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

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

- 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:

\[ \min_x \sum_{i=1}^{N} f_i(x). \]
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.
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 $\ell_1$-Regularization

- Last day of Master’s: try all features with $\ell_2$-Regularization:

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

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

$\ell_1$-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?
Optimizing with $\ell_1$-Regularization

- Last day of Master’s: try **all features** with $\ell_2$-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 $\ell_1$-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 $\ell_1$-Regularization

- Last day of Master’s: try all features with $\ell_2$-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 $\ell_1$-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?
We can re-write the regularized problem

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

as a constrained problem

$$\min \ f(x). \quad \|x\|_p \leq \tau$$
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 \( r(x) = \lambda \|x\|_2^2 \): 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 \( F'(x^*) = 0 \).
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 $r(x) = \lambda \|x\|_1$:

- We need $f'(x) = -\lambda \bar{d}$, for some sub-gradient $\bar{d}$ 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 \( d = 0 \).
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 \( x^\ast \) has gradient \( F'(x^\ast) = 0 \).

  When \( r(x) = \lambda \|x\|^2 \):
  - We need \( f'(x) = -\lambda x \).

- When \( F \) is convex and smooth,
  its minimizer \( x^\ast \) 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 $r(x) = \lambda \|x\|_2^2$:
  - 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 $\|x\|_1$. 

![Diagram showing the relationship between $F(x)$ and $r(x)$](image-url)
We want to optimize a smooth function with $\ell_1$-Regularization:

$$\min_x f(x) + \lambda \|x\|_1.$$
We want to optimize a smooth function with $\ell_1$-Regularization:

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

With $\ell_2$-Regularization, can use quasi-Newton methods.

http://www.di.ens.fr/~mschmidt/Software/minFunc.html
We want to optimize a smooth function with $\ell_1$-Regularization:

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

With $\ell_2$-Regularization, can use quasi-Newton methods.

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

The non-smooth $\ell_1$-regularizer breaks these methods.
We want to optimize a smooth function with $\ell_1$-Regularization:

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

With $\ell_2$-Regularization, can use quasi-Newton methods. 

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

The non-smooth $\ell_1$-regularizer breaks these methods.

But the regularizer is separable: $\|x\|_1 = \sum_j |x_j|.$
Optimization with $\ell_1$-Regularization

- We want to optimize a smooth function with $\ell_1$-Regularization:
  \[ \min_x f(x) + \lambda \|x\|_1. \]

- With $\ell_2$-Regularization, can use quasi-Newton methods.
  [http://www.di.ens.fr/~mschmidt/Software/minFunc.html](http://www.di.ens.fr/~mschmidt/Software/minFunc.html)

- The non-smooth $\ell_1$-regularizer breaks these methods.

- But the regularizer is separable: $\|x\|_1 = \sum_j |x_j|$.

- Can we extend quasi-Newton methods using this property?
Consider splitting each variable into a positive and negative part:

\[ x = x^+ - x^-, \text{ with } x^+ \geq 0, x^- \geq 0. \]
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 as a smooth objective with non-negative constraints:

\[
\min_{x} f(x) + \lambda \|x\|_1,
\]

as a smooth objective with non-negative constraints:

\[
\min_{x^+ \geq 0, x^- \geq 0} F(x) = f(x^+ - x^-) + \lambda \sum_{j} [x^+_j + x^-_j]
\]
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:

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

- Use methods for smooth bound-constrained optimization.
Classic bound-constrained optimizer is gradient projection:

\[ x^{k+1} \leftarrow [x^k - \alpha F'(x^k)]^+ . \]
Classic bound-constrained optimizer is gradient projection:

\[ x^{k+1} ← [x^k − \alpha F'(x^k)]^+. \]
Classic bound-constrained optimizer is gradient projection:

$$x^{k+1} \leftarrow [x^k - \alpha F'(x^k)]^+.$$
Classic bound-constrained optimizer is gradient projection:

\[ x^{k+1} \leftarrow [x^k - \alpha F'(x^k)]^+. \]
Classic bound-constrained optimizer is gradient projection:

\[ x^{k+1} \leftarrow [x^k - \alpha F'(x^k)]^+ . \]
Classic bound-constrained optimizer is gradient projection:

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

\[ x^{k+1} \leftarrow [x^k - \alpha F'(x^k)]^+. \]
Classic bound-constrained optimizer is gradient projection:

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

Convergence properties similar to gradient method.
Can we use a [quasi-]Newton step?

\[
x^{k+1} \leftarrow [x^k - \alpha H^{-1}_k F'(x^k)]^+,
\]
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?

$$x^{k+1} \leftarrow [x^k - \alpha H_k^{-1} F'(x^k)]^+, \quad \text{No, this does not work!}$$
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!
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!
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!
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!
For separable problems we can fix this by restricting $H_k$. 

