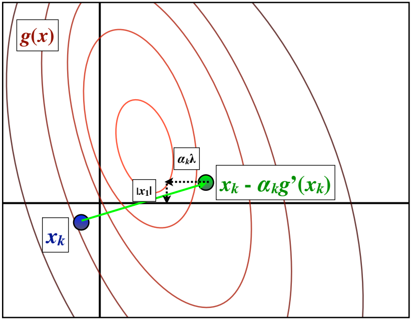


Special case of Iterative Soft-Thresholding Methods

- Iterative Soft-Thresholding methods are a special case:

$$g(x) = \lambda \|x\|_1.$$

- In this case, proximal operator shrinks $|x_i|$ by up to $\lambda\alpha$.

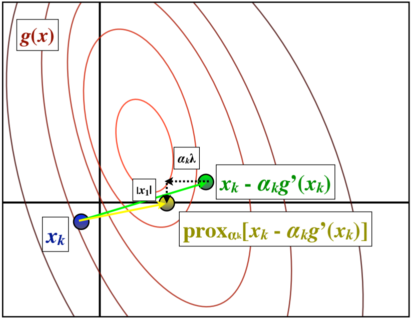


Special case of Iterative Soft-Thresholding Methods

- Iterative Soft-Thresholding methods are a special case:

$$g(x) = \lambda\|x\|_1.$$

- In this case, proximal operator shrinks $|x_i|$ by up to $\lambda\alpha$.



file:///Users/Mark/Pictures/2011/12Paris/MVI_0643.MOV

Proximal Gradient for Group ℓ_1-Regularization

- The group ℓ_1-regularizer is simple; we can compute the proximal operator in linear time. [Wright et al., 2009]

$$\mathrm{prox}_{\alpha||x_g||}[x_g] = \arg\min_x \frac{1}{2}||x - x_g||^2 + \alpha||x||$$
$$= \frac{x_g}{||x_g||}\max\{0, ||x_g|| - \alpha\}$$

Proximal Gradient and Proximal Newton

- The basic proximal-gradient step:

$$x^{k+1} \leftarrow \arg\min_x \frac{1}{2}||x - (x^k - \alpha f'(x^k))||^2 + \alpha g(x)$$

Proximal Gradient and Proximal Newton

- The basic proximal-gradient step:

$$x^{k+1} \leftarrow \arg\min_x \frac{1}{2}||x - (x^k - \alpha f'(x^k))||^2 + \alpha g(x)$$

- Same convergence rate as gradient method.

Proximal Gradient and Proximal Newton

- The basic proximal-gradient step:

$$x^{k+1} \leftarrow \arg\min_x \frac{1}{2}||x - (x^k - \alpha f'(x^k))||^2 + \alpha g(x)$$

- Same convergence rate as gradient method.
- To speed the convergence, we might consider Newton-like step:

$$x^{k+1} \leftarrow \arg\min_x \frac{1}{2}||x - (x^k - \alpha[H_k]^{-1} f'(x^k))||^2 + \alpha g(x).$$

Proximal Gradient and Proximal Newton

- The basic proximal-gradient step:

$$x^{k+1} \leftarrow \operatorname*{arg\,min}_x \frac{1}{2}||x - (x^k - \alpha f'(x^k))||^2 + \alpha g(x)$$

- Same convergence rate as gradient method.
- To speed the convergence, we might consider Newton-like step:

$$x^{k+1} \leftarrow \operatorname*{arg\,min}_x \frac{1}{2}||x - (x^k - \alpha[H_k]^{-1} f'(x^k))||^2 + \alpha g(x).$$

- But to ensure descent, we need to match the norms:

$$x^{k+1} \leftarrow \operatorname*{arg\,min}_x \frac{1}{2}||x - (x^k - \alpha[H_k]^{-1} f'(x^k))||^2_{H^k} + \alpha g(x)$$

Proximal Gradient and Proximal Newton

- The basic proximal-gradient step:

$$x^{k+1} \leftarrow \arg\min_x \frac{1}{2}||x - (x^k - \alpha f'(x^k))||^2 + \alpha g(x)$$

- Same convergence rate as gradient method.
- To speed the convergence, we might consider Newton-like step:

$$x^{k+1} \leftarrow \arg\min_x \frac{1}{2}||x - (x^k - \alpha [H_k]^{-1} f'(x^k))||^2 + \alpha g(x).$$

- But to ensure descent, we need to match the norms:

$$x^{k+1} \leftarrow \arg\min_x \frac{1}{2}||x - (x^k - \alpha [H_k]^{-1} f'(x^k))||^2_{H^k} + \alpha g(x)$$

- As before, this will expensive even when g is simple.

Inexact Proximal Newton

- Inexact proximal-Newton method:
  - Use a cheap inner solver to approximate the step.

Inexact Proximal Newton

- Inexact proximal-Newton method:
  - Use a cheap inner solver to approximate the step.
- Method analogous to PQN:
  - L-BFGS quasi-Newton Hessian approximation.
  - Proximal-gradient method as inner solver.

    [Beck & Teboulle, 2008, Hofling & Tibshirani, 2009, Wright et al., 2009]
  - Suitable for optimizing costly objectives with simple regularizers.

Inexact Proximal Newton

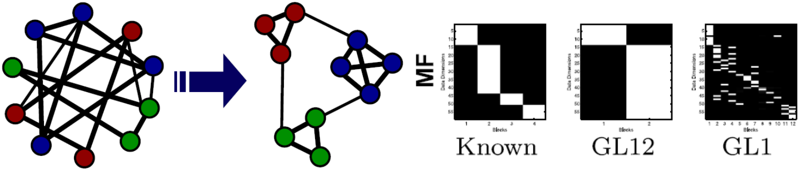
- Inexact proximal-Newton method:
  - Use a cheap inner solver to approximate the step.
- Method analogous to PQN:
  - L-BFGS quasi-Newton Hessian approximation.
  - Proximal-gradient method as inner solver.

    [Beck & Teboulle, 2008, Hofling & Tibshirani, 2009, Wright et al., 2009]
  - Suitable for optimizing costly objectives with simple regularizers.
- Proximal-Newton is increasing in popularity, e.g. NIPS 2012:
  - Becker & Fadili, Hsieh et al., Lee et al., Olsen et al., Pacheco & Sudderth.

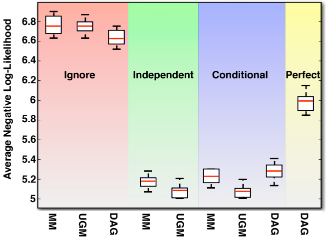
Motivation: Structure Learning in Graphical Models

PQN has been used in other structure learning applications:

- Learning variable groups [Marlin et al., 2009].



- Non-DAG approaches to causality [Duvenaud et al., 2010].



Outline

1. Sparsity
2. Group Sparsity
3. Structured Sparsity
4. Big-N Problems

Structure Learning with ℓ_1-Regularization

A list of papers on this topic (incomplete):

[Li & Yang, 2004], [Li & Yang, 2005], [Banerjee et al., 2006], [Huang et al., 2006], [Lee et al., 2006], [Meinshausen & Bühlmann, 2006], [Wainwright et al., 2006], [Dahinden et al., 2007], [Schmidt et al., 2007], [Shimamura et al., 2007], [Yuan & Lin, 2007], [d' Aspremont et al., 2008], [Banerjee et al., 2008], [Dahl et al., 2008], [Duchi et al., 2008], [Friedman et al., 2008], [Kolar & Xing, 2008], [Levina et al., 2008], [Schmidt et al., 2008], [Fan & Feng, 2009], [Höling & Tibshirani, 2009], [Krishnamurphy & d'Aspremont, 2009], [Lu, 2009a], [Lu, 2009b], [Marlin et al., 2009a], [Marlin et al., 2009b], [Schmidt et al., 2009], [Schmidt & Murphy, 2009], [Schnitzspan et al., 2009], [Yuan, 2009]. Many more since 2009...

Structure Learning with ℓ_1-Regularization

Many of these papers have made the pairwise assumption:

[Li & Yang, 2004], [Li & Yang, 2005], [Banerjee et al., 2006], [Huang et al., 2006], [Lee et al., 2006], [Meinshausen & Bühlmann, 2006], [Wainwright et al., 2006], [Dahinden et al., 2007], [Schmidt et al., 2007], [Shimamura et al., 2007], [Yuan & Lin, 2007], [d' Aspremont et al., 2008], [Banerjee et al., 2008], [Dahl et al., 2008], [Duchi et al., 2008], [Friedman et al., 2008], [Kolar & Xing, 2008], [Levina et al., 2008], [Schmidt et al., 2008], [Fan & Feng, 2009], [Höling & Tibshirani, 2009], [Krishnamurphy & d'Aspremont, 2009], [Lu, 2009a], [Lu, 2009b], [Marlin et al., 2009a], [Marlin et al., 2009b], [Schmidt et al., 2009], [Schmidt & Murphy, 2009], [Schnitzspan et al., 2009], [Yuan, 2009]. Many more since 2009...

Structure Learning with ℓ_1-Regularization

Many of these papers have made the pairwise assumption:

[Li & Yang, 2004], [Li & Yang, 2005], [Banerjee et al., 2006], [Huang et al., 2006], [Lee et al., 2006], [Meinshausen & Bühlmann, 2006], [Wainwright et al., 2006], [Dahinden et al., 2007], [Schmidt et al., 2007], [Shimamura et al., 2007], [Yuan & Lin, 2007], [d' Aspremont et al., 2008], [Banerjee et al., 2008], [Dahl et al., 2008], [Duchi et al., 2008], [Friedman et al., 2008], [Kolar & Xing, 2008], [Levina et al., 2008], [Schmidt et al., 2008], [Fan & Feng, 2009], [Höling & Tibshirani, 2009], [Krishnamurphy & d'Aspremont, 2009], [Lu, 2009a], [Lu, 2009b], [Marlin et al., 2009a], [Marlin et al., 2009b], [Schmidt et al., 2009], [Schmidt & Murphy, 2009], [Schnitzspan et al., 2009], [Yuan, 2009]. Many more since 2009...

Structure Learning with ℓ_1-Regularization

Many of these papers have made the pairwise assumption:

[Li & Yang, 2004], [Li & Yang, 2005], [Banerjee et al., 2006], [Huang et al., 2006], [Lee et al., 2006], [Meinshausen & Bühlmann, 2006], [Wainwright et al., 2006], [Dahinden et al., 2007], [Schmidt et al., 2007], [Shimamura et al., 2007], [Yuan & Lin, 2007], [d' Aspremont et al., 2008], [Banerjee et al., 2008], [Dahl et al., 2008], [Duchi et al., 2008], [Friedman et al., 2008], [Kolar & Xing, 2008], [Levina et al., 2008], [Schmidt et al., 2008], [Fan & Feng, 2009], [Höling & Tibshirani, 2009], [Krishnamurphy & d'Aspremont, 2009], [Lu, 2009a], [Lu, 2009b], [Marlin et al., 2009a], [Marlin et al., 2009b], [Schmidt et al., 2009], [Schmidt & Murphy, 2009], [Schnitzspan et al., 2009], [Yuan, 2009]. Many more since 2009...

