Tractable Big Data and Big Models in Machine Learning

Mark Schmidt

University of British Columbia
TAAI 2014

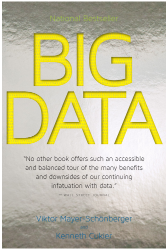
November 2014

Context: Big Data and Big Models

- We are collecting data at unprecedented rates.
  - Seen across many fields of science and engineering.
  - Not gigabytes, but terabytes or petabytes (and beyond).

Context: Big Data and Big Models

- We are collecting data at unprecedented rates.
  - Seen across many fields of science and engineering.
  - Not gigabytes, but terabytes or petabytes (and beyond).



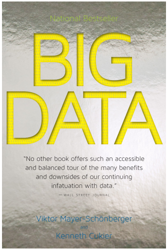
- Many important aspects to the 'big data' puzzle:
  - Distributed data storage and management, parallel computation, software paradigms, data mining, machine learning, privacy and security issues, reacting to other agents, power management, summarization and visualization.

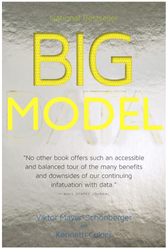
Context: Big Data and Big Models

- Machine learning uses big data to fit richer statistical models:
  - Vision, bioinformatics, speech, natural language, web, social.
  - Developping broadly applicable tools.
  - Output of models can be used for further analysis.

Context: Big Data and Big Models

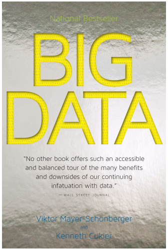
- Machine learning uses big data to fit richer statistical models:
  - Vision, bioinformatics, speech, natural language, web, social.
  - Developping broadly applicable tools.
  - Output of models can be used for further analysis.

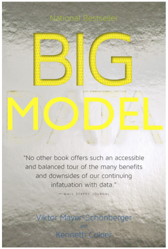




Context: Big Data and Big Models

- Machine learning uses big data to fit richer statistical models:
  - Vision, bioinformatics, speech, natural language, web, social.
  - Developping broadly applicable tools.
  - Output of models can be used for further analysis.

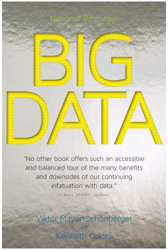


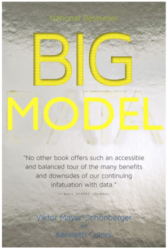


- Numerical optimization is at the core of many of these models.

Context: Big Data and Big Models

- Machine learning uses big data to fit richer statistical models:
  - Vision, bioinformatics, speech, natural language, web, social.
  - Developping broadly applicable tools.
  - Output of models can be used for further analysis.





- Numerical optimization is at the core of many of these models.
- But, traditional 'black-box' methods have difficulty with:
  - the large data sizes.
  - the large model complexities.

Two Issues in this Talk

- The first issue is computation:
  - We 'open up the black box', by using the structure of machine models to derive faster large-scale optimization algorithms.
  - Can lead to enormous speedups for big data and complex models.

(polynomial vs. exponential)

Two Issues in this Talk

- The first issue is computation:
  - We ‘open up the black box’, by using the structure of machine models to derive faster large-scale optimization algorithms.
  - Can lead to enormous speedups for big data and complex models.

(polynomial vs. exponential)

- The second issue is modeling:
  - By expanding the set of tractable problems, we can propose richer classes of statistical models that can be efficiently fit.

Two Issues in this Talk

- The first issue is computation:
  - We ‘open up the black box’, by using the structure of machine models to derive faster large-scale optimization algorithms.
  - Can lead to enormous speedups for big data and complex models.

    (polynomial vs. exponential)

- The second issue is modeling:
  - By expanding the set of tractable problems, we can propose richer classes of statistical models that can be efficiently fit.

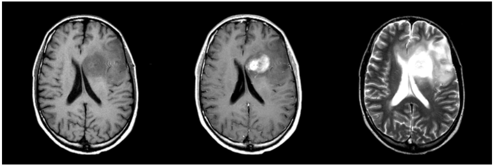
- My research tries to alternate between these two.

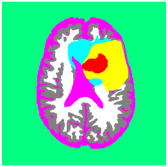
Outline

1. Structured sparsity (inexact proximal-gradient method)
2. Learning dependencies (costly models with simple constraints)
3. Fitting a huge dataset (stochastic average gradient)

Motivation: Automatic Brain Tumor Segmentation

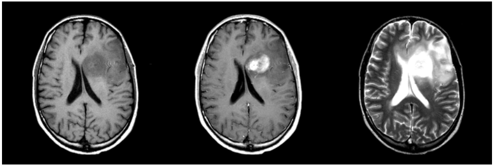
- Task: Segmentation of Multi-Modality MRI Data

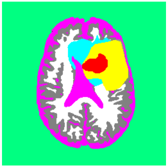




Motivation: Automatic Brain Tumor Segmentation

- Task: Segmentation of Multi-Modality MRI Data

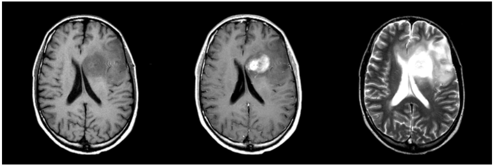


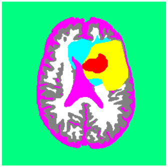


- Applications:
  - image-guided surgery
  - radiation target planning.
  - quantifying treatment response.
  - mining growth patterns.

Motivation: Automatic Brain Tumor Segmentation

- Task: Segmentation of Multi-Modality MRI Data

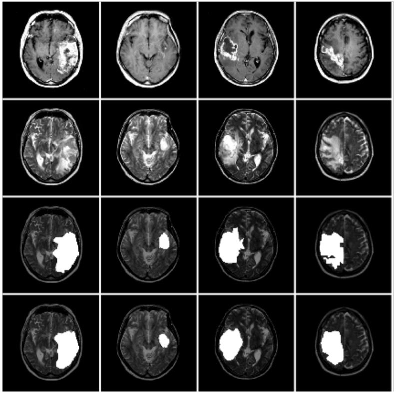




- Applications:
  - image-guided surgery
  - radiation target planning.
  - quantifying treatment response.
  - mining growth patterns.
- Challenges:
  - variety of tumor appearances.
  - similarity to normal tissue.

Motivation: Automatic Brain Tumor Segmentation

- Solution strategy:
  1. Incorporate prior knowledge by registration with template.
  2. Pixel-level classifier using image- and template-based features.



Motivation: Automatic Brain Tumor Segmentation

- Best performance with logistic regression:

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

Motivation: Automatic Brain Tumor Segmentation

- Best performance with logistic regression:

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

- Problem 1: Estimating x is slow:
  - 8 million voxels per volume.
  - Later in this talk: Big-N problems.

Motivation: Automatic Brain Tumor Segmentation

- Best performance with logistic regression:

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

- Problem 1: Estimating x is slow:
  - 8 million voxels per volume.
  - Later in this talk: Big-N problems.
- Problem 2: Designing features.
  - Lots of possible candidate features.
  - Using all features leads to over-fitting.
- Due to slow training time: manual feature selection.

Adding Regularization

- Strange idea: try **all features** with L2-Regularization:

$$\min_{x \in \mathbb{R}^P} \frac{1}{N} \sum_{i=1}^{N} f_i(x) + \lambda \sum_{i=1}^{P} x_i^2.$$

Adding Regularization

- Strange idea: try **all features** with L2-Regularization:

$$\min_{x \in \mathbb{R}^P} \frac{1}{N} \sum_{i=1}^{N} f_i(x) + \lambda \sum_{i=1}^{P} x_i^2.$$

  - Reduces over-fitting.
  - As good as best selected features.
  - Smooth function, so we can compute this on large datasets:

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

Adding Regularization

- Strange idea: try **all features** with L2-Regularization:

$$\min_{x \in \mathbb{R}^P} \frac{1}{N} \sum_{i=1}^{N} f_i(x) + \lambda \sum_{i=1}^{P} x_i^2.$$

  - Reduces over-fitting.
  - As good as best selected features.
  - Smooth function, so we can compute this on large datasets:

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

  - But, uses all features so slow to segment new images.

Adding Regularization

- Strange idea: try **all features** with L2-Regularization:

$$\min_{x \in \mathbb{R}^P} \frac{1}{N} \sum_{i=1}^{N} f_i(x) + \lambda \sum_{i=1}^{P} x_i^2.$$

  - Reduces over-fitting.
  - As good as best selected features.
  - Smooth function, so we can compute this on large datasets:

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

  - But, uses all features so slow to segment new images.

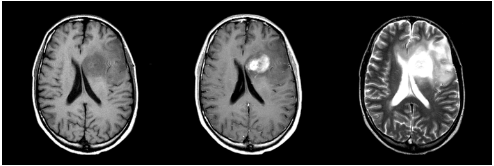
- Another strange idea: try **all features** with L1-Regularization:

$$\min_{x} \frac{1}{N} \sum_{i=1}^{N} f_i(x) + \lambda \sum_{i=1}^{P} |x_i|.$$

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

Feature Selection with L1-Regularization (Binary)

- Binary case:
  - Setting variable $x_j = 0$ removes the feature a_j.