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \alpha [D_k] - 1 F'(\mathbf{x}_k) + [\text{Birgin et al., 2000, Figueiredo et al., 2007}]

But is this too restrictive?

Only need $H_k$ diagonal with respect to:

$A \equiv \{ i | x_{ki} \leq \epsilon \text{ and } F'_i(\mathbf{x}_k) > 0 \}$

[\text{Gafni & Bertsekas, 1984}]

Re-arranging, we need $H_k = \begin{bmatrix} D_k & 0 \\ 0 & \bar{H}_k \end{bmatrix}$

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

Mark Schmidt

Opening up the Black Box
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]
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?
For **separable** problems we can fix this by restricting $H_k$.

Use a **diagonal** matrix $D_k$:

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

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

But is this too restrictive?

Only need $H_k$ **diagonal** with respect to:

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

[Ref: Gafni & Bertsekas, 1984]
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

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

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

  [Gafni & Bertsekas, 1984]

- Re-arranging, we need

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

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

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?
The original problem:

$$\min_x F(x) = f(x) + \lambda \|x\|_1.$$
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 \(-z^k\).
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 $\ell_1$-Regularization problems,

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

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

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

We can even try a Newton-like version:

$$
x_{k+1} = x_k - \alpha \left[ H_k \right]^{-1} z_k
$$

However, there are two problems with this step:

1. It may not decrease the objective.
2. The iterations are not sparse.

Mark Schmidt
Opening up the Black Box
The steepest descent direction for $\ell_1$-Regularization problems,

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

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

$$z_i = \begin{cases} 
F'_i(x) = f'_i(x) + \lambda \text{sign}(x_i), & |x_i| > 0 \\
0, & x_i = 0, |f'_i(x)| \leq \lambda \\
f'_i(x) - \lambda \text{sign}(f'_i(x)), & x_i = 0, |f'_i(x)| > \lambda 
\end{cases}$$
The steepest descent direction for $\ell_1$-Regularization problems,

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

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

$$z_i = \begin{cases} 
F'_i(x) = f'_i(x) + \lambda \text{sign}(x_i), & |x_i| > 0 \\
0, & x_i = 0, |f'_i(x)| \leq \lambda \\
 f'_i(x) - \lambda \text{sign}(f'_i(x)), & x_i = 0, |f'_i(x)| > \lambda 
\end{cases}$$

We can even try a Newton-like version:

$$x^{k+1} = x^k - \alpha [H_k]^{-1} z^k$$
The steepest descent direction for $\ell_1$-Regularization problems,

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

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

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

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.
Use orthant projection to get sparse iterates:

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

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

- Use **orthant projection** to get sparse iterates:

\[ x^{k+1} \leftarrow P_{O(x^k)}[x^k - \alpha[H_k]^{-1}z^k], \]

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

- Use orthant projection to get sparse iterates:

\[ x^{k+1} \leftarrow P_{O(x^k)}[x^k - \alpha[H_k]^{-1}z^k], \]

[Osborne et al., 2000, Andrew & Gao, 2007]
Use orthant projection to get sparse iterates:

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

[Osborne et al., 2000, Andrew & Gao, 2007]
Use orthant projection to get sparse iterates:

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

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

- Use orthant projection to get sparse iterates:

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

[Osborne et al., 2000, Andrew & Gao, 2007]
Use orthant projection to get sparse iterates:

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

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

- Use orthant projection to get sparse iterates:

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

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

- Variables that change sign become exactly zero.