Structure Learning with ℓ_1-Regularization

Many of these papers have made the pairwise assumption:

[Li & Yang, 2004], [Li & Yang, 2005], [Banerjee et al., 2006], [Huang et al., 2006], [Lee et al., 2006], [Meinshausen & Bühlmann, 2006], [Wainwright et al., 2006], [Dahinden et al., 2007], [Schmidt et al., 2007], [Shimamura et al., 2007], [Yuan & Lin, 2007], [d' Aspremont et al., 2008], [Banerjee et al., 2008], [Dahl et al., 2008], [Duchi et al., 2008], [Friedman et al., 2008], [Kolar & Xing, 2008], [Levina et al., 2008], [Schmidt et al., 2008], [Fan & Feng, 2009], [Höling & Tibshirani, 2009], [Krishnamurphy & d'Aspremont, 2009], [Lu, 2009a], [Lu, 2009b], [Marlin et al., 2009a], [Marlin et al., 2009b], [Schmidt et al., 2009], [Schmidt & Murphy, 2009], [Schnitzspan et al., 2009], [Yuan, 2009]. Many more since 2009...

Beyond Pairwise Potentials

- The pairwise assumption is inherent to Gaussian models.

Beyond Pairwise Potentials

- The pairwise assumption is inherent to Gaussian models.
- It has not traditionally been used in log-linear models.

  [Goodman, 1971, Bishop et al., 1975]

Beyond Pairwise Potentials

- The pairwise assumption is inherent to Gaussian models.
- It has not traditionally been used in log-linear models.

  [Goodman, 1971, Bishop et al., 1975]
- The assumption is restrictive if higher-order statistics matter.
- Eg. Mutations in both gene *A* and gene *B* lead to cancer.

Beyond Pairwise Potentials

- The pairwise assumption is inherent to Gaussian models.
- It has not traditionally been used in log-linear models.

  [Goodman, 1971, Bishop et al., 1975]
- The assumption is restrictive if higher-order statistics matter.
- Eg. Mutations in both gene A and gene B lead to cancer.
- We want to go beyond pairwise potentials.

General Log-Linear Models

- Log-linear models write the probability of a vector x as

$$\log p(x) = \sum_{A \subseteq S} w_A^T \phi_A(x_A) - \log Z$$

General Log-Linear Models

- Log-linear models write the probability of a vector x as

$$\log p(x) = \sum_{A \subseteq S} w_A^T \phi_A(x_A) - \log Z$$

- Setting $w_A = 0$ is equivalent to removing the potential.
- In pairwise models we assume $w_A = 0$ if $|A| > 2$.

Group ℓ_1-Regularization for Log-Linear Models

- We can extend group ℓ_1-regularization to the general case:

$$\min_w f(w) + \sum_{A \subseteq S} \lambda_A ||w_A||.$$

Group ℓ_1-Regularization for Log-Linear Models

- We can extend group ℓ_1-regularization to the general case:

$$\min_w f(w) + \sum_{A \subseteq S} \lambda_A ||w_A||.$$

- However,
  - We have an exponential number of variables.
  - Setting $w_A = 0$ does not give conditional independence.

Group ℓ_1-Regularization for Log-Linear Models

- We can extend group ℓ_1-regularization to the general case:

$$\min_{w} f(w) + \sum_{A \subseteq S} \lambda_A ||w_A||.$$

- However,
  - We have an exponential number of variables.
  - Setting $w_A = 0$ does not give conditional independence.
- Prior work restricted the cardinality (e.g., threeway models).

[Dahinden et al., 2007]

Hierarchical Log-Linear Models

- Instead of restricting cardinality, we use hierarchical inclusion:

Hierarchical Log-Linear Models

- Instead of restricting cardinality, we use hierarchical inclusion:
  - We can only have $(1, 2, 3)$ if we also have $(1, 2)$, $(1, 3)$, and $(2, 3)$.

Hierarchical Log-Linear Models

- Instead of restricting cardinality, we use hierarchical inclusion:
  - We can only have $(1, 2, 3)$ if we also have $(1, 2)$, $(1, 3)$, and $(2, 3)$.
- In general: If $w_A = 0$ then supersets B of A must have $w_B = 0$.

Hierarchical Log-Linear Models

- Instead of restricting cardinality, we use hierarchical inclusion:
  - We can only have $(1, 2, 3)$ if we also have $(1, 2)$, $(1, 3)$, and $(2, 3)$.
- In general: If $w_A = 0$ then supersets B of A must have $w_B = 0$.
- The class of hierarchical log-linear models:

  [Bishop et al., 1975]
  - Much larger than the set of pairwise models.
  - Can represent any positive distribution.
  - Group-sparsity corresponds to conditional independence.

Hierarchical Log-Linear Models

- Instead of restricting cardinality, we use hierarchical inclusion:
  - We can only have $(1, 2, 3)$ if we also have $(1, 2)$, $(1, 3)$, and $(2, 3)$.
- In general: If $w_A = 0$ then supersets B of A must have $w_B = 0$.
- The class of hierarchical log-linear models:

  [Bishop et al., 1975]
  - Much larger than the set of pairwise models.
  - Can represent any positive distribution.
  - Group-sparsity corresponds to conditional independence.
- But, how can we encourage this structured sparsity?

Structured Sparsity for Hierarchical Constraints

- Can enforce a hierarchy with overlapping group ℓ_1-regularization.

  [Bach, 2008, Zhao et al., 2009]

Structured Sparsity for Hierarchical Constraints

- Can enforce a hierarchy with overlapping group ℓ_1-regularization.

  [Bach, 2008, Zhao et al., 2009]

- Example:
  - If we want $A = 0$ to mean $B = 0$, use two groups $\{B\}$ and $\{A, B\}$,

$$\lambda_{\{B\}}||w_B||_2 + \lambda_{\{A,B\}}||w_{A,B}||_2.$$

  - To make w_A non-zero, pay $\lambda_{\{A,B\}}$.
  - To make w_B non-zero, pay λ_B (but also $\lambda_{\{A,B\}}$ if $w_A = 0$).
  - If $w_B \neq 0$, no penalty for making w_A non-zero.

Structured Sparsity for Hierarchical Constraints

- Can enforce a hierarchy with overlapping group ℓ_1-regularization.

  [Bach, 2008, Zhao et al., 2009]

- Example:
  - If we want $A = 0$ to mean $B = 0$, use two groups $\{B\}$ and $\{A, B\}$,

$$\lambda_{\{B\}}||w_B||_2 + \lambda_{\{A,B\}}||w_{A,B}||_2.$$

  - To make w_A non-zero, pay $\lambda_{\{A,B\}}$.
  - To make w_B non-zero, pay λ_B (but also $\lambda_{\{A,B\}}$ if $w_A = 0$).
  - If $w_B \neq 0$, no penalty for making w_A non-zero.

- We can learn hierarchical models by solving

$$\min_w f(w) + \sum_{A \subseteq S} \lambda_A \|w_{A^*}\|,$$

where $A^* = \{B | A \subseteq B\}$. [Schmidt & Murphy, 2010]

Active Set Method

- But can we avoid looking at all higher-order potentials?

Active Set Method

- But can we avoid looking at all higher-order potentials?
- Heuristic: only consider adding groups that satisfy hierarchichy.
  (And that are sub-optimal. E.g., poorly estimated by the model.)

Active Set Method

- But can we avoid looking at all higher-order potentials?
- Heuristic: only consider adding groups that satisfy hierarchichy.
  (And that are sub-optimal. E.g., poorly estimated by the model.)
- Convex analogue of [Cheeseman, 1983, Gevarter, 1987].
- Guarantees weak form of global optimality.

Example of Active Set Method

Initial boundary groups.

1 2 3 4 5

1,2 1,3 1,4 1,5 2,3 2,4 2,5 3,4 3,5 4,5

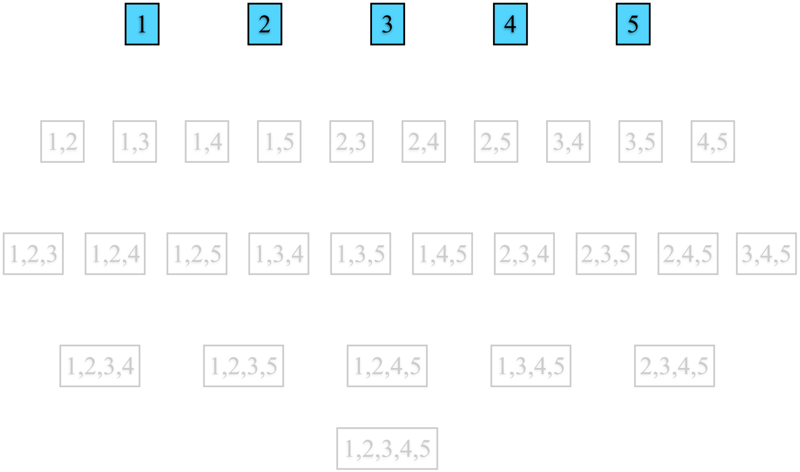
1,2,3 1,2,4 1,2,5 1,3,4 1,3,5 1,4,5 2,3,4 2,3,5 2,4,5 3,4,5

1,2,3,4 1,2,3,5 1,2,4,5 1,3,4,5 2,3,4,5

1,2,3,4,5

Example of Active Set Method

Optimize initial boundary groups.



Example of Active Set Method

Find new **active groups**.