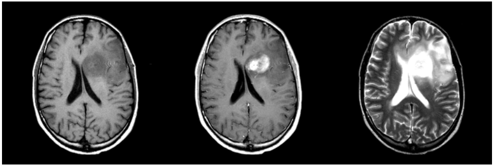


- Because we classify using the sign of $x^T a$:

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = x^T a$$

Feature Selection with L1-Regularization (Binary)

- Binary case:
  - Setting variable $x_j = 0$ removes the feature a_j.

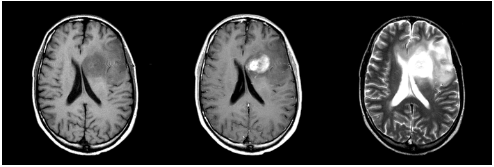


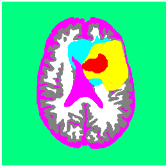
- Because we classify using the sign of $x^T a$:

$$\begin{bmatrix} 0 & x_2 & 0 & x_4 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = x^T a$$

Variable Selection with L1-Regularization

- *C*-class case:
  - Setting variable $x_j = 0$ may **not** remove the feature a_j.



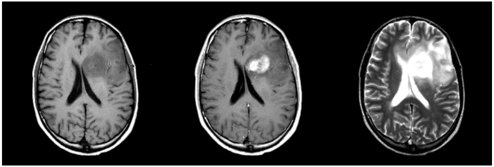


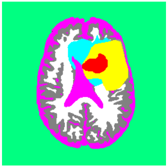
- Because we classify using the maximum of $x_c^T a$:

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} x_1^T a \\ x_2^T a \\ x_3^T a \\ x_3^T a \end{bmatrix}$$

Variable Selection with L1-Regularization

- C-class case:
  - Setting variable $x_j = 0$ may **not** remove the feature a_j.



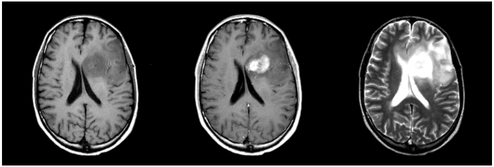


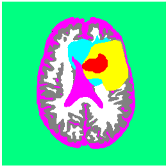
- Because we classify using the maximum of $x_c^T a$:

$$\begin{bmatrix} 0 & x_{12} & 0 & x_{14} & 0 \\ 0 & x_{22} & x_{23} & x_{24} & 0 \\ x_{31} & x_{32} & 0 & x_{34} & 0 \\ 0 & 0 & 0 & x_{44} & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} x_1^T a \\ x_2^T a \\ x_3^T a \\ x_3^T a \end{bmatrix}$$

Feature Selection with Group L1-Regularization

- *C*-class case:
  - Setting group $\{x_{1j}, x_{2j}, x_{3j}, x_{4j}, x_{5j}\} = 0$ removes the feature a_j.





- Because we classify using the maximum of $x_c^T a$:

$$\begin{bmatrix} 0 & x_{12} & 0 & x_{14} & 0 \\ 0 & x_{22} & 0 & x_{24} & 0 \\ 0 & x_{32} & 0 & x_{34} & 0 \\ 0 & x_{42} & 0 & x_{44} & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} x_1^T a \\ x_2^T a \\ x_3^T a \\ x_3^T a \end{bmatrix}$$

Group L1-Regularization

- L1-Regularization encourages sparsity in variables x_i.

$$\min_x \frac{1}{N}\sum_{i=1}^N f_i(x) + \lambda \sum_{i=1}^P |x_i|.$$

Group L1-Regularization

- L1-Regularization encourages sparsity in variables x_i.

$$\min_x \frac{1}{N}\sum_{i=1}^{N} f_i(x) + \lambda \sum_{i=1}^{P} |x_i|.$$

- Group L1-regularization encourages sparsity in groups x_g:

$$\min_x \frac{1}{N}\sum_{i=1}^{N} f_i(x) + \lambda \sum_{g \in \mathcal{G}} \|x_g\|.$$

Group L1-Regularization

- L1-Regularization encourages sparsity in variables x_i.

$$\min_x \frac{1}{N}\sum_{i=1}^N f_i(x) + \lambda \sum_{i=1}^P |x_i|.$$

- Group L1-regularization encourages sparsity in groups x_g:

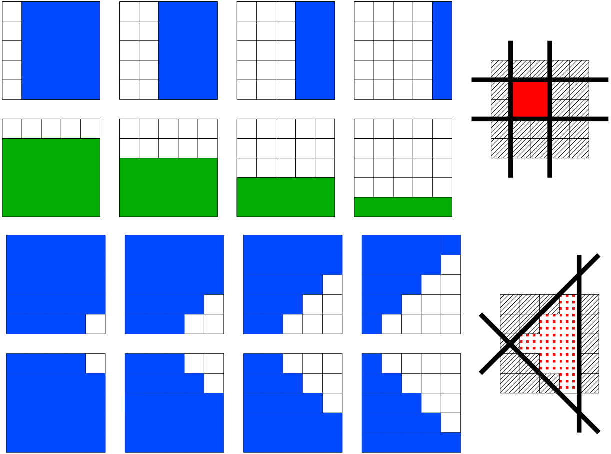
$$\min_x \frac{1}{N}\sum_{i=1}^N f_i(x) + \lambda \sum_{g\in\mathcal{G}} \|x_g\|.$$

- Structured sparsity generalizes groups to other structures.

Structured Sparsity Examples

- Examples of structured sparsity:

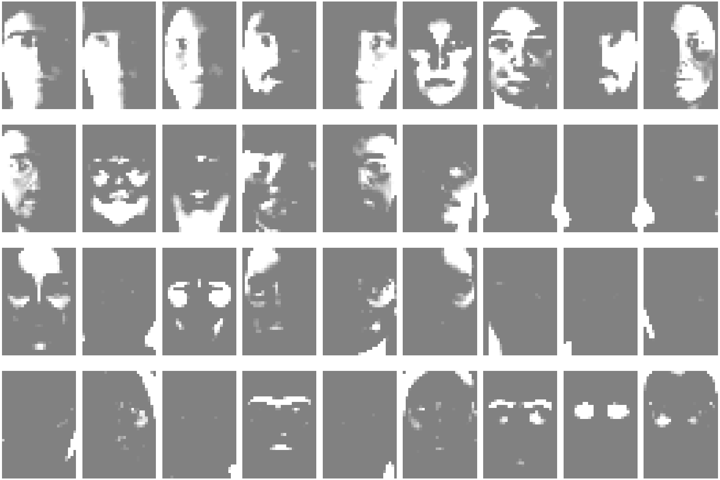
Structured sparsity to select convex regions:



Structured Sparsity Examples

- Examples of structured sparsity:

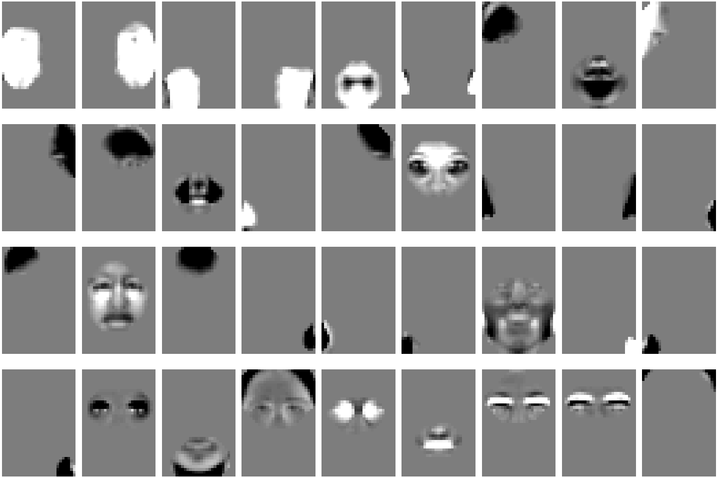
Dictionary learned with non-negative matrix factorization:



Structured Sparsity Examples

- Examples of structured sparsity:

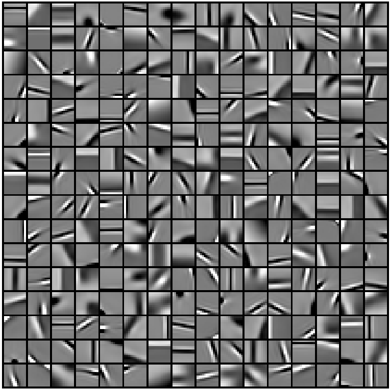
Dictionary learned with structured sparsity:

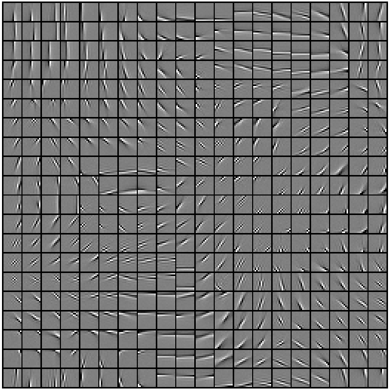


Structured Sparsity Examples

- Examples of structured sparsity:

Spatially-structured dictionary with structured sparsity:

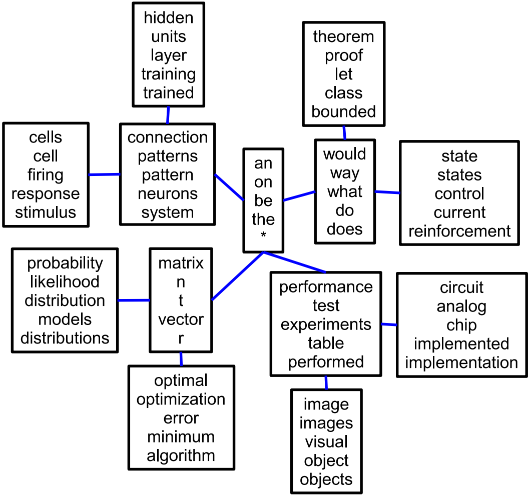




Structured Sparsity Examples

- Examples of structured sparsity:

Tree-structured dictionary with structured sparsity:



Structured Sparsity Examples

- Examples of structured sparsity:
  - A linear model with variable interactions:

$$m(x) = x_1 + x_2 + x_3 + x_{12} + x_{13} + x_{23} + x_{123}.$$

  - E.g., Mutations in both gene *A* and gene *B* lead to cancer.