Mark Schmidt
Opening up the Black Box
We can guarantee descent using diagonal scaling:

\[ x^{k+1} \leftarrow P_{O(x^k)}[x^k - \alpha[D_k]^{-1}z^k]. \]
We can guarantee descent using diagonal scaling:

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

Less restrictive: diagonal with respect to variables near zero:

\[ \mathcal{A} = \{ i \mid |x_i^k| \leq \epsilon \}, \quad \mathcal{F} = \{ i \mid |x_i^k| > \epsilon \} \]
Two-Metric Sub-Gradient Projection

- We can guarantee descent using diagonal scaling:

\[ x^{k+1} \leftarrow P_{O(x^k)}[x^k - \alpha[D_k]^{-1}z^k]. \]

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

\[ A = \{ i : |x_i^k| \leq \epsilon \}, \quad F = \{ i : |x_i^k| > \epsilon \} \]

- Two-metric sub-gradient projection:

\[ x^{k+1}_F \leftarrow P_{O(x^k_F)}[x^k_F - \alpha[H_k]^{-1}F_F(x^k)]. \]
\[ x^{k+1}_A \leftarrow P_{O(x^k_A)}[x^k_A - \alpha[D_k]^{-1}z^k_A], \]
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:
  \[ \mathcal{A} = \{ i \mid |x^k_i| \leq \epsilon \}, \quad \mathcal{F} = \{ i \mid |x^k_i| > \epsilon \} \]

- 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):

![Graph showing function evaluations and objective value minus optimal for different methods.]
Similar ideas used in many $\ell_1$-Regularization solvers.

Discussion

- Similar ideas used in many $\ell_1$-Regularization solvers.

- Recent methods consider two more issues:
  - **Sub-Optimization**: Identify variables likely to stay zero.
    [El Ghaoui et al., 2010].
  - **Continuation**: Start with a large $\lambda$ and slowly decrease it.
    [Xiao and Zhang, 2012]
Discussion

- Similar ideas used in many $\ell_1$-Regularization solvers.

- Recent methods consider two more issues:
  - **Sub-Optimization**: Identify variables likely to stay zero.
    [El Ghaoui et al., 2010].
  - **Continuation**: Start with a large $\lambda$ 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.

Mark Schmidt
Opening up the Black Box
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 $\ell_1$-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 coronary heart disease.
Motivation: Structure Learning in CRFs

- Task: early detection of coronary 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 coronary heart disease.
- Assess motion of 16 heart segments using CRF.
- But, do not know the best correlation structure.
- Perform structure learning with $\ell_1$-regularization.
We want to fit a Markov random field with unknown structure.
We want to fit a **Markov random field** with unknown structure.
We want to fit a Markov random field with unknown structure.
We want to fit a Markov random field with unknown structure.

Learn a sparse structure by $\ell_1$-regularization of edge weights.

[Lee et al. 2006, Wainwright et al. 2006]
In some cases, we want sparsity in groups of parameters:

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

1. Multi-class variables [Lee et al., 2006].
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].
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].
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].
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].
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]
Encourage group sparsity using group $\ell_1$-regularization:

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

where

$$\|x\|_{1,p} = \sum_g \|x_g\|_p.$$
Encourage group sparsity using group $\ell_1$-regularization:

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

where

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

This is $\ell_1$-regularization of group norms.

Typically $p = 2$, but other norms give other properties.
Structure Learning with Group $\ell_1$-Regularization

- Encourage group sparsity using group $\ell_1$-regularization:
  \[
  \min_x f(x) + \lambda \|x\|_{1,p},
  \]
  where
  \[
  \|x\|_{1,p} = \sum_g \|x_g\|_p.
  \]
- This is $\ell_1$-regularization of group norms.
- Typically $p = 2$, but other norms give other properties.
Effect of Different Group Norms

- Group $\ell_1$-Regularization with the $\ell_2$ group norm.
- Encourages group sparsity.
Effect of Different Group Norms

- Group $\ell_1$-Regularization with the $\ell_\infty$ group norm.
- Encourages group sparsity and parameter tieing.
Group $\ell_1$-Regularization with the nuclear group norm.

Encourages group sparsity and low-rank.
Optimization with Group $\ell_1$-Regularization

We’ll focus on the group $\ell_1$-regularized optimization:

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

where $f$ is the CRF (expensive) objective.
We’ll focus on the group $\ell_1$-regularized optimization:

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

where $f$ is the CRF (expensive) objective.