1 2 3 4 5

1,2 1,3 1,4 1,5 2,3 2,4 2,5 3,4 3,5 4,5

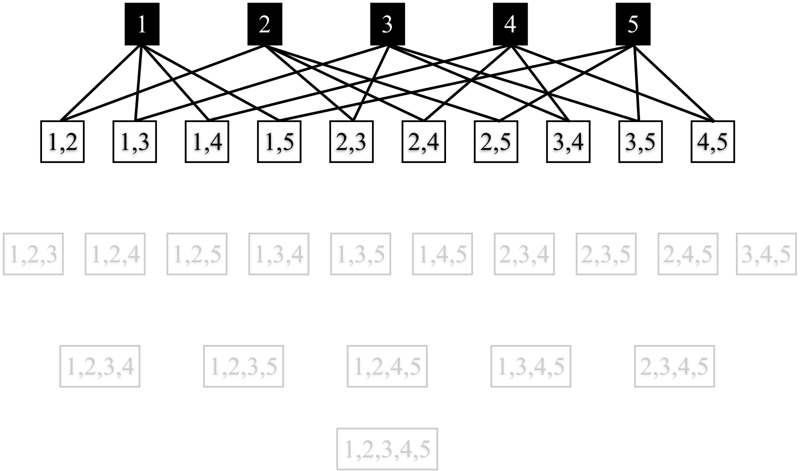
1,2,3 1,2,4 1,2,5 1,3,4 1,3,5 1,4,5 2,3,4 2,3,5 2,4,5 3,4,5

1,2,3,4 1,2,3,5 1,2,4,5 1,3,4,5 2,3,4,5

1,2,3,4,5

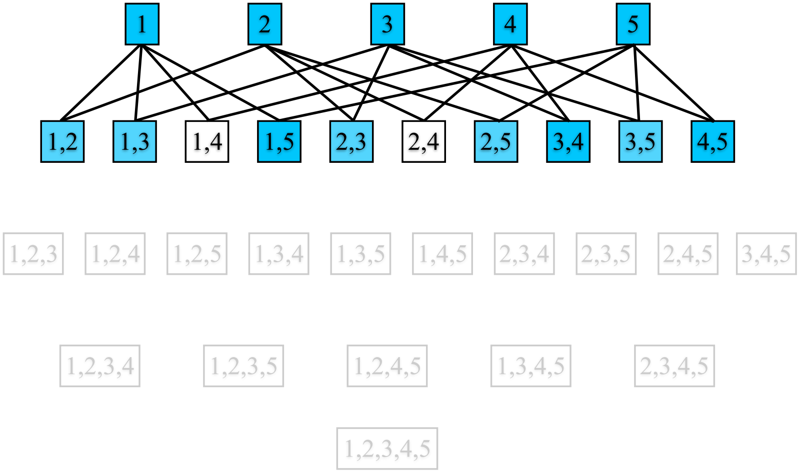
Example of Active Set Method

Find new boundary groups.



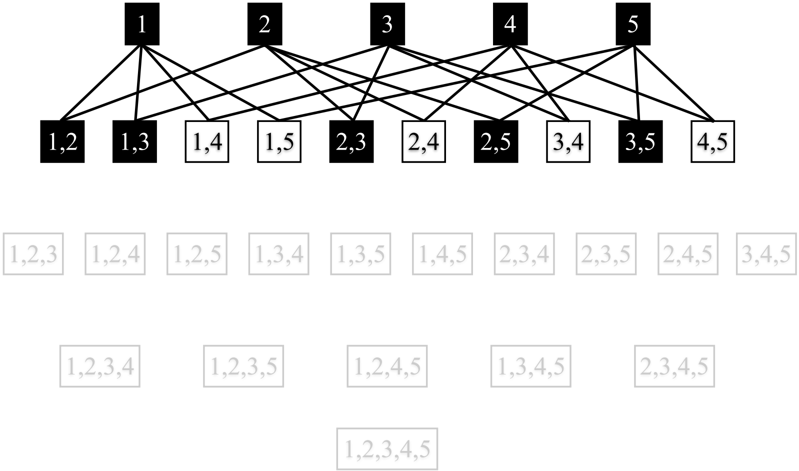
Example of Active Set Method

Optimize active groups and sub-optimal boundary groups.



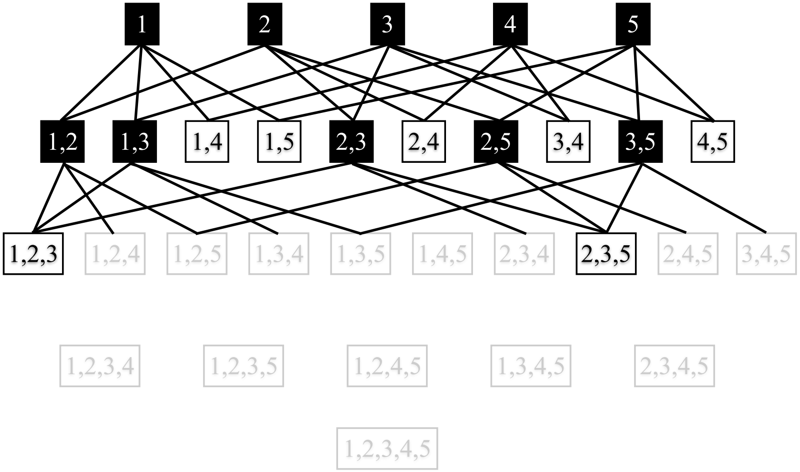
Example of Active Set Method

Find new **active groups**.



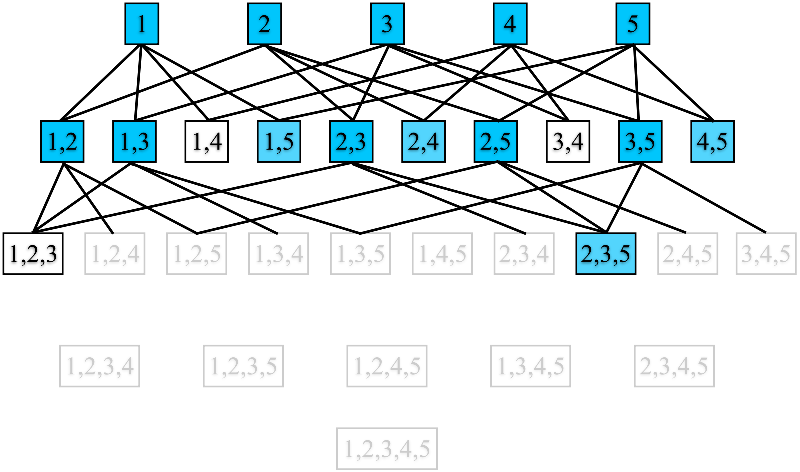
Example of Active Set Method

Find new boundary groups.



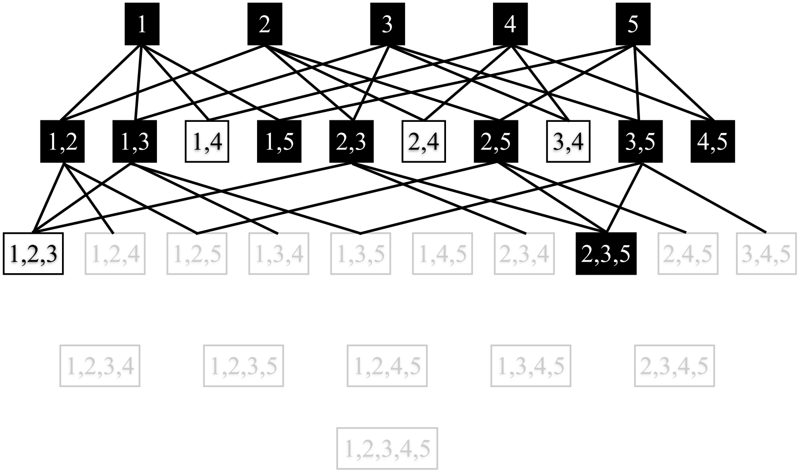
Example of Active Set Method

Optimize active groups and sub-optimal boundary groups.



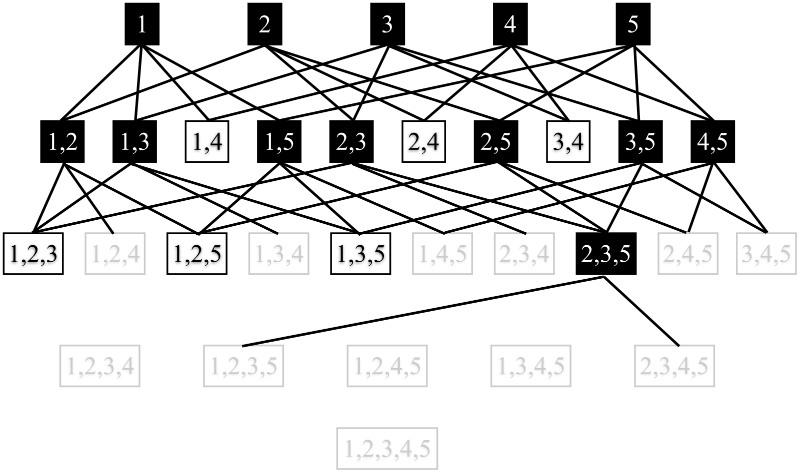
Example of Active Set Method

Find new **active groups**.



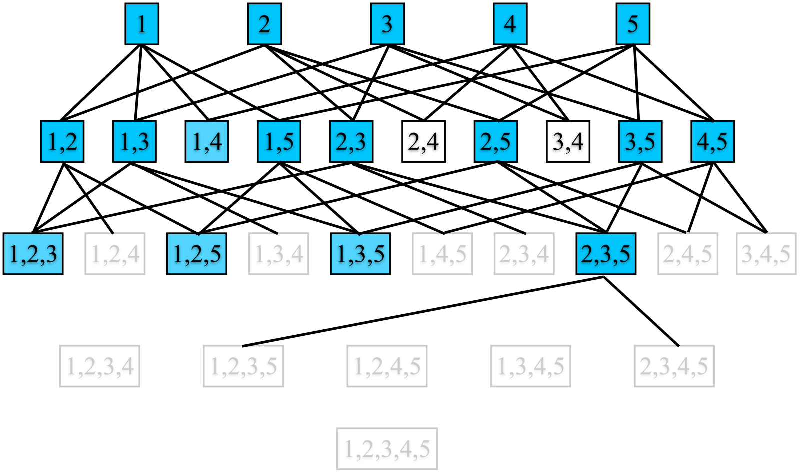
Example of Active Set Method

Find new boundary groups.



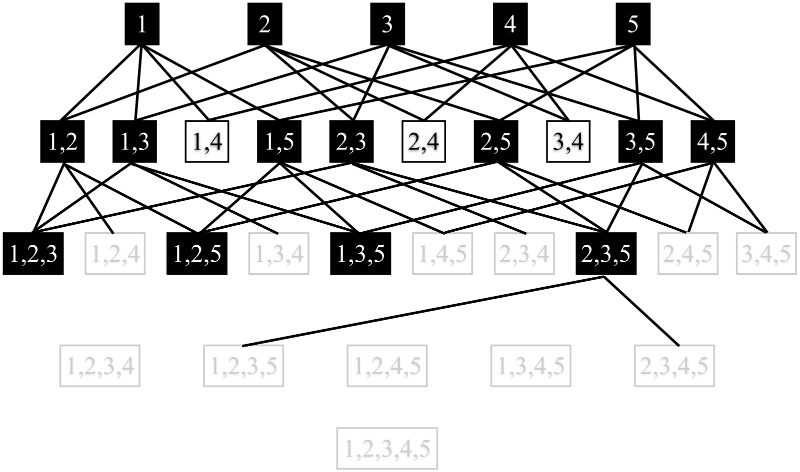
Example of Active Set Method

Optimize active groups and sub-optimal boundary groups.