Structured Sparsity Examples

- Examples of structured sparsity:
  - A linear model with variable interactions:

$$m(x) = x_1 + x_2 + x_3 + x_{12} + x_{13} + x_{23} + x_{123}.$$

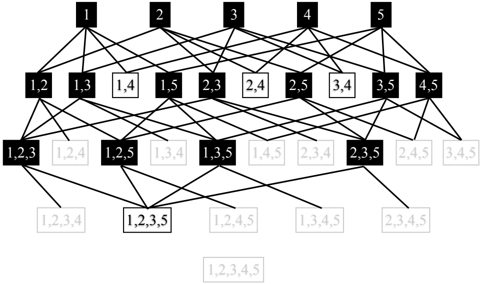
  - E.g., Mutations in both gene A and gene B lead to cancer.
  - We can't consider all 2^P possible interations.

Structured Sparsity Examples

- Examples of structured sparsity:
  - A linear model with variable interactions:

$$m(x) = x_1 + x_2 + x_3 + x_{12} + x_{13} + x_{23} + x_{123}.$$

  - E.g., Mutations in both gene A and gene B lead to cancer.
  - We can't consider all 2^P possible interations.
  - Structured sparsity on the hierarchical models.



Where does the sparsity come from?

- Unfortunately, all these regularizers are non-smooth.

Where does the sparsity come from?

- Unfortunately, all these regularizers are non-smooth.
- Consider our problem

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

data fitting term + regularizer

Where does the sparsity come from?

- Unfortunately, all these regularizers are non-smooth.
- Consider our problem

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

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

- A solution must have:
  - -gradient(data-fitting term) = subgradient(regularizer).
- Non-smoothness at zero ‘catches’ many solution:

Where does the sparsity come from?

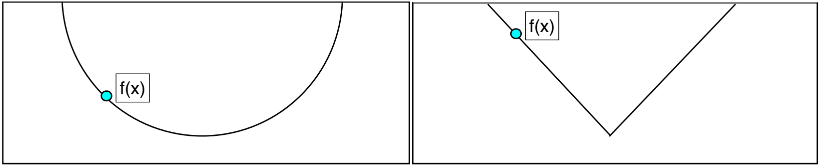
- Unfortunately, all these regularizers are non-smooth.
- Consider our problem

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

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

- A solution must have:
  - -gradient(data-fitting term) = subgradient(regularizer).
- Non-smoothness at zero 'catches' many solution:

L2-regularization L1-regularization



Where does the sparsity come from?

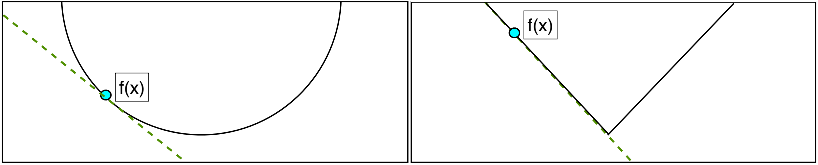
- Unfortunately, all these regularizers are non-smooth.
- Consider our problem

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

data fitting term + regularizer

- A solution must have:
  - -gradient(data-fitting term) = subgradient(regularizer).
- Non-smoothness at zero 'catches' many solution:

L2-regularization L1-regularization



Where does the sparsity come from?

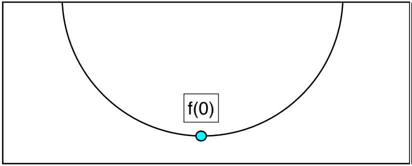
- Unfortunately, all these regularizers are non-smooth.
- Consider our problem

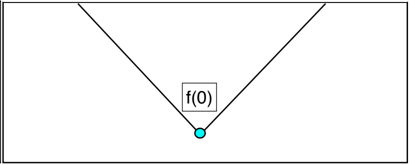
$$\min_{x \in \mathbb{R}^P} \quad \frac{1}{N}\sum_{i=1}^{N} f_i(x) \quad + \quad r(x)$$

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

- A solution must have:
  - -gradient(data-fitting term) = subgradient(regularizer).
- Non-smoothness at zero 'catches' many solution:

L2-regularization | L1-regularization





Where does the sparsity come from?

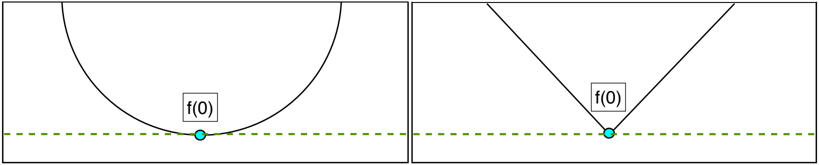
- Unfortunately, all these regularizers are non-smooth.
- Consider our problem

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

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

- A solution must have:
  - -gradient(data-fitting term) = subgradient(regularizer).
- Non-smoothness at zero 'catches' many solution:

L2-regularization L1-regularization



Where does the sparsity come from?

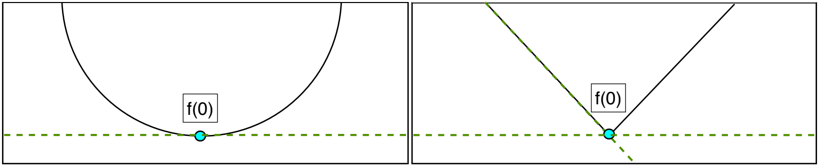
- Unfortunately, all these regularizers are non-smooth.
- Consider our problem

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

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

- A solution must have:
  - -gradient(data-fitting term) = subgradient(regularizer).
- Non-smoothness at zero 'catches' many solution:

L2-regularization L1-regularization



Where does the sparsity come from?

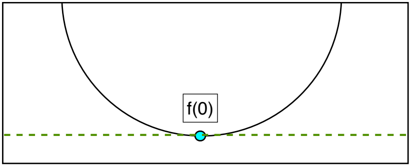
- Unfortunately, all these regularizers are non-smooth.
- Consider our problem

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

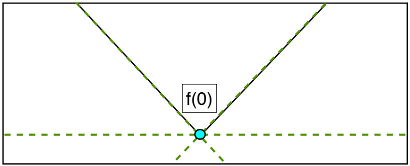
data fitting term $+$ regularizer

- A solution must have:
  - -gradient(data-fitting term) = subgradient(regularizer).
- Non-smoothness at zero 'catches' many solution:

L2-regularization



L1-regularization



Where does the sparsity come from?

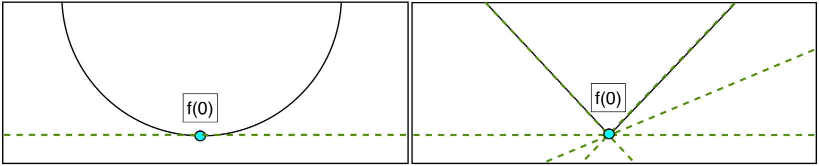
- Unfortunately, all these regularizers are non-smooth.
- Consider our problem

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

data fitting term $+$ regularizer

- A solution must have:
  - -gradient(data-fitting term) = subgradient(regularizer).
- Non-smoothness at zero 'catches' many solution:

L2-regularization L1-regularization



Where does the sparsity come from?

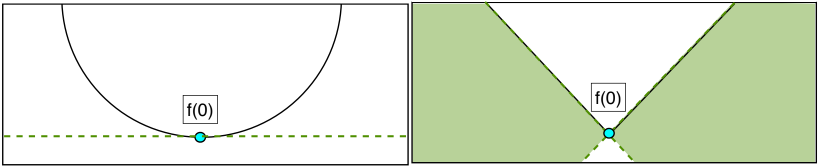
- Unfortunately, all these regularizers are non-smooth.
- Consider our problem

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

data fitting term $+$ regularizer

- A solution must have:
  - -gradient(data-fitting term) = subgradient(regularizer).
- Non-smoothness at zero 'catches' many solution:

L2-regularization L1-regularization



Can we solve huge-dimensional non-smooth optimization problems?

Black-Box Smooth and Non-Smooth Optimization

Can we solve huge-dimensional non-smooth optimization problems?

- Black-box model of large-scale optimization:
  - Algorithm can use $O(P)$ time to choose an iterate x^t.
  - Algorithm receives the function and subgradient at x^t.

Black-Box Smooth and Non-Smooth Optimization

Can we solve huge-dimensional non-smooth optimization problems?

- Black-box model of large-scale optimization:
  - Algorithm can use $O(P)$ time to choose an iterate x^t.
  - Algorithm receives the function and subgradient at x^t.
- How many iterations does it take to reach an accuracy of ϵ?

Black-Box Smooth and Non-Smooth Optimization

Can we solve huge-dimensional non-smooth optimization problems?