The regularizer is non-separable.
We’ll focus on the group $\ell_1$-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 $\ell_1$-Regularization

- We’ll focus on the group $\ell_1$-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?
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:

$$\min_{||x_g|| \leq t_g} F(x) = f(x) + \lambda \sum_g t_g.$$
We can re-write the non-smooth objective

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

as a smooth objective with norm-cone constraints:

$$\min_{\|x_g\| \leq t_g} F(x) = f(x) + \lambda \sum_g t_g.$$
We can re-write the non-smooth objective

\[
\min_x f(x) + \lambda \sum_g \|x\|,
\]

as a smooth objective with norm-cone constraints:

\[
\min_{\|x_g\| \leq t_g} F(x) = f(x) + \lambda \sum_g t_g.
\]
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:
  \[
  \min_{\|x\|_p \leq t} F(x) = f(x) + \lambda \sum g\ t_g.
  \]

- 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.
A general form of projected gradient:

\[ x^{k+1} \leftarrow \arg \min_{x \in C} \| x - (x^k - \alpha F'(x^k)) \| \]
A general form of projected gradient:

$$x^{k+1} \leftarrow \arg \min_{x \in C} ||x - (x^k - \alpha F'(x^k))||$$
A general form of projected gradient:

$$x^{k+1} \leftarrow \arg \min_{x \in C} ||x - (x^k - \alpha F'(x^k))||$$
A general form of projected gradient:

\[ x^{k+1} \leftarrow \arg \min_{x \in C} ||x - (x^k - \alpha F'(x^k))|| \]
A general form of projected gradient:

\[ x^{k+1} \leftarrow \arg \min_{x \in C} \| x - (x^k - \alpha F'(x^k)) \| \]
We can consider a Newton-like step:

\[ x^{k+1} \leftarrow \arg \min_{x \in \mathcal{C}} \| x - (x^k - \alpha \left[ H_k \right]^{-1} F'(x^k)) \|, \]

but as we saw this doesn’t work.

Projected Newton

- We can consider a Newton-like step:
  \[
  x^{k+1} \leftarrow \arg \min_{x \in \mathcal{C}} \| x - (x^k - \alpha [H_k]^{-1} F'(x^k)) \|,
  \]
  but as we saw this doesn’t work.

- **Projected Newton** methods project under the same norm:
  \[
  x^{k+1} \leftarrow \arg \min_{x \in \mathcal{C}} \| x - (x^k - \alpha [H_k]^{-1} F'(x_k)) \|_{H_k},
  \]
  where \( \| x \|_{H_k} = \sqrt{x^T H_k x} \).

[Levitin & Polyak, 1966]
Projected Newton

- We can consider a Newton-like step:

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

  but as we saw this doesn’t work.

- **Projected Newton** methods project under the same norm:

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

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

  [Levitin & Polyak, 1966]

- Convergence properties similar to Newton’s method.
Projected Newton methods equivalently minimize a constrained quadratic approximation:

\[
x^{k+1} \leftarrow \arg \min_{x \in C} F(x^k) + \langle F'(x^k), x - x^k \rangle + \frac{1}{2\alpha} \|x - x_k\|^2_{H_k}.
\]
Projected Newton methods equivalently minimize a constrained quadratic approximation:

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

This is expensive even with simple constraints.
Inexact Projected Newton

- Projected Newton methods equivalently minimize a constrained quadratic approximation:

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

- This is expensive even with simple constraints.

- Solution: use a cheap approximate solver.
Inexact Projected Newton

Feasible Set

$f(x)$

$x^k$

Feasible Set
Inexact Projected Newton

Feasible Set

\[ f(x) \]

\[ x^k \]

\[ Q(x, \alpha) \]

Feasible Set
Inexact Projected Newton

\[ f(x) \]

\[ x_k \]

\[ P \]

\[ C \]

\[ x_k - \alpha f(x_k) \]

\[ Q(x, \alpha) \]

\[ PC[x^k - \alpha \nabla f(x^k)] \]

Feasible Set
Inexact Projected Newton

Feasible Set

$f(x)$

$x_k$

$Q(x, \alpha)$

Feasible Set
Inexact Projected Newton

Feasible Set

\( f(x) \)

\( x^k \)

\( Q(x, \alpha) \)

Feasible Set
Inexact Projected Newton

\begin{equation}
\min_{x \in C} Q(x, \alpha)
\end{equation}

Feasible Set
Can we terminate this early?