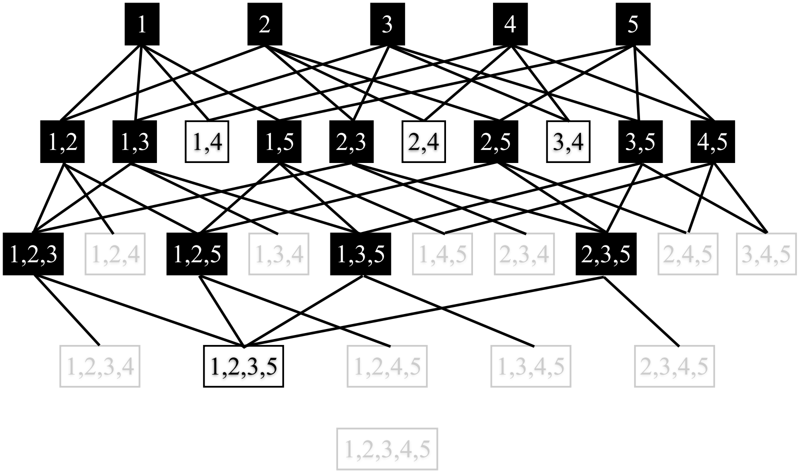
Example of Active Set Method

Find new **active groups**.



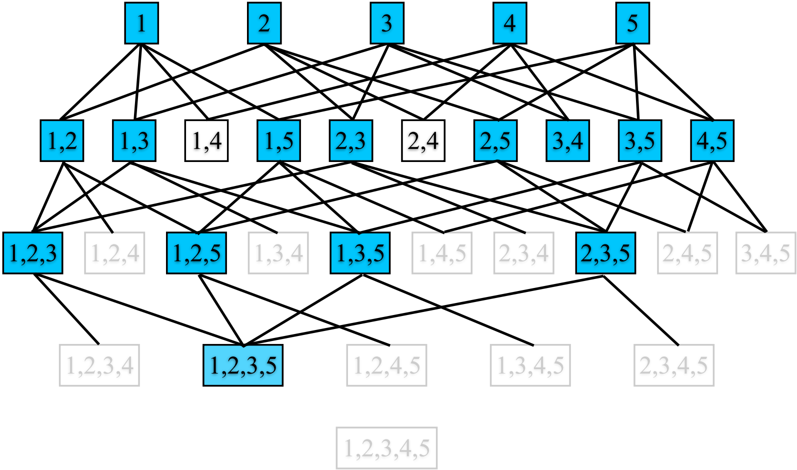
Example of Active Set Method

Find new boundary groups.



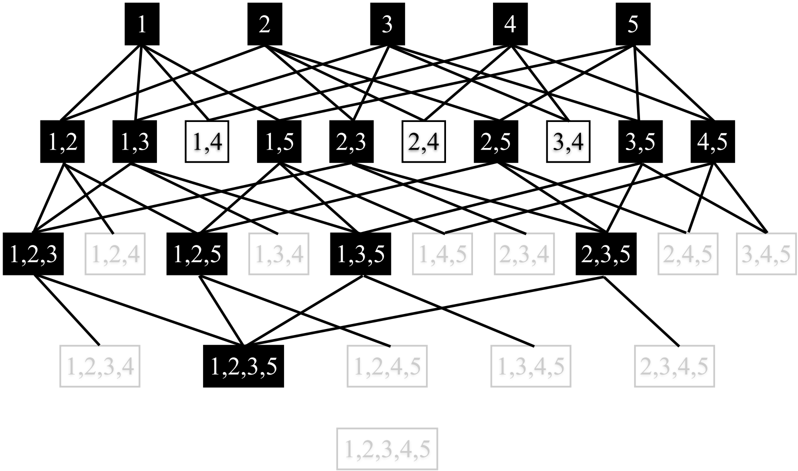
Example of Active Set Method

Optimize active groups and sub-optimal boundary groups.



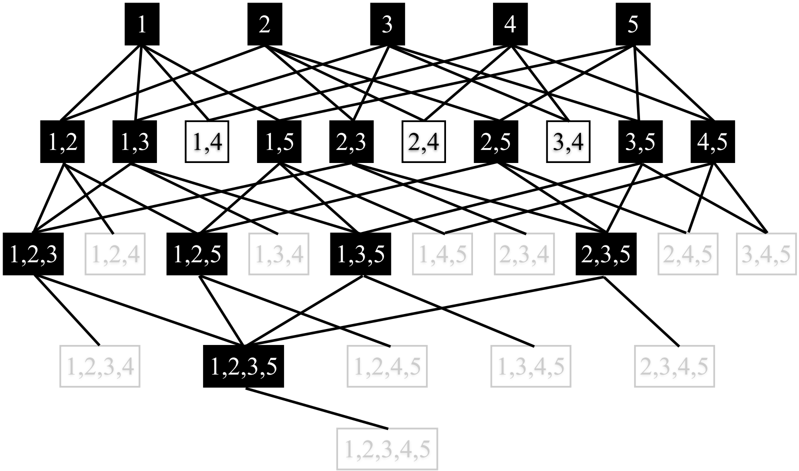
Example of Active Set Method

Find new **active groups**.



Example of Active Set Method

No new boundary groups, so we are done.



Example of Active Set Method

- We only considered:
  - 4 of 10 possible threeway interactions.
  - 1 of 5 possible fourway interactions.
  - No fiveway interactions.

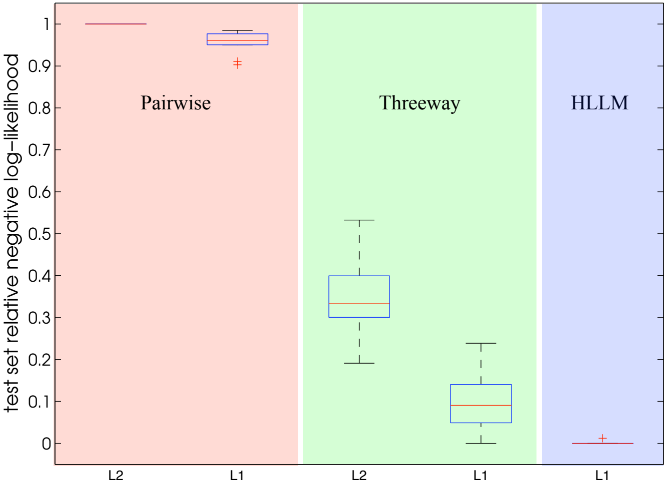
Example of Active Set Method

- We only considered:
  - 4 of 10 possible threeway interactions.
  - 1 of 5 possible fourway interactions.
  - No fiveway interactions.
- The heuristic can reduce the space exponentially.

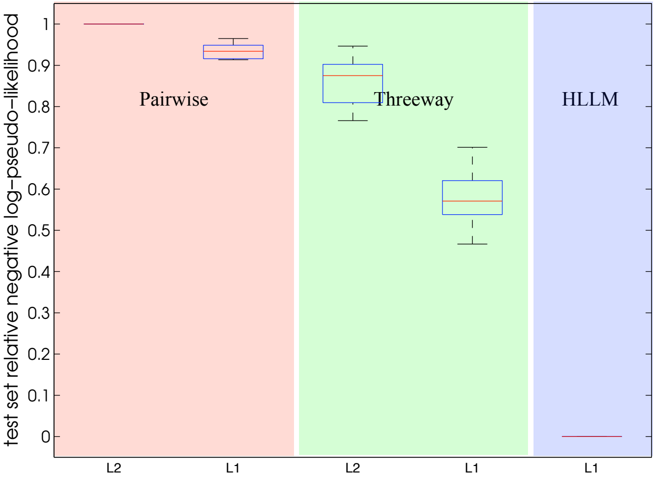
Example of Active Set Method

- We only considered:
  - 4 of 10 possible threeway interactions.
  - 1 of 5 possible fourway interactions.
  - No fiveway interactions.
- The heuristic can reduce the space exponentially.
- In practice, do the heuristic and higher-order potentials help?

Flow Cytometry Data



Traffic Flow Data



Structured Sparsity for Hierarchical Constraints

- We now turn to the overlapping group ℓ_1-regularization problem,

$$\min_x f(x) + \lambda \sum_g ||x_g||,$$

  where the groups g may not overlap.

- Non-smooth is regularizer is not simple.
- But we can use that each term is simple.

Converting to a Constrained Problem

- Constrained re-formulation:

$$\min_{\|x_g\| \leq t_g} f(x) + \lambda \sum_g t_g.$$

Converting to a Constrained Problem

- Constrained re-formulation:

$$\min_{\|x_g\| \leq t_g} f(x) + \lambda \sum_g t_g.$$

- We can efficiently project onto each constraint.
- But projections aren't independent since groups overlap.

Converting to a Constrained Problem

- Constrained re-formulation:

$$\min_{\|x_g\| \leq t_g} f(x) + \lambda \sum_g t_g.$$

- We can efficiently project onto each constraint.
- But projections aren't independent since groups overlap.
- We want the projection onto the intersection of simple sets.

Cyclic Projection Algorithms

Projecting onto the intersection of simple sets is a classic problem:

Cyclic Projection Algorithms

Projecting onto the intersection of simple sets is a classic problem:

- Cyclically projecting onto two subspaces converges to the projection onto their intersections. [von Neumann, 1933]

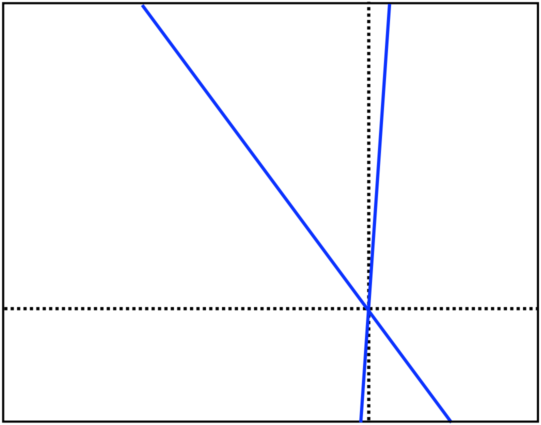
von Neumann's Result

Definition 13.7: If $\phi_1, \phi_2, \ldots$ is a sequence $\sum$ of s.v. operators, if f is an element of $\prod_{n=1}^{\infty} D(\phi_n)$ such that $\lim_{n\to\infty} \phi_n f$ exists, and if D is the set of all such elements f, then $\sum$ is said to have a limit ϕ over D, and, for $f \in D = D(\phi)$, $\phi f = \lim_{n\to\infty} \phi_n f$.

THEOREM 13.7. If $E = P_M$ and $F = P_N$, then the sequence $\sum_1$ of operators $E, FE, EFE, FEFE, \ldots$ has a limit G, the sequence $\sum_2$: $F, EF, FEF, \ldots$ has the same limit G, and $G = P_{MN}$. (The condition $EF = FE$ need not hold.)