- Black-box model of large-scale optimization:
  - Algorithm can use $O(P)$ time to choose an iterate x^t.
  - Algorithm receives the function and subgradient at x^t.
- How many iterations does it take to reach an accuracy of ϵ?
- With standard subgradient-continuity and curvature assumptions:
  - Smooth problems can be solved in $O(\log(1/\epsilon))$ iterations.

(polynomial-time)

Black-Box Smooth and Non-Smooth Optimization

Can we solve huge-dimensional non-smooth optimization problems?

- Black-box model of large-scale optimization:
  - Algorithm can use $O(P)$ time to choose an iterate x^t.
  - Algorithm receives the function and subgradient at x^t.

- How many iterations does it take to reach an accuracy of ϵ?

- With standard subgradient-continuity and curvature assumptions:
  - Smooth problems can be solved in $O(\log(1/\epsilon))$ iterations.

    (polynomial-time)
  - Non-smooth problems can be solved in $O(1/\epsilon)$ iterations.

    (exponential-time)

Opening Up the Black Box

- Bad news:
  - Any non-smooth method requires $\Omega(1/\epsilon)$ in the worst case.
  - Huge difference in practice between non-smooth and smooth.

Opening Up the Black Box

- Bad news:
  - Any non-smooth method requires $\Omega(1/\epsilon)$ in the worst case.
  - Huge difference in practice between non-smooth and smooth.
- Is large-scale L1-regularization not feasible?

Opening Up the Black Box

- Bad news:
  - Any non-smooth method requires $\Omega(1/\epsilon)$ in the worst case.
  - Huge difference in practice between non-smooth and smooth.
- Is large-scale L1-regularization not feasible?
  - No, we don't have a general non-smooth black-box:

$$\min_{x \in \mathbb{R}^P} \quad \underbrace{\frac{1}{N}\sum_{i=1}^{N} f(x)}_{\text{smooth}} \quad + \quad \underbrace{r(x)}_{\text{'simple'}}$$

Opening Up the Black Box

- Bad news:
  - Any non-smooth method requires $\Omega(1/\epsilon)$ in the worst case.
  - Huge difference in practice between non-smooth and smooth.
- Is large-scale L1-regularization not feasible?
  - No, we don't have a general non-smooth black-box:

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

$$\text{smooth} \quad + \quad \text{`simple'}$$

- Proximal-gradient methods solve these problems in $O(\log(1/\epsilon))$.

Converge Rate of Gradient Method

- To minimize a smooth objective

$$\min_{x \in \mathbb{R}^P} f(x),$$

the gradient method minimizes the approximation

$$x^{t+1} = \arg\min_{x \in \mathbb{R}^P} f(x^t) + f'(x^t)^T(x - x^t) + \frac{1}{2\alpha}\|x - x^t\|^2.$$

Converge Rate of Gradient Method

- To minimize a smooth objective

$$\min_{x \in \mathbb{R}^P} f(x),$$

the gradient method minimizes the approximation

$$x^{t+1} = \arg\min_{x \in \mathbb{R}^P} f(x^t) + f'(x^t)^T(x - x^t) + \frac{1}{2\alpha}\|x - x^t\|^2.$$

yielding the iteration

$$x^{t+1} = x^t - \alpha f'(x^t),$$

and requiring $O(\kappa \log(1/\epsilon))$ iterations.

Converge Rate of Gradient Method

- To minimize a smooth objective

$$\min_{x \in \mathbb{R}^P} f(x),$$

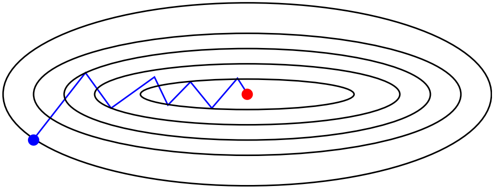
the gradient method minimizes the approximation

$$x^{t+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^P} f(x^t) + f'(x^t)^T (x - x^t) + \frac{1}{2\alpha}\|x - x^t\|^2.$$

yielding the iteration

$$x^{t+1} = x^t - \alpha f'(x^t),$$

and requiring $O(\kappa \log(1/\epsilon))$ iterations.



Converge Rate of Gradient Method

- To minimize a smooth objective

$$\min_{x \in \mathbb{R}^P} f(x),$$

the gradient method minimizes the approximation

$$x^{t+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^P} f(x^t) + f'(x^t)^T(x - x^t) + \frac{1}{2\alpha}\|x - x^t\|^2.$$

yielding the iteration

$$x^{t+1} = x^t - \alpha f'(x^t),$$

and requiring $O(\kappa \log(1/\epsilon))$ iterations.

Converge Rate of Gradient Method

- To minimize a smooth objective

$$\min_{x \in \mathbb{R}^P} f(x),$$

the gradient method minimizes the approximation

$$x^{t+1} = \arg\min_{x \in \mathbb{R}^P} f(x^t) + f'(x^t)^T(x - x^t) + \frac{1}{2\alpha}\|x - x^t\|^2.$$

yielding the iteration

$$x^{t+1} = x^t - \alpha f'(x^t),$$

and requiring $O(\kappa \log(1/\epsilon))$ iterations.

- Accelerated gradient method requires $O(\sqrt{\kappa} \log(1/\epsilon))$.
- Spectral gradient method is faster in practice.

Converge Rate of Proximal-Gradient Method

- To minimize a smooth objective

$$\min_{x \in \mathbb{R}^P} f(x),$$

the gradient method minimizes the approximation

$$x^{t+1} = \arg\min_{x \in \mathbb{R}^P} f(x^t) + f'(x^t)^T (x - x^t) + \frac{1}{2\alpha} \|x - x^t\|^2.$$

yielding the iteration

$$x^{t+1} = x^t - \alpha f'(x^t),$$

and requiring $O(\kappa \log(1/\epsilon))$ iterations.

- Accelerated gradient method requires $O(\sqrt{\kappa} \log(1/\epsilon))$.
- Spectral gradient method is faster in practice.

Converge Rate of Proximal-Gradient Method

- To minimize a smooth plus simple objective

$$\min_{x \in \mathbb{R}^P} f(x) + r(x),$$

the gradient method minimizes the approximation

$$x^{t+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^P} f(x^t) + f'(x^t)^T(x - x^t) + \frac{1}{2\alpha}\|x - x^t\|^2.$$

yielding the iteration

$$x^{t+1} = x^t - \alpha f'(x^t),$$

and requiring $O(\kappa \log(1/\epsilon))$ iterations.

- Accelerated gradient method requires $O(\sqrt{\kappa} \log(1/\epsilon))$.
- Spectral gradient method is faster in practice.

Converge Rate of Proximal-Gradient Method

- To minimize a smooth plus simple objective

$$\min_{x \in \mathbb{R}^P} f(x) + r(x),$$

the proximal-gradient method minimizes the approximation

$$x^{t+1} = \arg\min_{x \in \mathbb{R}^P} f(x^t) + f'(x^t)^T(x - x^t) + \frac{1}{2\alpha}\|x - x^t\|^2 + r(x).$$

yielding the iteration

$$x^{t+1} = \text{prox}_{\alpha r}[x^t - \alpha f'(x^t)],$$

and requiring $O(\kappa \log(1/\epsilon))$ iterations.

- Accelerated gradient method requires $O(\sqrt{\kappa} \log(1/\epsilon))$.
- Spectral gradient method is faster in practice.

Converge Rate of Proximal-Gradient Method

- To minimize a smooth plus simple objective

$$\min_{x \in \mathbb{R}^P} f(x) + r(x),$$

the proximal-gradient method minimizes the approximation

$$x^{t+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^P} f(x^t) + f'(x^t)^T(x - x^t) + \frac{1}{2\alpha}\|x - x^t\|^2 + r(x).$$

yielding the iteration

$$x^{t+1} = \text{prox}_{\alpha r}[x^t - \alpha f'(x^t)],$$

and still requiring $O(\kappa \log(1/\epsilon))$ iterations.

- Accelerated gradient method requires $O(\sqrt{\kappa} \log(1/\epsilon))$.
- Spectral gradient method is faster in practice.

Converge Rate of Proximal-Gradient Method

- To minimize a smooth plus simple objective

$$\min_{x \in \mathbb{R}^P} f(x) + r(x),$$

the proximal-gradient method minimizes the approximation

$$x^{t+1} = \underset{x \in \mathbb{R}^P}{\arg\min}\, f(x^t) + f'(x^t)^T(x - x^t) + \frac{1}{2\alpha}\|x - x^t\|^2 + r(x).$$

yielding the iteration

$$x^{t+1} = \text{prox}_{\alpha r}[x^t - \alpha f'(x^t)],$$

and still requiring $O(\kappa \log(1/\epsilon))$ iterations.

- Accelerated proximal-gradient method requires $O(\sqrt{\kappa}\log(1/\epsilon))$.
- Spectral proximal-gradient method is faster in practice.

Converge Rate of Proximal-Gradient Method

- To minimize a smooth plus simple objective

$$\min_{x \in \mathbb{R}^P} f(x) + r(x),$$

the proximal-gradient method minimizes the approximation

$$x^{t+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^P} f(x^t) + f'(x^t)^T(x - x^t) + \frac{1}{2\alpha}\|x - x^t\|^2 + r(x).$$

yielding the iteration

$$x^{t+1} = \text{prox}_{\alpha r}[x^t - \alpha f'(x^t)],$$

and still requiring $O(\kappa \log(1/\epsilon))$ iterations.