For small enough $\alpha$, 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
Can we terminate this early?

- For small enough $\alpha$, 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
Can we terminate this early?
- For small enough $\alpha$, 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
Can we terminate this early?
- For small enough $\alpha$, 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 $\alpha$, 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](http://www.di.ens.fr/~mschmidt/Software/PQN.html)
Comparing PQN to first-order methods on a graphical model structure learning problem. [Gasch et al., 2000, Duchi et al., 2008].
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?
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.
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 $f$,

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

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

- We can equivalently write this as the quadratic optimization

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

- 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 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 \(f\),

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

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

- 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 \( f \),

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

- 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 proximal-gradient algorithm:

\[ x_{k+1} = \text{prox}_{\alpha g}[x_k - \alpha f'(x_k)]. \]
Projected-gradient methods are a special case:

\[ g(x) = \begin{cases} 
0 & \text{if } x \in C \\
\infty & \text{if } x \notin C.
\end{cases} \]
Projected-gradient methods are a special case:

$$g(x) = \begin{cases} 
0 & \text{if } x \in C \\
\infty & \text{if } x \notin C.
\end{cases}$$
Projected-gradient methods are a special case:

\[
g(x) = \begin{cases} 
0 & \text{if } x \in C \\
\infty & \text{if } x \notin C.
\end{cases}
\]
Projected-gradient methods are a special case:

\[
g(x) = \begin{cases} 
0 & \text{if } x \in C \\
\infty & \text{if } x \notin C.
\end{cases}
\]
Projected-gradient methods are a special case:

\[ g(x) = \begin{cases} 
0 & \text{if } x \in C \\
\infty & \text{if } x \notin C.
\end{cases} \]
Iterative Soft-Thresholding methods are a special case:

\[ g(x) = \lambda \|x\|_1. \]
Iterative Soft-Thresholding methods are a special case:

\[ g(x) = \lambda \|x\|_1. \]

In this case, proximal operator shrinks \(|x_i|\) by up to \(\lambda \alpha\).
Iterative Soft-Thresholding methods are a special case:

\[ g(x) = \lambda \|x\|_1. \]

In this case, proximal operator shrinks \(|x_i|\) by up to \(\lambda \alpha\).
Iterative Soft-Thresholding methods are a special case:

\[ g(x) = \lambda \| x \|_1. \]

In this case, proximal operator shrinks \( |x_i| \) by up to \( \lambda \alpha \).
Iterative Soft-Thresholding methods are a special case:

\[ g(x) = \lambda \|x\|_1. \]

In this case, proximal operator shrinks \(|x_i|\) by up to \(\lambda \alpha\).
Iterative Soft-Thresholding methods are a special case:

\[ g(x) = \lambda \|x\|_1. \]

In this case, proximal operator shrinks \(|x_i|\) by up to \(\lambda \alpha\).
The group $\ell_1$-regularizer is simple; we can compute the proximal operator in linear time. [Wright et al., 2009]

\[
\text{prox}_{\alpha\|x_g\|}[x_g] = \arg \min_x \frac{1}{2}\|x - x_g\|^2 + \alpha\|x\|
\]

\[
= \frac{x_g}{\|x_g\|} \max\{0, \|x_g\| - \alpha\}
\]
The basic proximal-gradient step:

$$x^{k+1} \leftarrow \arg\min_x \frac{1}{2} \| x - (x^k - \alpha f'(x^k)) \|^2 + \alpha g(x)$$
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.
The basic proximal-gradient step:

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

Same convergence rate as gradient method.

To speed the convergence, we might consider Newton-like step:

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

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

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

- 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 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.
  - 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
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...
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...
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...
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...
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...
The pairwise assumption is inherent to Gaussian models.
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 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.
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 $|A| > 2$. 
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 $|A| > 2$. 
We can extend group $\ell_1$-regularization to the general case:

$$\min_w f(w) + \sum_{A \subseteq S} \lambda_A \|w_A\|.$$
We can extend group $\ell_1$-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.
We can extend group $\ell_1$-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., three-way models).