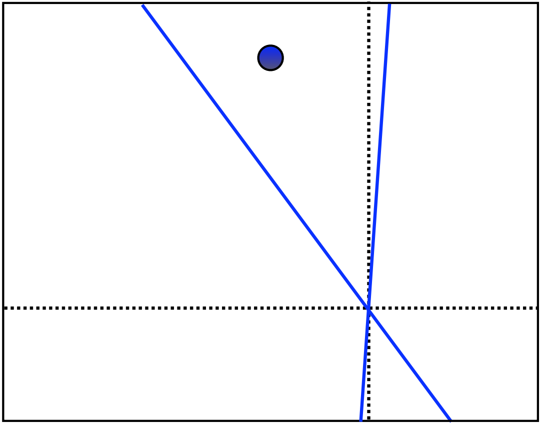
Proof: Let A_n be the n^{th} operator of the sequence $\sum_1$. Then $(A_m f, A_n g) = (A_{m+n-\varepsilon} f, g)$, where $\varepsilon = 1$ if m and n have the same parity and $\varepsilon = 0$ if m and n have opposite parity. It must be shown that if f is any element of S, then $\lim_{n\to\infty} A_n f$ exists. But $\|A_m f - A_n f\|^2 = (A_m f - A_n f, A_m f - A_n f) =$

Take two intersecting subspaces.



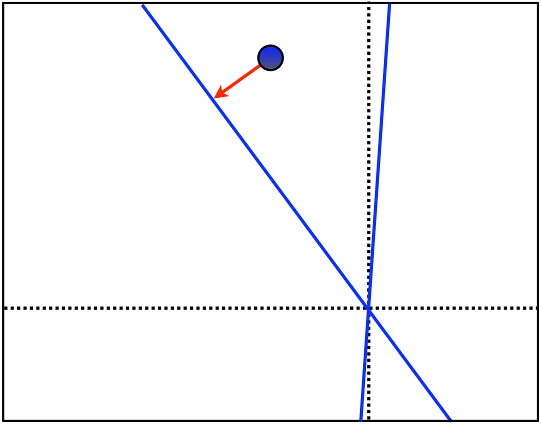
von Neumann's Result

We want to project a point onto their intersection.



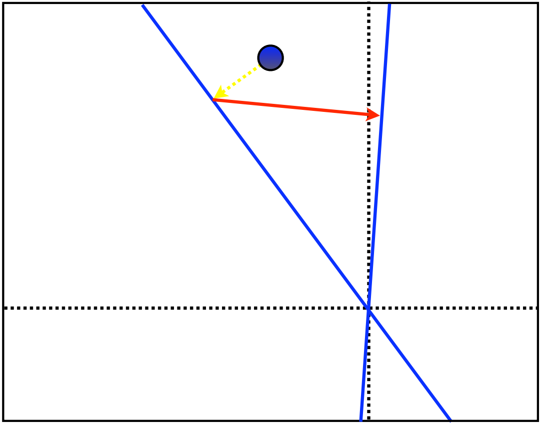
von Neumann's Result

Project onto subspace 1.



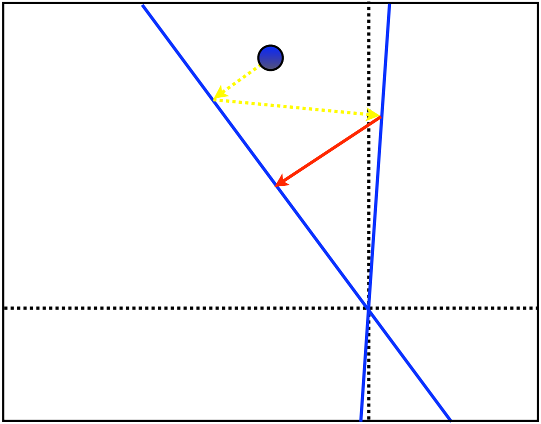
von Neumann's Result

Project onto subspace 2.

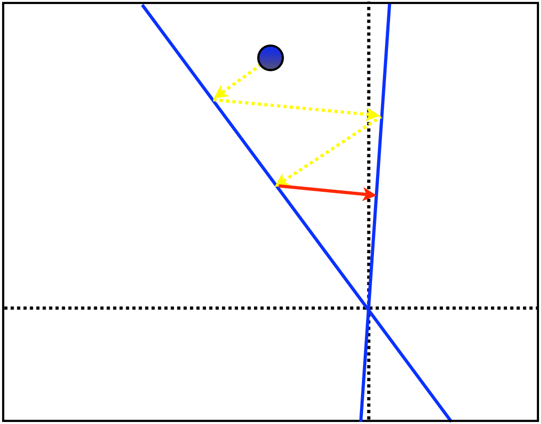


von Neumann's Result

Project onto subspace 1.

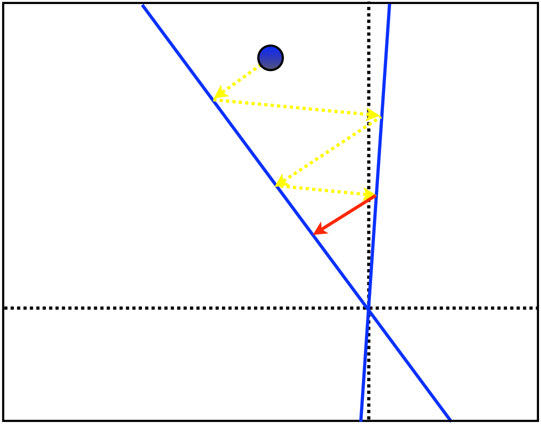


Project onto subspace 2.



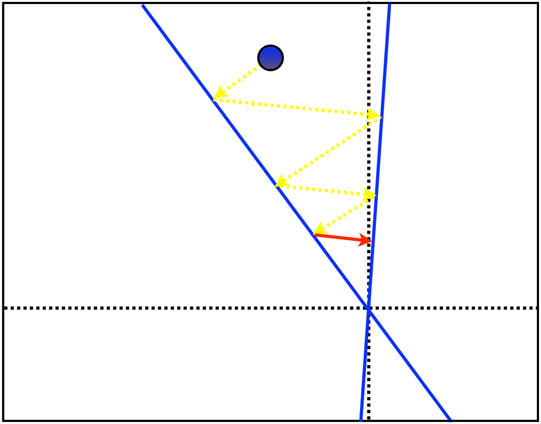
von Neumann's Result

Project onto subspace 1.



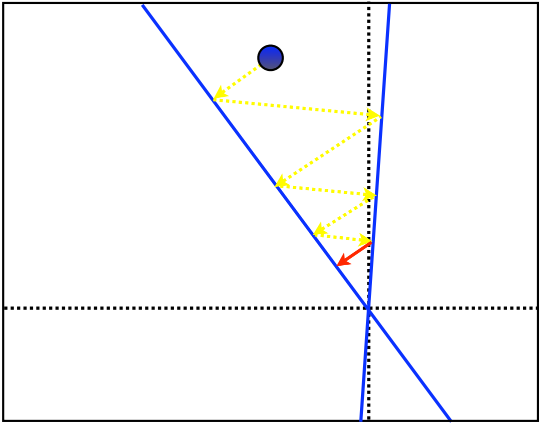
von Neumann's Result

Project onto subspace 2.



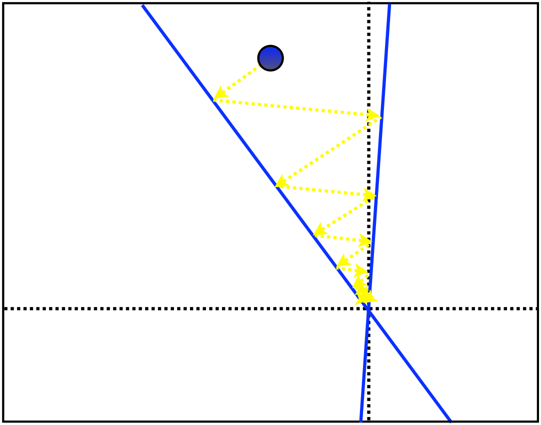
von Neumann's Result

Project onto subspace 1.



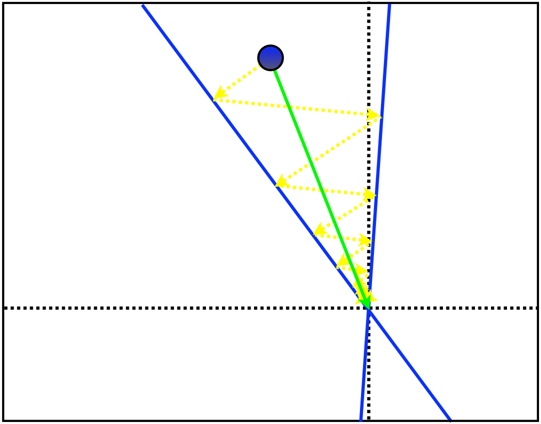
von Neumann's Result

And keep going...



von Neumann's Result

The limit is the projection onto the intersection.



Cyclic Projection Algorithms

Projecting onto the intersection of simple sets is a classic problem:

- Cyclically projecting onto two subspaces converges to the projection onto their intersections. [von Neumann, 1933]

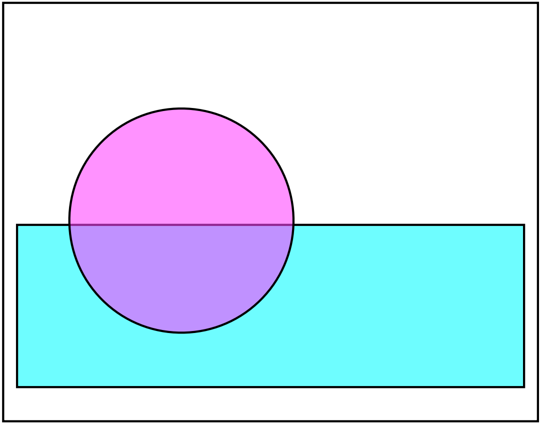
Cyclic Projection Algorithms

Projecting onto the intersection of simple sets is a classic problem:

- Cyclically projecting onto two subspaces converges to the projection onto their intersections. [von Neumann, 1933]
- Cyclically projecting onto convex sets converges to a point in their interesections. [Bregman, 1965]

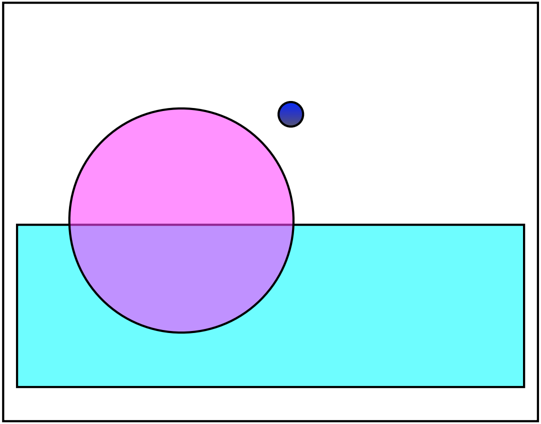
Bregman's Algorithm

We have an arbitrary number of convex sets.



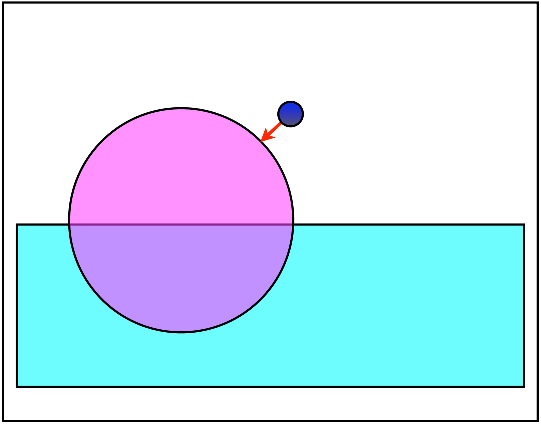
Bregman's Algorithm

Start with some initial point.