- Accelerated proximal-gradient method requires $O(\sqrt{\kappa} \log(1/\epsilon))$.
- Spectral proximal-gradient method is faster in practice.
- Non-smooth optimization at the speed of smooth optimization.

Proximal Operator, Iterative Soft Thresholding

- The proximal operator is the solution to

$$\text{prox}_r[y] = \arg\min_{x \in \mathbb{R}^P} \; r(x) + \frac{1}{2}\|x - y\|^2.$$

Proximal Operator, Iterative Soft Thresholding

- The proximal operator is the solution to

$$\text{prox}_r[y] = \underset{x \in \mathbb{R}^P}{\arg\min} \; r(x) + \frac{1}{2}\|x - y\|^2.$$

- For L1-regularization, we obtain iterative soft-thresholding:

$$x^+ = \text{softThresh}_{\alpha\lambda}[x - \alpha f'(x)].$$

Proximal Operator, Iterative Soft Thresholding

- The proximal operator is the solution to

$$\text{prox}_r[y] = \mathop{\arg\min}_{x \in \mathbb{R}^P} \; r(x) + \frac{1}{2}\|x - y\|^2.$$

- For L1-regularization, we obtain iterative soft-thresholding:

$$x^+ = \text{softThresh}_{\alpha\lambda}[x - \alpha f'(x)].$$

- Example with $\lambda = 1$:

Input	Threshold	Soft-Threshold
$\begin{bmatrix} 0.6715 \\ -1.2075 \\ 0.7172 \\ 1.6302 \\ 0.4889 \end{bmatrix}$		

Proximal Operator, Iterative Soft Thresholding

- The proximal operator is the solution to

$$\text{prox}_r[y] = \mathop{\arg\min}_{x \in \mathbb{R}^P} r(x) + \frac{1}{2}\|x - y\|^2.$$

- For L1-regularization, we obtain iterative soft-thresholding:

$$x^+ = \text{softThresh}_{\alpha\lambda}[x - \alpha f'(x)].$$

- Example with $\lambda = 1$:

Input	Threshold	Soft-Threshold
$\begin{bmatrix} 0.6715 \\ -1.2075 \\ 0.7172 \\ 1.6302 \\ 0.4889 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -1.2075 \\ 0 \\ 1.6302 \\ 0 \end{bmatrix}$	

Proximal Operator, Iterative Soft Thresholding

- The proximal operator is the solution to

$$\text{prox}_r[y] = \mathop{\arg\min}_{x \in \mathbb{R}^P} r(x) + \frac{1}{2}\|x - y\|^2.$$

- For L1-regularization, we obtain iterative soft-thresholding:

$$x^+ = \text{softThresh}_{\alpha\lambda}[x - \alpha f'(x)].$$

- Example with $\lambda = 1$:

Input	Threshold	Soft-Threshold
$\begin{bmatrix} 0.6715 \\ -1.2075 \\ 0.7172 \\ 1.6302 \\ 0.4889 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -1.2075 \\ 0 \\ 1.6302 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -0.2075 \\ 0 \\ 0.6302 \\ 0 \end{bmatrix}$

Special case of Projected-Gradient Methods

- Projected-gradient methods are another special case:

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

Special case of Projected-Gradient Methods

- Projected-gradient methods are another special case:

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

gives

$$x^+ = \text{project}_{\mathcal{C}}[x - \alpha f'(x)],$$

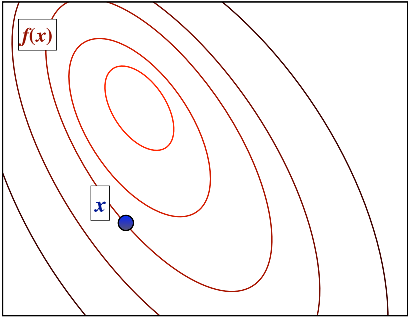
Special case of Projected-Gradient Methods

- Projected-gradient methods are another special case:

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

gives

$$x^+ = \text{project}_{\mathcal{C}}[x - \alpha f'(x)],$$



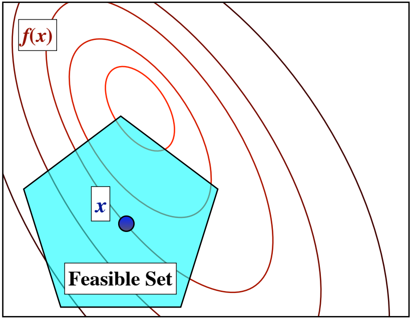
Special case of Projected-Gradient Methods

- Projected-gradient methods are another special case:

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

gives

$$x^+ = \text{project}_{\mathcal{C}}[x - \alpha f'(x)],$$



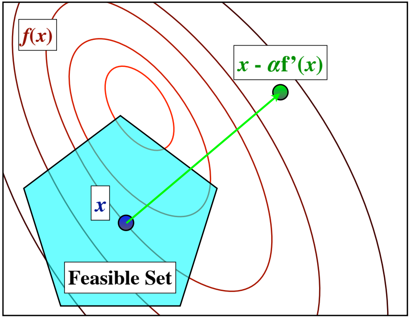
Special case of Projected-Gradient Methods

- Projected-gradient methods are another special case:

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

gives

$$x^+ = \text{project}_{\mathcal{C}}[x - \alpha f'(x)],$$



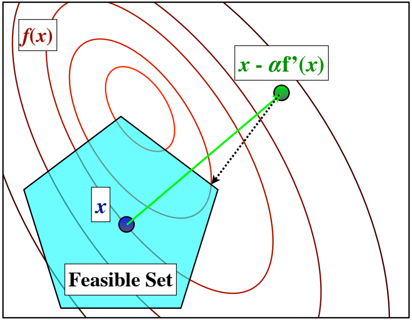
Special case of Projected-Gradient Methods

- Projected-gradient methods are another special case:

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

gives

$$x^+ = \text{project}_{\mathcal{C}}[x - \alpha f'(x)],$$



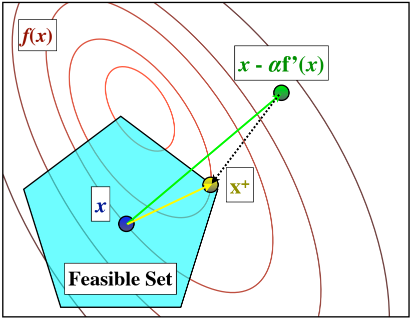
Special case of Projected-Gradient Methods

- Projected-gradient methods are another special case:

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

gives

$$x^+ = \text{project}_{\mathcal{C}}[x - \alpha f'(x)],$$



Exact Proximal-Gradient Methods

- For what problems can we apply these methods?

Exact Proximal-Gradient Methods

- For what problems can we apply these methods?
- We can efficiently compute the proximity operator for:
  1. L1-Regularization.

Exact Proximal-Gradient Methods

- For what problems can we apply these methods?
- We can efficiently compute the proximity operator for:
  1. L1-Regularization.
  2. Group ℓ_1-Regularization.

Exact Proximal-Gradient Methods

- For what problems can we apply these methods?
- We can efficiently compute the proximity operator for:
  1. L1-Regularization.
  2. Group ℓ_1-Regularization.
  3. Lower and upper bounds.

Exact Proximal-Gradient Methods

- For what problems can we apply these methods?
- We can efficiently compute the proximity operator for:
  1. L1-Regularization.
  2. Group ℓ_1-Regularization.
  3. Lower and upper bounds.
  4. One linear constraint.

Exact Proximal-Gradient Methods

- For what problems can we apply these methods?
- We can efficiently compute the proximity operator for:
  1. L1-Regularization.
  2. Group ℓ_1-Regularization.
  3. Lower and upper bounds.
  4. One linear constraint.
  5. Probability constraints.

Exact Proximal-Gradient Methods

- For what problems can we apply these methods?
- We can efficiently compute the proximity operator for:
  1. L1-Regularization.
  2. Group ℓ_1-Regularization.
  3. Lower and upper bounds.
  4. One linear constraint.
  5. Probability constraints.
  6. A few other simple regularizers/constraints.

Exact Proximal-Gradient Methods

- For what problems can we apply these methods?
- We can efficiently compute the proximity operator for:
  1. L1-Regularization.
  2. Group ℓ_1-Regularization.
  3. Lower and upper bounds.
  4. One linear constraint.
  5. Probability constraints.
  6. A few other simple regularizers/constraints.
- For many problems we can not efficiently compute this operator.

Inexact Proximal-Gradient Methods

- We can efficiently approximate the proximity operator for:

Inexact Proximal-Gradient Methods

- We can efficiently approximate the proximity operator for:
  1. Structured sparsity.

Inexact Proximal-Gradient Methods

- We can efficiently approximate the proximity operator for:
  1. Structured sparsity.
  2. Penalties on the differences between variables.

Inexact Proximal-Gradient Methods

- We can efficiently approximate the proximity operator for:
  1. Structured sparsity.
  2. Penalties on the differences between variables.
  3. Regularizers and constraints on the singular values of matrices.

Inexact Proximal-Gradient Methods

- We can efficiently approximate the proximity operator for:
  1. Structured sparsity.
  2. Penalties on the differences between variables.
  3. Regularizers and constraints on the singular values of matrices.
  4. Sums of simple functions.