[Dahinden et al., 2007]
Hierarchical Log-Linear Models

- Instead of restricting cardinality, we use **hierarchical inclusion**:

\[ (1, 2, 3) \text{ if } (1, 2), (1, 3), \text{ and } (2, 3). \]

In general: If \( w_A = 0 \) then supersets \( B \) of \( A \) must have \( w_B = 0 \).

The class of hierarchical log-linear models: 
- 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?
Instead of restricting cardinality, we use hierarchical inclusion:

- We can only have \((1, 2, 3)\) if we also have \((1, 2)\), \((1, 3)\), and \((2, 3)\).
Instead of restricting cardinality, we use \textit{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 \(w_B = 0\).
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.
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?
Can enforce a hierarchy with **overlapping group $\ell_1$-regularization**.

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

\[
\text{Example: If we want } A = 0 \text{ to mean } B = 0, \text{ use two groups } \{B\} \text{ and } \{A, B\}, \lambda_{\{B\}} \|w_B\|_2 + \lambda_{\{A, B\}} \|w_{A, B}\|_2.
\]

To make $w_A$ non-zero, pay $\lambda_{\{A, B\}}$. To make $w_B$ non-zero, pay $\lambda_B$ (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

\[
\min_w f(w) + \sum_{A \subseteq S} \lambda_A \|w_A^*\|_2,
\]

where $A^* = \{B | A \subseteq B\}$.

[Bach, 2008, Zhao et al., 2009]
Can enforce a hierarchy with overlapping group $\ell_1$-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 $\lambda_B$ (but also $\lambda_{\{A, B\}}$ if $w_A = 0$).
- If $w_B \neq 0$, no penalty for making $w_A$ non-zero.
Can enforce a hierarchy with overlapping group $\ell_1$-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 $\lambda_B$ (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]
But can we avoid looking at all higher-order potentials?
But can we avoid looking at all higher-order potentials?

Heuristic: only consider adding groups that satisfy hierarchy.
(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 hierarchy. (And that are sub-optimal. E.g., poorly estimated by the model.)
- Guarantees weak form of global optimality.
Example of Active Set Method

Initial boundary groups.

1,2,3,4,5

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
5
Optimize 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

Find new **active groups**.

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
1
2
3
4
5

Mark Schmidt
Opening up the Black Box
Example of Active Set Method

Find new boundary groups.

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
1
2
3
4
5

Mark Schmidt
Opening up the Black Box
Example of Active Set Method

Optimize active groups and sub-optimal boundary groups.

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 **active groups**.
Example of Active Set Method

Find new boundary groups.
Example of Active Set Method

Optimize active groups and sub-optimal boundary groups.

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
1
2
3
4
5
Example of Active Set Method

Find new active groups.

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

Mark Schmidt
Opening up the Black Box
Example of Active Set Method

Find new boundary groups.

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

Mark Schmidt
Opening up the Black Box
Example of Active Set Method

Optimize active groups and sub-optimal boundary groups.
Example of Active Set Method

Find new active groups.

1,2,3,4,5
Example of Active Set Method

Find new boundary groups.

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

1,2,3,4,5
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.
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 three-way interactions.
  - 1 of 5 possible four-way interactions.
  - No five-way interactions.
- The heuristic can reduce the space exponentially.
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

Mark Schmidt
Opening up the Black Box
Traffic Flow Data

Mark Schmidt
Opening up the Black Box
We now turn to the overlapping group $\ell_1$-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:

\[
\min_{\|x_g\| \leq t_g} \quad f(x) + \lambda \sum_{g} t_g.
\]
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.
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.
Projecting onto the intersection of simple sets is a classic problem:
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]

For polyhedral sets, Dykstra’s algorithm has a linear convergence rate. [Deutsch and Hundal, 1994]
Definition 13.7: If $\mathcal{O}_1, \mathcal{O}_2, \ldots$ is a sequence of s.v. operators, if $f$ is an element of $\bigcap_{n=1}^{\infty} \text{dom}(\mathcal{O}_n)$ such that $\lim_{n \to \infty} \mathcal{O}_n f$ exists, and if $D$ is the set of all such elements $f$, then $\sum_n$ is said to have a limit $\mathcal{O}$ over $D$, and, for $f \in D$, $\sum_n \mathcal{O} f(n) = D(\mathcal{O}), \mathcal{O} f = \lim_{n \to \infty} \mathcal{O} f(n)$.