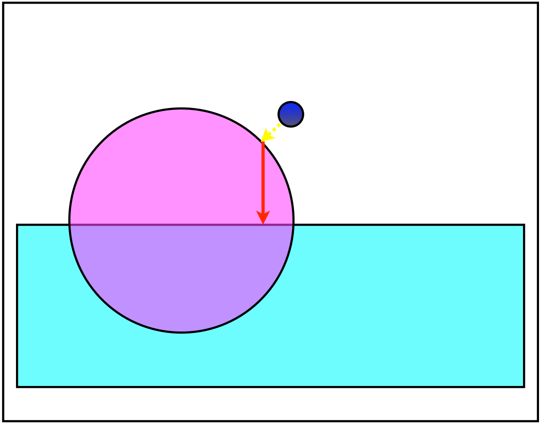
Bregman's Algorithm

Project onto convex set 1.



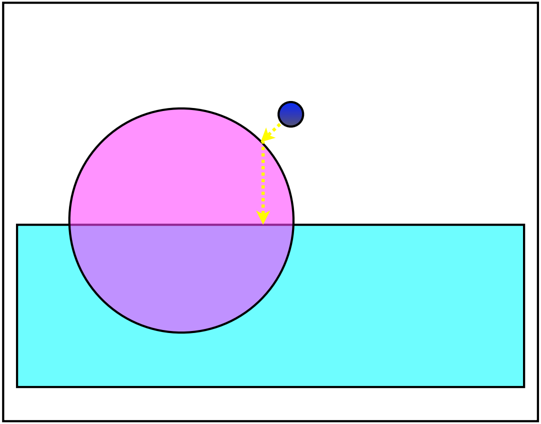
Bregman's Algorithm

Project onto convex set 2.



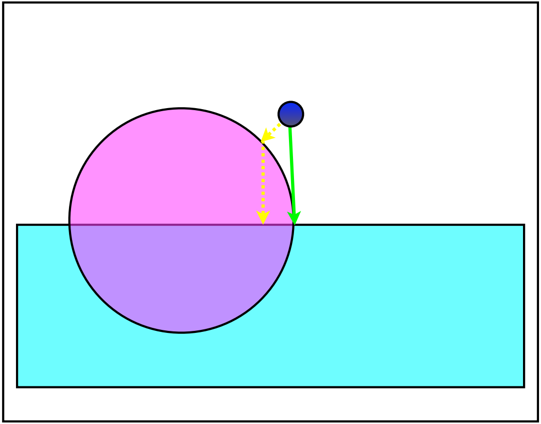
Bregman's Algorithm

The limit is a point in the intersection.



Bregman's Algorithm

In general, the limit is not the projection.



Cyclic Projection Algorithms

Projecting onto the intersection of simple sets is a classic problem:

- Cyclically projecting onto two subspaces converges to the projection onto their intersections. [von Neumann, 1933]
- Cyclically projecting onto convex sets converges to a point in their interesections. [Bregman, 1965]

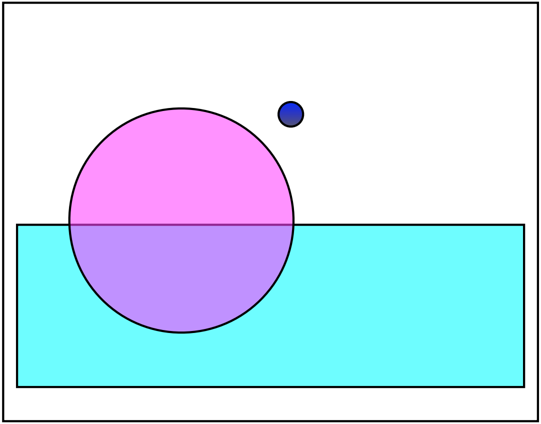
Cyclic Projection Algorithms

Projecting onto the intersection of simple sets is a classic problem:

- Cyclically projecting onto two subspaces converges to the projection onto their intersections. [von Neumann, 1933]
- Cyclically projecting onto convex sets converges to a point in their interesections. [Bregman, 1965]
- A simple modification makes the method converge to the projection onto their intersections. [Dykstra, 1983]

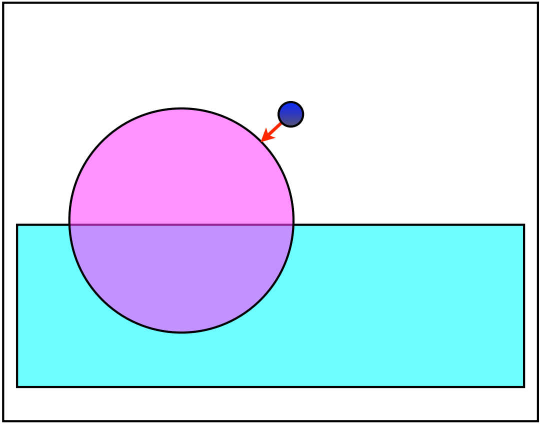
Dykstra's Algorithm

We want to project a point onto the intersection of convex sets.



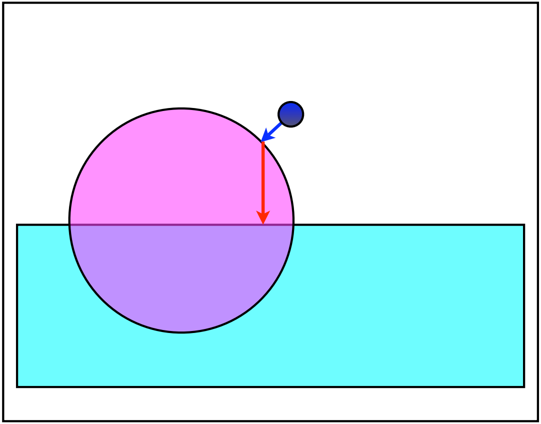
Dykstra's Algorithm

Project onto convex set 1, and store the difference.



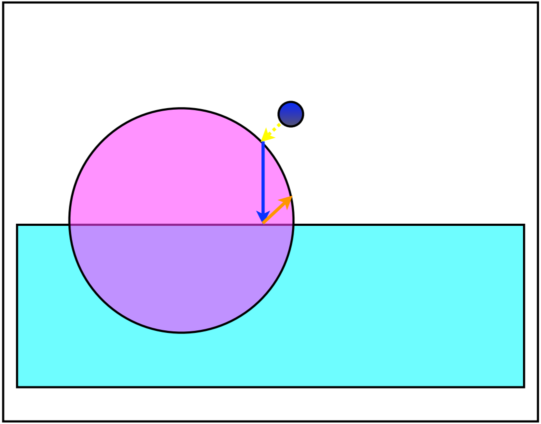
Dykstra's Algorithm

Project onto convex set 2, and store the difference.



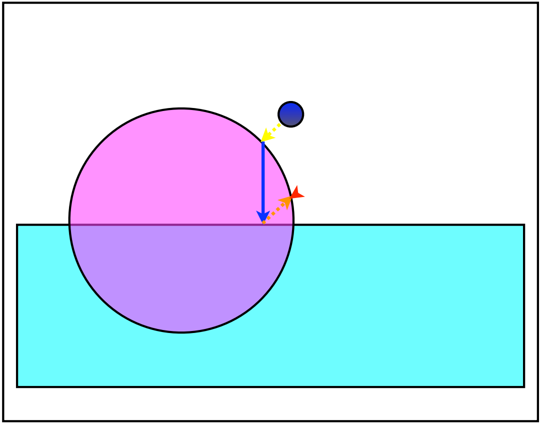
Dykstra's Algorithm

Remove the difference from projecting on convex set 1.



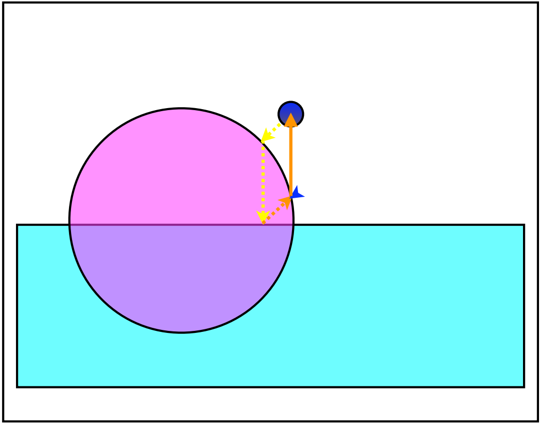
Dykstra's Algorithm

Project onto convex set 1, and store the difference.



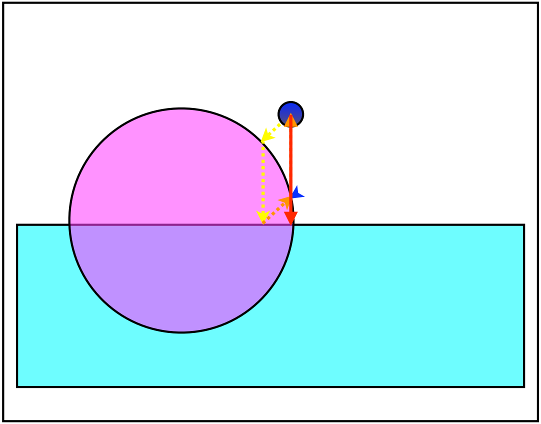
Dykstra's Algorithm

Remove the difference from projecting on convex set 2.



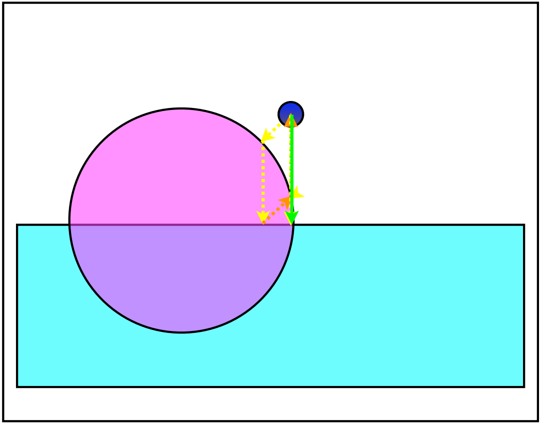
Dykstra's Algorithm

Project onto convex set 2, and store the difference.



Dykstra's Algorithm

The limit is the projection onto the intersection.