Inexact Proximal-Gradient Methods

- We can efficiently approximate the proximity operator for:
  1. Structured sparsity.
  2. Penalties on the differences between variables.
  3. Regularizers and constraints on the singular values of matrices.
  4. Sums of simple functions.
- Many recent works use inexact proximal-gradient methods:

  Cai et al. [2010], Liu & Ye [2010], Barbero & Sra [2011], Fadili & Peyré [2011], Ma et al. [2011]

Inexact Proximal-Gradient Methods

- We can efficiently approximate the proximity operator for:
  1. Structured sparsity.
  2. Penalties on the differences between variables.
  3. Regularizers and constraints on the singular values of matrices.
  4. Sums of simple functions.
- Many recent works use inexact proximal-gradient methods:

  Cai et al. [2010], Liu & Ye [2010], Barbero & Sra [2011], Fadili & Peyré [2011], Ma et al. [2011]
- Do inexact methods have the $O(\kappa \log(1/\epsilon))$ rate?

Inexact Proximal-Gradient Methods

- We can efficiently approximate the proximity operator for:
  1. Structured sparsity.
  2. Penalties on the differences between variables.
  3. Regularizers and constraints on the singular values of matrices.
  4. Sums of simple functions.
- Many recent works use inexact proximal-gradient methods:

  Cai et al. [2010], Liu & Ye [2010], Barbero & Sra [2011], Fadili & Peyré [2011], Ma et al. [2011]
- Do inexact methods have the $O(\kappa \log(1/\epsilon))$ rate?
  - *Yes, if the errors are appropriately controlled*. [Schmidt et al., 2011]

Convergence Rate of Inexact Proximal-Gradient

Proposition *[Schmidt et al., 2011] If the sequences of gradient errors* $\{||e_t||\}$ *and proximal errors* $\{\sqrt{\varepsilon_t}\}$ *are in* $\{O((1-\kappa^{-1})^t)\}$*, then the inexact proximal-gradient method requires* $O(\kappa \log(1/\epsilon))$ *iterations.*

Convergence Rate of Inexact Proximal-Gradient

Proposition *[Schmidt et al., 2011] If the sequences of gradient errors* $\{||e_t||\}$ *and proximal errors* $\{\sqrt{\varepsilon_t}\}$ *are in* $\{O((1-\kappa^{-1})^t)\}$*, then the inexact proximal-gradient method requires* $O(\kappa \log(1/\epsilon))$ *iterations.*

- Classic result as a special case (constants are good).
- The rates degrades gracefully if the errors are larger.

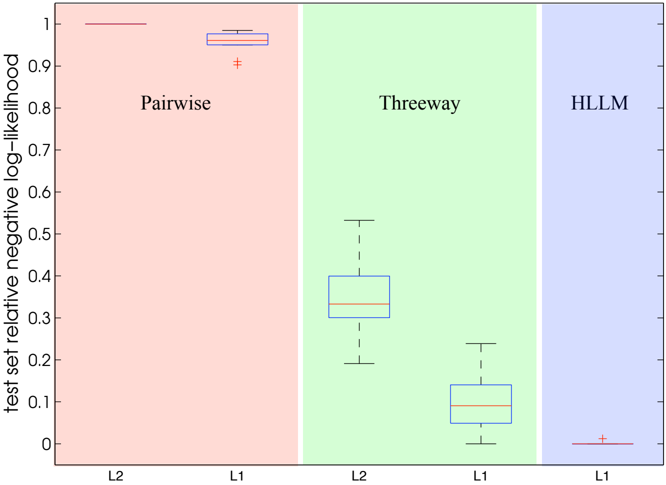
Convergence Rate of Inexact Proximal-Gradient

Proposition *[Schmidt et al., 2011] If the sequences of gradient errors $\{||e_t||\}$ and proximal errors $\{\sqrt{\varepsilon_t}\}$ are in $\{O((1-\kappa^{-1})^t)\}$, then the inexact proximal-gradient method requires $O(\kappa \log(1/\epsilon))$ iterations.*

- Classic result as a special case (constants are good).
- The rates degrades gracefully if the errors are larger.
- We also showed the $O(\sqrt{\kappa}\log(1/\epsilon))$ accelerated method rate.
- We also considered weaker convexity assumptions on f.
- Huge improvement in practice over black-box methods.

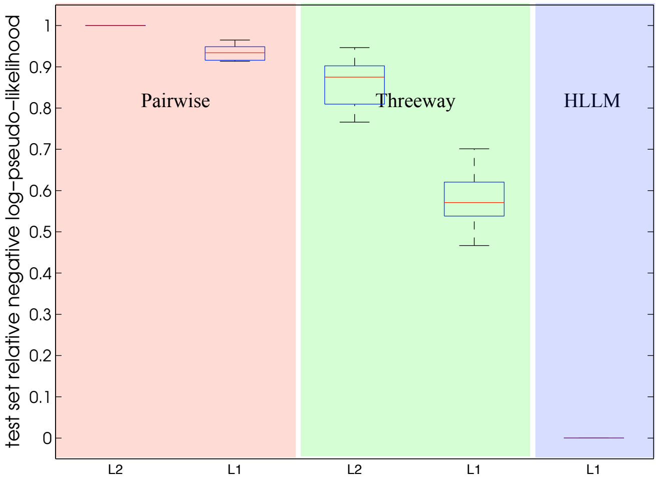
Flow Cytometry Data

Using structured sparsity to fit a hierarchical log-linear model (HLLM):



Traffic Flow Data

Using structured sparsity to fit a hierarchical log-linear model (HLLM):



Discussion

- Theoretical justification for what works in practice.
- Significantly extends class of tractable problems.
- Many subsequent applications with inexact proximal operators:
  - Genomic expression, model predictive control, neuroimaging, satellite image fusion, simulating flow fields.

Discussion

- Theoretical justification for what works in practice.
- Significantly extends class of tractable problems.
- Many subsequent applications with inexact proximal operators:
  - Genomic expression, model predictive control, neuroimaging, satellite image fusion, simulating flow fields.

- But, it assumes computing $f'(x)$ and $\text{prox}_r[x]$ have similar cost.

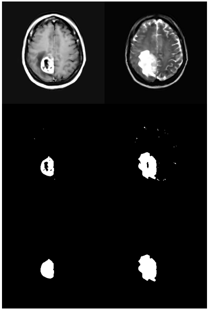
Discussion

- Theoretical justification for what works in practice.
- Significantly extends class of tractable problems.
- Many subsequent applications with inexact proximal operators:
  - Genomic expression, model predictive control, neuroimaging, satellite image fusion, simulating flow fields.

- But, it assumes computing $f'(x)$ and $\text{prox}_r[x]$ have similar cost.
- Often $f'(x)$ is much more expensive:
  - We may have a large dataset.
  - Data-fitting term might be complex.
- Particularly true for structured output prediction:
  - Text, biological sequences, speech, images, matchings, graphs.

Motivation: Automatic Brain Tumor Segmentation

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



- Data-fitting term is solution of 8-million node graph-cut problem.

Outline

1. Structured sparsity (inexact proximal-gradient method)
2. Learning dependencies (costly models with simple constraints)
3. Fitting a huge dataset (stochastic average gradient)

Motivation: Graphical Model Structure Learning

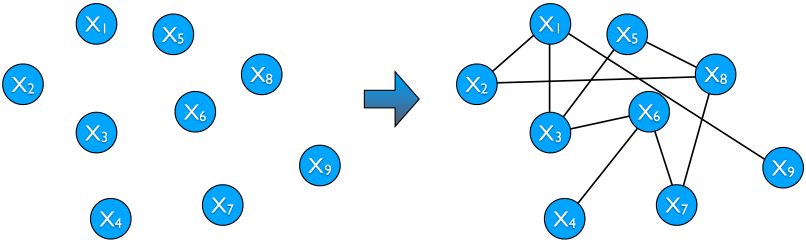
Discovering the dependencies between variables:

car	drive	files	hockey	mac	league	pc	win
0	0	1	0	1	0	1	0
0	0	0	1	0	1	0	1
1	1	0	0	0	0	0	0
0	1	1	0	1	0	0	0
0	0	1	0	0	0	1	1

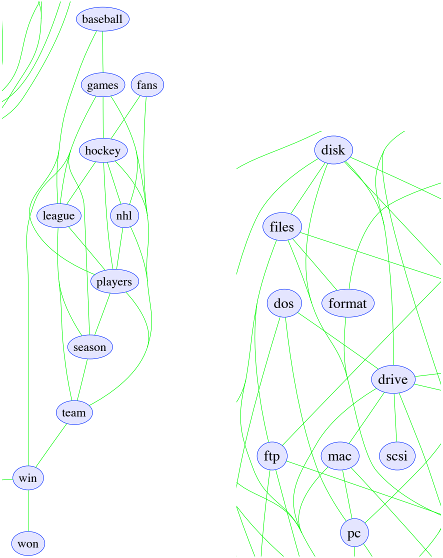
Motivation: Graphical Model Structure Learning

Discovering the dependencies between variables:

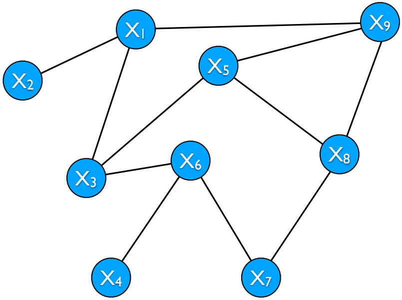
car	drive	files	hockey	mac	league	pc	win
0	0	1	0	1	0	1	0
0	0	0	1	0	1	0	1
1	1	0	0	0	0	0	0
0	1	1	0	1	0	0	0
0	0	1	0	0	0	1	1



Example: Graphical Model Structure Learning

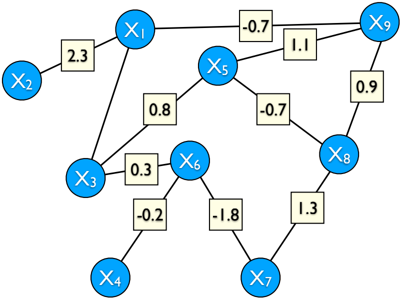


Structure Learning with ℓ_1-Regularization



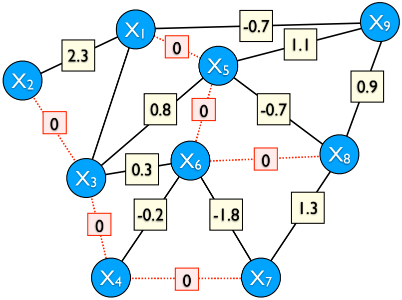
- We want to fit a Markov random field with unknown structure.