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 $(A_m f, A_n f) = (A_{m+n-\xi} f, g)$, where $\xi = 1$ if $m$ and $n$ have the same parity and $\xi = 0$ 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 $\|A_m f - A_n f\|_n^2 = (A_m f - A_n f, A_m f - A_n f) = \ldots$
von Neumann's Result

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.
Project onto subspace 1.
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.
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]
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 intersections. [Bregman, 1965]
We have an arbitrary number of convex sets.
Start with some initial point.
Project onto convex set 1.
Project onto convex set 2.
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 intersections. [Bregman, 1965]
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 intersections. [Bregman, 1965]
- A simple modification makes the method converge to the projection onto their intersections. [Dykstra, 1983]
We want to project a point onto the intersection of convex sets.
Dykstra’s Algorithm

Project onto convex set 1, and store the difference.
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.
Remove the difference from projecting on convex set 2.
Project onto convex set 2, and store the difference.
Dykstra’s Algorithm

The limit is the projection onto the intersection.
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 intersections. [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]
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 intersections. [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]
We can efficiently compute the proximity operator for:

1. $\ell_1$-Regularization.
2. Group $\ell_1$-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 $\ell_1$-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.
5. Combinations of simple functions.
We can efficiently compute the proximity operator for:

1. $\ell_1$-Regularization.
2. Group $\ell_1$-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 $\ell_1$-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.
5. Combinations of simple functions.
We can efficiently compute the proximity operator for:

1. $\ell_1$-Regularization.
2. Group $\ell_1$-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 $\ell_1$-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.
5. Combinations of simple functions.
Can inexact proximal-gradient methods achieve the fast rates?
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)
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 \( \{\sqrt{\epsilon_k}\} \) 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
Large-scale machine learning: large $N$, large $P$

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

- $N$: number of observations (inputs)
- $P$: dimension of each observation

Regularized empirical risk minimization:

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

- data fitting term + regularizer

Applications to any data-oriented field:

- Vision, bioinformatics, speech, natural language, web.
**Large-scale machine learning**: large $N$, large $P$

- $N$: number of observations (inputs)
- $P$: dimension of each observation

**Regularized empirical risk minimization**:

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

- data fitting term + 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.
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 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.
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 $\ell_2$-regularized least-squares,

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

Other examples include any $\ell_2$-regularized convex loss:

- logistic regression, Huber regression, smooth SVMs, CRFs, etc.
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 \textit{linear in} \( N \).
  - Quasi-Newton methods still require \( O(N) \).
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]:

  \[
  \theta_t = \theta_t - \frac{1}{n} \sum_{i=1}^{n} f'_i(\theta_t - 1)
  \]

- **Stochastic** gradient descent:

  \[
  \theta_t = \theta_t - \frac{1}{n} \sum_{i=1}^{n} f'_i(\theta_t - 1)
  \]

Stochastic vs. deterministic methods

**Deterministic** gradient method [Cauchy, 1847]:

\[
\theta_t = \theta_t - \frac{1}{n} \sum_{i=1}^{n} f'_i(\theta_t - 1)
\]

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

\[
\theta_t = \theta_t - \frac{1}{n} \sum_{i=1}^{n} f'_i(\theta_t - 1)
\]
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.

Stochastic vs. deterministic methods

- **Goal** = best of both worlds: linear rate with $O(1)$ iteration cost
time
log(excess cost)
stochastic
deterministic
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.