Cyclic Projection Algorithms

Projecting onto the intersection of simple sets is a classic problem:

- Cyclically projecting onto two subspaces converges to the projection onto their intersections. [von Neumann, 1933]
- Cyclically projecting onto convex sets converges to a point in their interesections. [Bregman, 1965]
- A simple modification makes the method converge to the projection onto their intersections. [Dykstra, 1983]
- For polyhedral sets, Dykstra's algorithm has a linear convergence rate. [Deutsch and Hundal, 1994]

Cyclic Projection Algorithms

Projecting onto the intersection of simple sets is a classic problem:

- Cyclically projecting onto two subspaces converges to the projection onto their intersections. [von Neumann, 1933]
- Cyclically projecting onto convex sets converges to a point in their interesections. [Bregman, 1965]
- A simple modification makes the method converge to the projection onto their intersections. [Dykstra, 1983]
- For polyhedral sets, Dykstra's algorithm has a linear convergence rate. [Deutsch and Hundal, 1994]
- Proximal versions of Dykstra's algorithm have recently been developed. [Bauschke and Combettes, 2008]

Exact and Inexact Proximal-Gradient Methos

- We can efficiently compute the proximity operator for:
  1. ℓ_1-*Regularization*.
  2. *Group* ℓ_1-*Regularization*.
  3. *Lower and upper bound constraints*.
  4. *Hyper-plane and half-space constraints*.
  5. *Simplex constraints*.
  6. *Euclidean cone constraints*.

Exact and Inexact Proximal-Gradient Methos

- We can efficiently compute the proximity operator for:
  1. *ℓ_1-Regularization*.
  2. *Group ℓ_1-Regularization*.
  3. *Lower and upper bound constraints*.
  4. *Hyper-plane and half-space constraints*.
  5. *Simplex constraints*.
  6. *Euclidean cone constraints*.
- We can efficiently approximate the proximity operator for:
  1. *Overlapping group ℓ_1-regularization with general groups*.

Exact and Inexact Proximal-Gradient Methos

- We can efficiently compute the proximity operator for:
  1. *ℓ_1-Regularization*.
  2. *Group ℓ_1-Regularization*.
  3. *Lower and upper bound constraints*.
  4. *Hyper-plane and half-space constraints*.
  5. *Simplex constraints*.
  6. *Euclidean cone constraints*.
- We can efficiently approximate the proximity operator for:
  1. *Overlapping group ℓ_1-regularization with general groups*.
  2. *Total-variation regularization and generalizations like the graph-guided fused-LASSO*.
  3. *Nuclear-norm regularization and other regularizers on the singular values of matrices*.
  4. *Positive semi-definite cone*.
  5. *Combinations of simple functions*.

Convergence Rate of Inexact Proximal-Gradient

- Can inexact proximal-gradient methods achieve the fast rates?

Convergence Rate of Inexact Proximal-Gradient

- Can inexact proximal-gradient methods achieve the fast rates?
- Exact proximal-gradient methods have

$$f(x^k) - f(x^*) = O((1 - \mu/L)^{2k}).$$

(the same convergence rate as gradient methods)

Convergence Rate of Inexact Proximal-Gradient

- Can inexact proximal-gradient methods achieve the fast rates?
- Exact proximal-gradient methods have

$$f(x^k) - f(x^*) = O((1 - \mu/L)^{2k}).$$

(the same convergence rate as gradient methods)

Proposition. *If the sequences $\{||e_k||\}$ and $\{\sqrt{\varepsilon_k}\}$ are in $O(\rho^k)$ for $\rho < (1 - \mu/L)$ then the basic proximal-gradient method achieves*

$$f(x^k) - f(x^*) = O((1 - \mu/L)^{2k}).$$

Convergence Rate of Inexact Proximal-Gradient

- Can inexact proximal-gradient methods achieve the fast rates?
- Exact proximal-gradient methods have

$$f(x^k) - f(x^*) = O((1-\mu/L)^{2k}).$$

(the same convergence rate as gradient methods)

Proposition. *If the sequences $\{||e_k||\}$ and $\{\sqrt{\varepsilon_k}\}$ are in $O(\rho^k)$ for $\rho < (1-\mu/L)$ then the basic proximal-gradient method achieves*

$$f(x^k) - f(x^*) = O((1-\mu/L)^{2k}).$$

- We show analogous results for accelerated proximal-gradient methods, including when $\mu = 0$. [Schmidt et al., 2011]

Outline

1. Sparsity
2. Group Sparsity
3. Structured Sparsity
4. Big-N Problems

Context: Machine Learning for “Big Data”

- **Large-scale machine learning**: large N, large P
  - N: number of observations (inputs)
  - P: dimension of each observation

Context: Machine Learning for “Big Data”

- **Large-scale machine learning**: large N, large P
  - N: number of observations (inputs)
  - P: dimension of each observation
- **Regularized empirical risk minimization**:

$$\min_{x \in \mathbb{R}^P} \frac{1}{N} \sum_{i=1}^{N} f_i(x) \quad + \quad \lambda r(x)$$

$$\text{data fitting term} \quad + \quad \text{regularizer}$$

- Applications to any data-oriented field:
  - Vision, bioinformatics, speech, natural language, web.

Context: Machine Learning for "Big Data"

- **Large-scale machine learning**: large N, large P
  - N: number of observations (inputs)
  - P: dimension of each observation
- **Regularized empirical risk minimization**:

$$\underbrace{\min_{x \in \mathbb{R}^P} \frac{1}{N} \sum_{i=1}^{N} f_i(x)}_{\text{data fitting term}} \quad + \quad \underbrace{\lambda r(x)}_{\text{regularizer}}$$

- Applications to any data-oriented field:
  - Vision, bioinformatics, speech, natural language, web.
- Main practical challenges:
  - Designing/learning good features.
  - Efficiently solving the problem when N or P are very large.

Big-N Problems

- We want to minimize the sum of a finite set of smooth functions:

$$\min_{x \in \mathbb{R}^P} f(x) := \frac{1}{N} \sum_{i=1}^{N} f_i(x).$$

Big-N Problems

- We want to minimize the sum of a finite set of smooth functions:

$$\min_{x \in \mathbb{R}^P} f(x) := \frac{1}{N} \sum_{i=1}^{N} f_i(x).$$

- We are interested in cases where N is very large.

Big-N Problems

- We want to minimize the sum of a finite set of smooth functions:

$$\min_{x \in \mathbb{R}^P} f(x) := \frac{1}{N} \sum_{i=1}^{N} f_i(x).$$

- We are interested in cases where N is very large.
- Simple example is ℓ_2-regularized least-squares,

$$f_i(x) := (a_i^T x - b_i)^2 + \frac{\lambda}{2} \|x\|^2.$$

- Other examples include any ℓ_2-regularized convex loss:
  - logistic regression, Huber regression, smooth SVMs, CRFs, etc.

Stochastic vs. Deterministic Gradient Methods

- We consider minimizing $f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x)$.

Stochastic vs. Deterministic Gradient Methods

- We consider minimizing $f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x)$.
- Deterministic gradient method [Cauchy, 1847]:

$$x_{t+1} = x_t - \alpha_t f'(x_t) = x_t - \frac{\alpha_t}{N}\sum_{i=1}^{N} f_i'(x_t).$$

  - Linear convergence rate: $O(\rho^t)$.
  - Iteration cost is linear in N.
  - Quasi-Newton methods still require $O(N)$.

Stochastic vs. Deterministic Gradient Methods

- We consider minimizing $f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x)$.
- Deterministic gradient method [Cauchy, 1847]:

$$x_{t+1} = x_t - \alpha_t f'(x_t) = x_t - \frac{\alpha_t}{N}\sum_{i=1}^{N} f_i'(x_t).$$

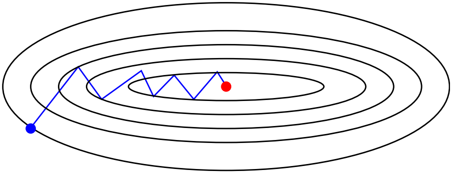
  - Linear convergence rate: $O(\rho^t)$.
  - Iteration cost is linear in N.
  - Quasi-Newton methods still require $O(N)$.
- Stochastic gradient method [Robbins & Monro, 1951]:
  - Random selection of $i(t)$ from $\{1, 2, \ldots, N\}$.

$$x_{t+1} = x_t - \alpha_t f_{i(t)}(x_t).$$

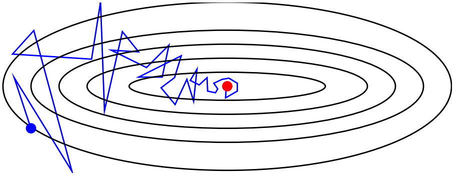
  - Iteration cost is independent of N.
  - Sublinear $O(1/t)$ convergence rate.

Stochastic vs. Deterministic Gradient Methods

- We consider minimizing $g(x) = \frac{1}{N}\sum_{i=1}^{n} f_i(x)$.
- Deterministic gradient method [Cauchy, 1847]:

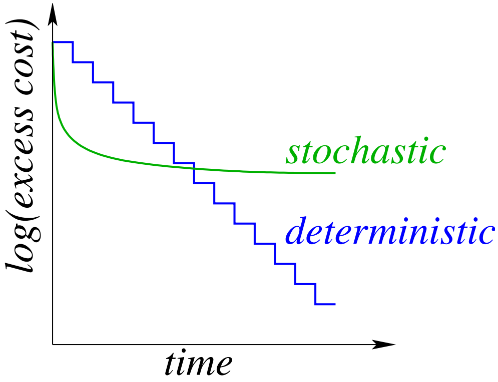


- Stochastic gradient method [Robbins & Monro, 1951]:



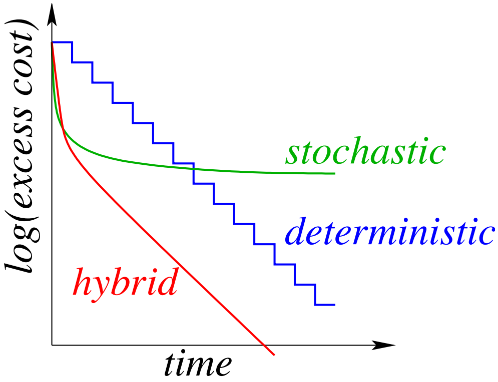
Motivation for New Methods

- FG method has $O(N)$ cost with $O(\rho^k)$ rate.
- SG method has $O(1)$ cost with $O(1/k)$ rate.



Motivation for New Methods

- FG method has $O(N)$ cost with $O(\rho^k)$ rate.
- SG method has $O(1)$ cost with $O(1/k)$ rate.



- Goal is $O(1)$ cost with $O(\rho^k)$ rate.