Structure Learning with ℓ_1-Regularization



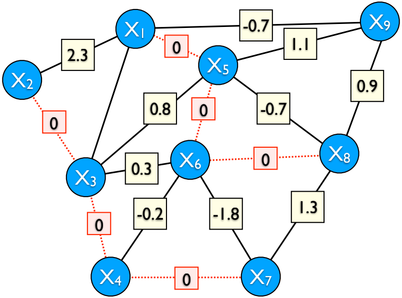
- We want to fit a Markov random field with unknown structure.

Structure Learning with ℓ_1-Regularization



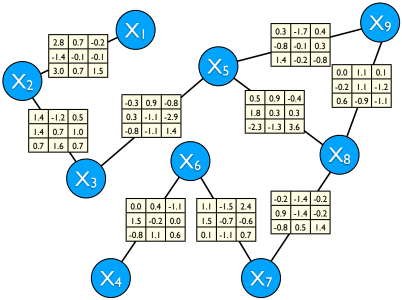
- We want to fit a Markov random field with unknown structure.

Structure Learning with ℓ_1-Regularization



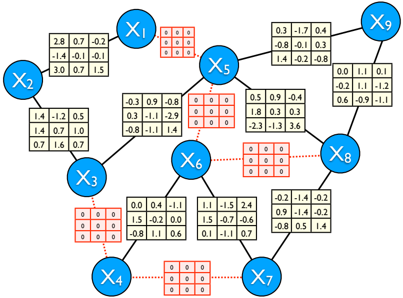
- We want to fit a Markov random field with unknown structure.
- Learn a sparse structure by ℓ_1-regularization of edge weights.

Structure Learning with Group ℓ_1-Regularization



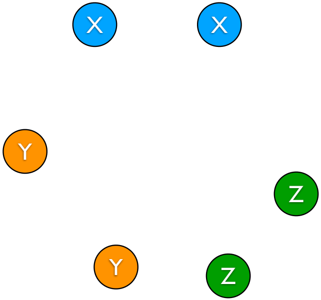
- In some cases, we want sparsity in groups of parameters:
  1. Multi-class variables [Lee et al., 2006].

Structure Learning with Group ℓ_1-Regularization



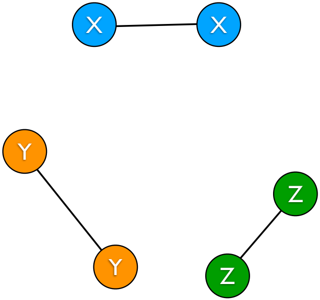
- In some cases, we want sparsity in groups of parameters:
  1. Multi-class variables [Lee et al., 2006].

Structure Learning with Group ℓ_1-Regularization



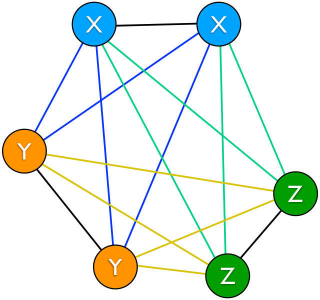
- In some cases, we want sparsity in groups of parameters:
  1. Multi-class variables [Lee et al., 2006].
  2. Blockwise-sparsity [Duchi et al., 2008].

Structure Learning with Group ℓ_1-Regularization



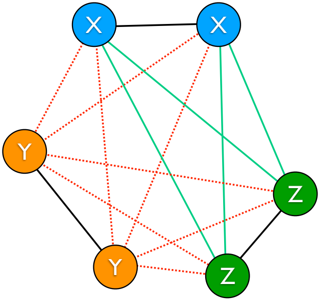
- In some cases, we want sparsity in groups of parameters:
  1. Multi-class variables [Lee et al., 2006].
  2. Blockwise-sparsity [Duchi et al., 2008].

Structure Learning with Group ℓ_1-Regularization



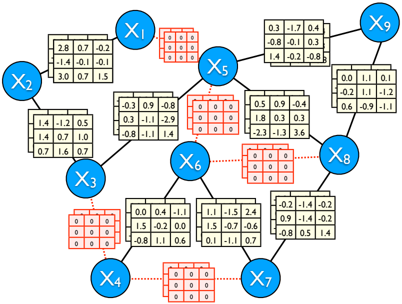
- In some cases, we want sparsity in groups of parameters:
  1. Multi-class variables [Lee et al., 2006].
  2. Blockwise-sparsity [Duchi et al., 2008].

Structure Learning with Group ℓ_1-Regularization



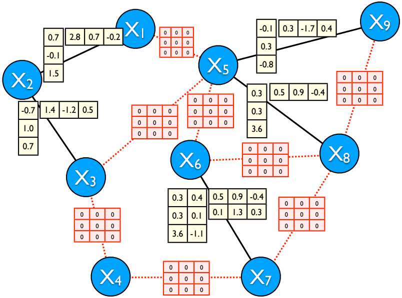
- In some cases, we want sparsity in groups of parameters:
  1. Multi-class variables [Lee et al., 2006].
  2. Blockwise-sparsity [Duchi et al., 2008].

Structure Learning with Group ℓ_1-Regularization



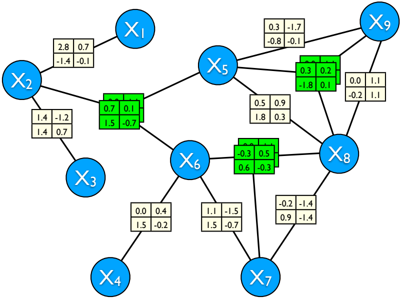
- In some cases, we want sparsity in groups of parameters:
  1. Multi-class variables [Lee et al., 2006].
  2. Blockwise-sparsity [Duchi et al., 2008].
  3. Conditional random fields [Schmidt et al., 2008].

Structure Learning with Group ℓ_1-Regularization



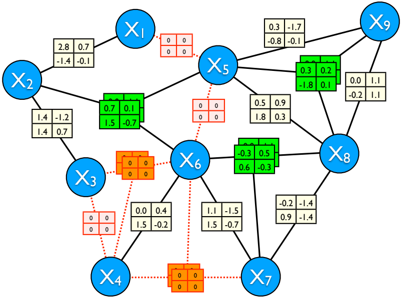
- In some cases, we want sparsity in groups of parameters:
  1. Multi-class variables [Lee et al., 2006].
  2. Blockwise-sparsity [Duchi et al., 2008].
  3. Conditional random fields [Schmidt et al., 2008].
  4. Low-rank Edges [Schmidt, 2010].

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].
    4. Low-rank Edges [Schmidt, 2010].
    5. Higher-order models [Schmidt & Murphy, 2010].

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].
  4. Low-rank Edges [Schmidt, 2010].
  5. Higher-order models [Schmidt & Murphy, 2010].

Costly Data-Fitting Term, Simple Regularizer

- These problems and many others have the form:

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

costly smooth + simple

Costly Data-Fitting Term, Simple Regularizer

- These problems and many others have the form:

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

costly smooth + simple

- Different than classic optimization (like linear programming).

(cheap smooth plus complex non-smooth)

Costly Data-Fitting Term, Simple Regularizer

- These problems and many others have the form:

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

costly smooth + simple

- Different than classic optimization (like linear programming).

(cheap smooth plus complex non-smooth)

- Inspiration from the smooth case:
  - For smooth high-dimensional problems, L-BFGS outperform accelerated/spectral gradient methods.

Quasi-Newton Methods

- Gradient method for optimizing a smooth f:

$$x^+ = x - \alpha f'(x).$$

Quasi-Newton Methods

- Gradient method for optimizing a smooth f:

$$x^+ = x - \alpha f'(x).$$

- Newton-like methods alternatively use:

$$x^+ = x - \alpha H^{-1} f'(x).$$

- H approximates the second-derivative matrix.