Stochastic vs. deterministic methods

- **Goal** = best of both worlds: linear rate with $O(1)$ iteration cost
- **Hybrid** log(excess cost)
- **Stochastic**
- **Deterministic**
- **Time**

Goal is $O(1)$ cost with $O(\rho^k)$ rate.
A variety of methods have been proposed to speed up SG methods:

- **Momentum, gradient/iterate averaging**

- **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**

- **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
Existing linear convergence results:

- **Constant step-size SG, accelerated SG**
  - Linear convergence but only up to a fixed tolerance

- **Hybrid methods, incremental average gradient**
  - Linear rate but iterations make full passes through the data
Existing linear convergence results:

- **Constant step-size SG, accelerated SG**
  - Linear convergence but only up to a fixed tolerance

- **Hybrid methods, incremental average gradient**
  - 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
Is it possible to have a general linearly convergent algorithm with iteration cost independent of $N$?
Is it possible to have a general linearly convergent algorithm with iteration cost independent of $N$?

YES!
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 $f'_{i(t)}(x^t)$,

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

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

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

- Memory: $x^t_i$ is the last iterate where $i$ was selected.
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 $f'_{i(t)}(x^t)$,

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

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

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

**Memory:** $x^t_i$ 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.
Assume only that:

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

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

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

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

This rate is "very fast":
- Well-conditioned problems: constant non-trivial reduction per pass: $(1 - \frac{1}{8}N)N \leq \exp(-\frac{1}{8}) = 0.8825$.
- Badly-conditioned problems, almost same as deterministic method. (Deterministic has rate $(1 - \mu L)^2t$ with $\alpha = \frac{1}{L}$, but $N$ times slower.)
Convergence Rate of SAG

Assume only that:

- $f_i$ is convex, $f_i'$ is $L$–continuous, $f$ is $\mu$-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 $\alpha \leq \frac{1}{16L}$).
Assume only that:

- $f_i$ is convex, $f_i'$ is $L$–continuous, $f$ is $\mu$-strongly convex.

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

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

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

- Gradient method has rate $(L - \mu L + \mu)^2 = 0.99998$.
- Accelerated gradient method has rate $(1 - \sqrt{\mu L}) = 0.99761$.
- SAG (N iterations) has rate $(1 - \min\{\mu 16, 1/8 N\}) = 0.88250$.
- Fastest possible deterministic method: $(\sqrt{L} - \sqrt{\mu \sqrt{L}} + \sqrt{\mu})^2 = 0.99048$.

SAG beats two lower bounds: Stochastic gradient bound (of $O(1/t)$). Deterministic gradient bound (for typical $L$, $\mu$, and $N$).
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
  \[ \left( 1 - \sqrt{\mu L} \right) = 0.99761. \]

- **SAG** (N iterations) has rate
  \[ \left( 1 - \min \{ \mu 16L, \frac{1}{8N} \} \right)^N = 0.88250. \]

- **Fastest possible deterministic method**:
  \[ \left( \sqrt{L} - \sqrt{\mu \sqrt{L} + \mu} \right)^2 = 0.99048. \]

**SAG** beats two lower bounds:

- Stochastic gradient bound (of $O(1/t)$).
- Deterministic gradient bound (for typical $L$, $\mu$, and $N$).
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 $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$. 
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** $(1 - \sqrt{\frac{\mu}{L}}) = 0.99761$.
- **SAG (N iterations) has rate** $(1 - \min \left\{ \frac{\mu}{16L}, \frac{1}{8N} \right\})^N = 0.88250$. 

**Fastest possible deterministic method:** $(\sqrt{L} - \sqrt{\mu} \sqrt{L} + \sqrt{\mu})^2 = 0.99048$.

SAG beats two lower bounds: 

- Stochastic gradient bound (of $O(1/t)$).
- Deterministic gradient bound (for typical $L$, $\mu$, and $N$).
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** $(1 - \sqrt{\frac{\mu}{L}}) = 0.99761$.
- **SAG (N iterations) has rate** $(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\})^N = 0.88250$.
- **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 $O\left(\frac{1}{t}\right)$).
- Deterministic gradient bound (for typical $L$, $\mu$, and $N$).
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 $L$, $\mu$, and $N$).
Convergence Rate in Convex Case

Assume only that:

- $f_i$ is convex, $f'_i$ is $L$–continuous, some $x^*$ exists.
Assume only that:

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

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

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

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

Assume only that:

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

**Theorem.** With $\alpha_t \leq \frac{1}{16L}$ 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 $\mu$ is locally bigger around $x^*$.  

Mark Schmidt

Opening up the Black Box
Comparing FG and SG Methods

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

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!
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!

Mark Schmidt
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]
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!