Prior Work on Speeding up SG Methods

A variety of methods have been proposed to speed up SG methods:

- **Momentum, gradient/iterate averaging**
  - Polyak & Juditsky (1992), Tseng (1998), Kushner & Yin (2003) Nesterov (2009), Xiao (2010), Hazan & Kale (2011), Rakhlin et al. (2012)
- **Stochastic version of deterministic methods**
  - Bordes et al. (2009), Sunehag et al. (2009), Ghadimi and Lan (2010), Martens (2010), Xiao (2010), Duchi et al. (2011)

Prior Work on Speeding up SG Methods

A variety of methods have been proposed to speed up SG methods:

- **Momentum, gradient/iterate averaging**
  - Polyak & Juditsky (1992), Tseng (1998), Kushner & Yin (2003) Nesterov (2009), Xiao (2010), Hazan & Kale (2011), Rakhlin et al. (2012)
- **Stochastic version of deterministic methods**
  - Bordes et al. (2009), Sunehag et al. (2009), Ghadimi and Lan (2010), Martens (2010), Xiao (2010), Duchi et al. (2011)
- None of these methods improve on the $O(1/t)$ rate

Prior Work on Speeding up SG Methods

Existing linear convergence results:

- **Constant step-size SG, accelerated SG**
  - Kesten (1958), Delyon and Juditsky (1993), Nedic and Bertsekas (2000)
  - Linear convergence but only up to a fixed tolerance
- **Hybrid methods, incremental average gradient**
  - Bertsekas (1997), Blatt et al. (2007), Friedlander and Schmidt (2012)
  - Linear rate but iterations make full passes through the data

Prior Work on Speeding up SG Methods

Existing linear convergence results:

- **Constant step-size SG, accelerated SG**
  - Kesten (1958), Delyon and Juditsky (1993), Nedic and Bertsekas (2000)
  - Linear convergence but only up to a fixed tolerance
- **Hybrid methods, incremental average gradient**
  - Bertsekas (1997), Blatt et al. (2007), Friedlander and Schmidt (2012)
  - Linear rate but iterations make full passes through the data
- **Special Problems Classes**
  - Collins et al. (2008), Strohmer & Vershynin (2009), Schmidt and Le Roux (2012), Shalev-Shwartz and Zhang (2012)
  - Linear rate but limited choice for the f_i's

Stochastic Average Gradient

- **Is it possible to have a general linearly convergent algorithm with iteration cost independent of N?**

Stochastic Average Gradient

- **Is it possible to have a general linearly convergent algorithm with iteration cost independent of N?**
  - YES!

Stochastic Average Gradient

- **Is it possible to have a general linearly convergent algorithm with iteration cost independent of N?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select $i(t)$ from $\{1, 2, \ldots, n\}$ and compute $f'_{i(t)}(x^t)$,

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} f'_i(x^t)$$

Stochastic Average Gradient

- **Is it possible to have a general linearly convergent algorithm with iteration cost independent of N?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select $i(t)$ from $\{1, 2, \ldots, n\}$ and compute $f'_{i(t)}(x^t)$,

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} f'_i(x^t)$$

Stochastic Average Gradient

- **Is it possible to have a general linearly convergent algorithm with iteration cost independent of N?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select $i(t)$ from $\{1, 2, \dots, n\}$ and compute $f'_{i(t)}(x^t)$,

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} f'_i(x_i^t)$$

    - **Memory**: x_i^t is the last iterate where i was selected.

Stochastic Average Gradient

- **Is it possible to have a general linearly convergent algorithm with iteration cost independent of N?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select $i(t)$ from $\{1, 2, \ldots, n\}$ and compute $f'_{i(t)}(x^t)$,

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} f'_i(x_i^t)$$

    - **Memory**: x_i^t is the last iterate where i was selected.
  - Assumes gradients of other examples don't change.
  - Assumption becomes accurate as $||x^{t+1} - x^t|| \to 0$.

Stochastic Average Gradient

- **Is it possible to have a general linearly convergent algorithm with iteration cost independent of N?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select $i(t)$ from $\{1, 2, \dots, n\}$ and compute $f'_{i(t)}(x^t)$,

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} f'_i(x_i^t)$$

    - **Memory**: x_i^t is the last iterate where i was selected.
  - Assumes gradients of other examples don't change.
  - Assumption becomes accurate as $||x^{t+1} - x^t|| \to 0$.
  - Stochastic variant of increment average gradient (IAG).

    [Blatt et al. 2007]
  - $O(NP)$ memory requirements reduced to $O(N)$ for many problems.

Convergence Rate of SAG

Assume only that:

- f_i is convex, f_i' is $L-$continuous, f is μ-strongly convex.

Convergence Rate of SAG

Assume only that:

- f_i is convex, f_i' is $L-$continuous, f is μ-strongly convex.

Theorem. *With $\alpha = \frac{1}{16L}$ the SAG iterations satisfy*

$$\mathbb{E}[f(x^t) - f(x^*)] = O\left(\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^t\right).$$

- **Convergence rate of $O(\rho^t)$ with cost of $O(1)$** (true for $\alpha \leq \frac{1}{16L}$).

Convergence Rate of SAG

Assume only that:

- f_i is convex, f_i' is $L-$continuous, f is μ-strongly convex.

Theorem. *With $\alpha = \frac{1}{16L}$ the SAG iterations satisfy*

$$\mathbb{E}[f(x^t) - f(x^*)] = O\left(\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^t\right).$$

- **Convergence rate of $O(\rho^t)$ with cost of $O(1)$** (true for $\alpha \leq \frac{1}{16L}$).
- This rate is "very fast":
  - Well-conditioned problems: constant non-trivial reduction per pass:

$$\left(1 - \frac{1}{8N}\right)^N \leq \exp\left(-\frac{1}{8}\right) = 0.8825.$$

  - Badly-conditioned problems, almost same as deterministic method. (deterministic has rate $\left(1 - \frac{\mu}{L}\right)^{2t}$ with $\alpha = \frac{1}{L}$, but N times slower)

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$ (*rcv1 data set):*

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$ (*rcv1 data set):*
  - *Gradient method has rate* $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$ (*rcv1 data set):*
  - *Gradient method has rate* $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - *Accelerated gradient method has rate* $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$ (*rcv1 data set):*
  - *Gradient method has rate* $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - *Accelerated gradient method has rate* $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - *SAG (N iterations) has rate* $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$ (*rcv1 data set):*
  - *Gradient method has rate* $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - *Accelerated gradient method has rate* $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - *SAG (N iterations) has rate* $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible deterministic method:* $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$ (*rcv1 data set*):
  - *Gradient method has rate* $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - *Accelerated gradient method has rate* $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - *SAG (N iterations) has rate* $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible deterministic method:* $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.
- *SAG beats two lower bounds:*
  - *Stochastic gradient bound (of $O(1/t)$).*
  - *Deterministic gradient bound (for typical L, μ, and N).*

Convergence Rate in Convex Case

Assume only that:

- f_i is convex, f_i' is $L-$continuous, some x^* exists.

Convergence Rate in Convex Case

Assume only that:

- f_i is convex, f_i' is $L-$continuous, some x^* exists.

***Theorem**. With $\alpha_t \leqslant \frac{1}{16L}$ the SAG iterations satisfy*

$$\mathbb{E}[f(x^t) - f(x^*)] = O(1/N)$$

- **Faster than SG lower bound of $O(1/\sqrt{N})$.**

Convergence Rate in Convex Case

Assume only that:

- f_i is convex, f_i' is $L-$continuous, some x^* exists.

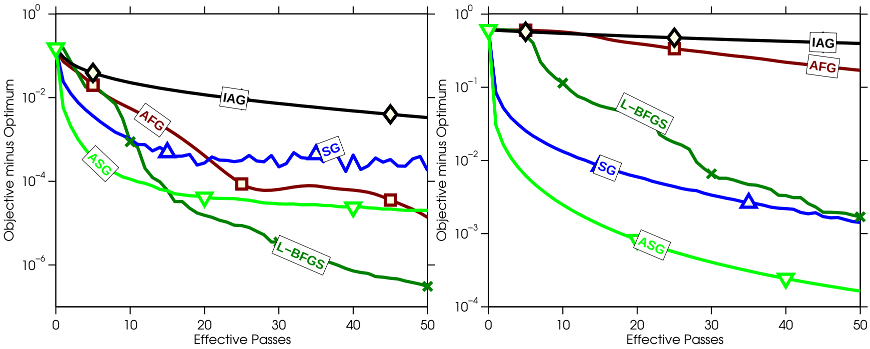
***Theorem**. With $\alpha_t \leqslant \frac{1}{16L}$ the SAG iterations satisfy*

$$\mathbb{E}[f(x^t) - f(x^*)] = O(1/N)$$

- **Faster than SG lower bound of $O(1/\sqrt{N})$.**
- Same algorithm and step-size as strongly-convex case:
  - Algorithm is adaptive to strong-convexity.
  - Faster convergence rate if μ is locally bigger around x^*.

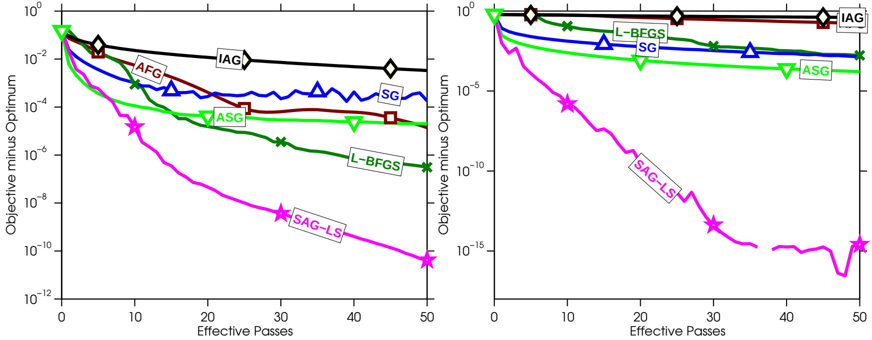
Comparing FG and SG Methods

- quantum ($n = 50000$, $p = 78$) and rcv1 ($n = 697641$, $p = 47236$)



SAG Compared to FG and SG Methods

- quantum ($n = 50000$, $p = 78$) and rcv1 ($n = 697641$, $p = 47236$)



Conclusion and Open Problems

- Fast theoretical convergence using the ‘sum’ structure.

Conclusion and Open Problems

- Fast theoretical convergence using the ‘sum’ structure.
- Simple algorithm, empirically better than theory predicts.

Conclusion and Open Problems

- Fast theoretical convergence using the ‘sum’ structure.
- Simple algorithm, empirically better than theory predicts.
- Allows adaptive step-size and approximate optimality measures.

Conclusion and Open Problems

- Fast theoretical convergence using the 'sum' structure.
- Simple algorithm, empirically better than theory predicts.
- Allows adaptive step-size and approximate optimality measures.
- Subsequent work:
  - Constrained and non-smooth problems.

    [Mairal, 2013, Wong et al., 2013]
  - Memory-free methods.

    [Johnson and Zhang, 2013, Zhang et al., 2013]
  - Non-uniform sampling.

    [Schmidt et al., 2013]

Conclusion and Open Problems

- Fast theoretical convergence using the ‘sum’ structure.
- Simple algorithm, empirically better than theory predicts.
- Allows adaptive step-size and approximate optimality measures.
- Subsequent work:
  - Constrained and non-smooth problems.

    [Mairal, 2013, Wong et al., 2013]
  - Memory-free methods.

    [Johnson and Zhang, 2013, Zhang et al., 2013]
  - Non-uniform sampling.

    [Schmidt et al., 2013]
- Thanks for coming!