Quasi-Newton Methods

- Gradient method for optimizing a smooth f:

$$x^+ = x - \alpha f'(x).$$

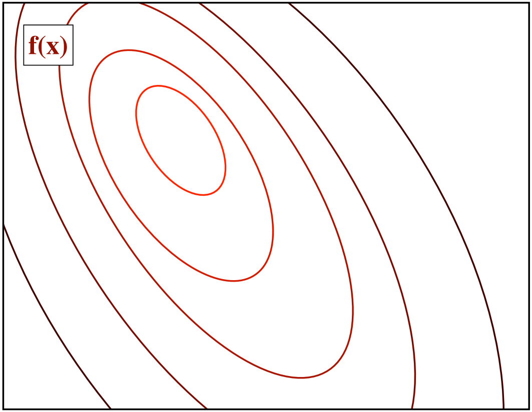
- Newton-like methods alternatively use:

$$x^+ = x - \alpha H^{-1} f'(x).$$

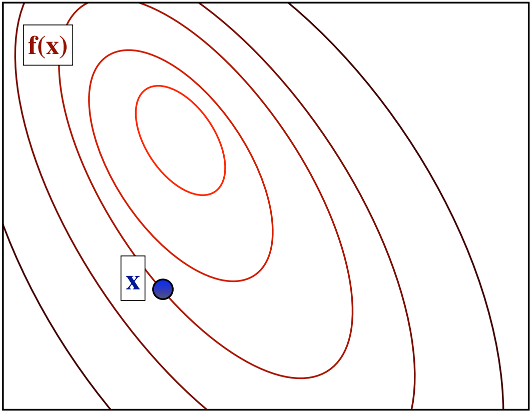
- H approximates the second-derivative matrix.
- L-BFGS is a particular strategy to choose the H values:
  - Based on gradient differences.
  - Linear storage and linear time.

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

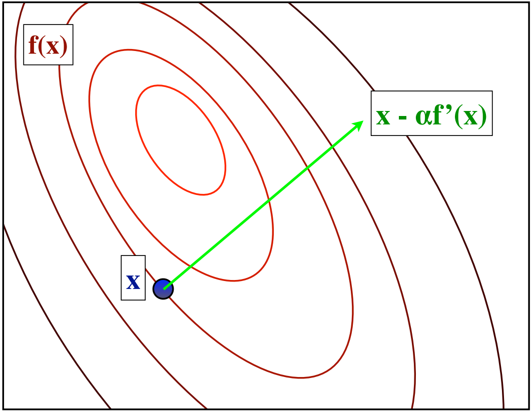
Gradient Method and Newton’s Method



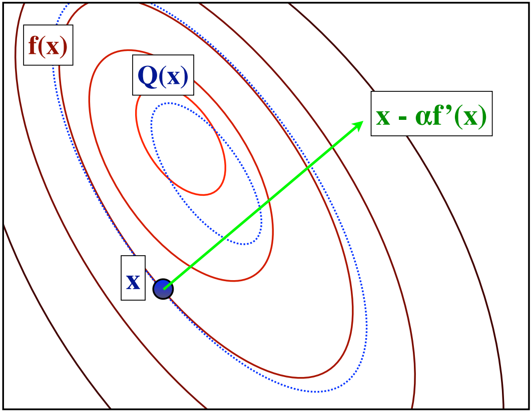
Gradient Method and Newton's Method



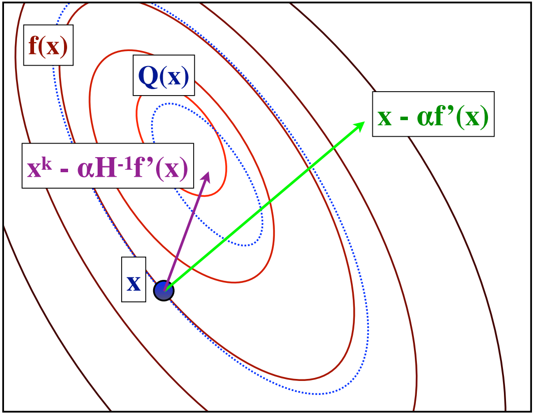
Gradient Method and Newton's Method



Gradient Method and Newton's Method



Gradient Method and Newton’s Method



Naive Proximal Quasi-Newton Method

- Proximal-gradient method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha f'(x)].$$

Naive Proximal Quasi-Newton Method

- Proximal-gradient method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha f'(x)].$$

- Can we just plug in the Newton-like step?

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)].$$

Naive Proximal Quasi-Newton Method

- Proximal-gradient method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha f'(x)].$$

- Can we just plug in the Newton-like step?

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)].$$

- NO!

Naive Proximal Quasi-Newton Method

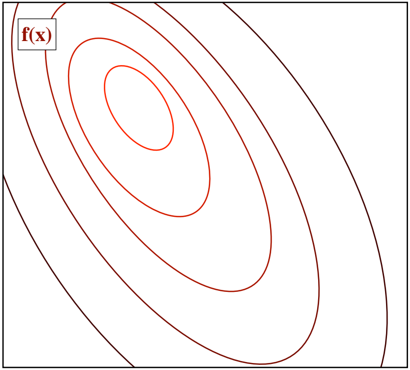
- Proximal-gradient method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha f'(x)].$$

- Can we just plug in the Newton-like step?

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)].$$

- NO!



Naive Proximal Quasi-Newton Method

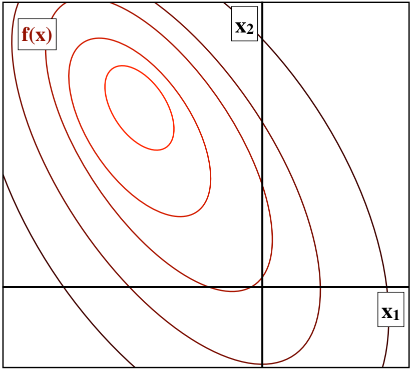
- Proximal-gradient method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha f'(x)].$$

- Can we just plug in the Newton-like step?

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)].$$

- NO!



Naive Proximal Quasi-Newton Method

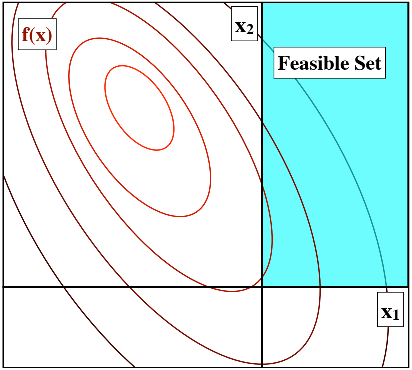
- Proximal-gradient method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha f'(x)].$$

- Can we just plug in the Newton-like step?

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)].$$

- NO!



Naive Proximal Quasi-Newton Method

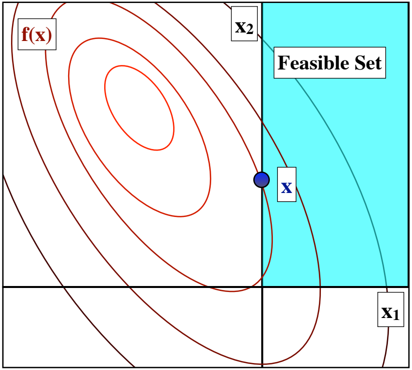
- Proximal-gradient method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha f'(x)].$$

- Can we just plug in the Newton-like step?

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)].$$

- NO!



Naive Proximal Quasi-Newton Method

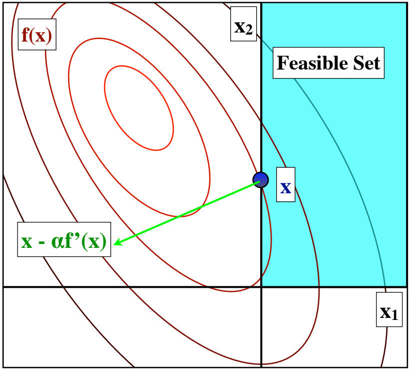
- Proximal-gradient method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha f'(x)].$$

- Can we just plug in the Newton-like step?

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)].$$

- NO!



Naive Proximal Quasi-Newton Method

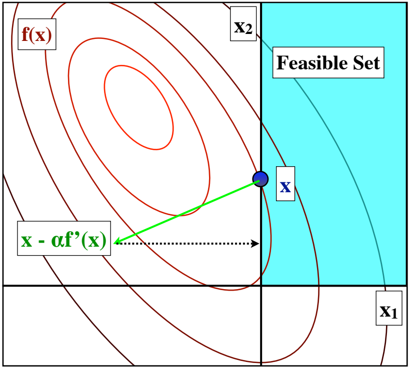
- Proximal-gradient method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha f'(x)].$$

- Can we just plug in the Newton-like step?

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)].$$

- NO!



Naive Proximal Quasi-Newton Method

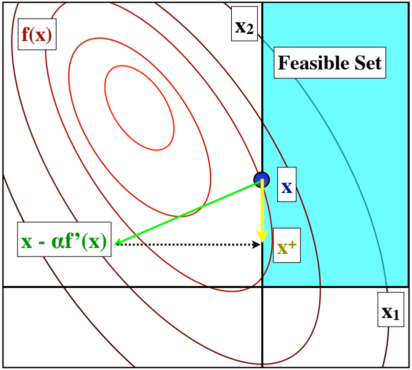
- Proximal-gradient method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha f'(x)].$$

- Can we just plug in the Newton-like step?

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)].$$

- NO!



Naive Proximal Quasi-Newton Method

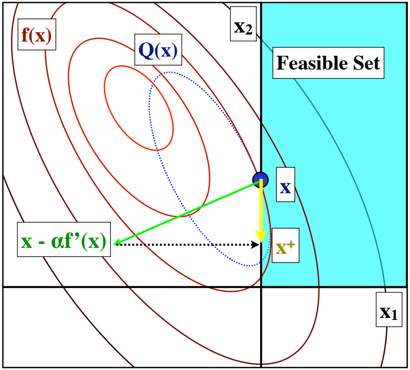
- Proximal-gradient method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha f'(x)].$$

- Can we just plug in the Newton-like step?

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)].$$

- NO!



Naive Proximal Quasi-Newton Method

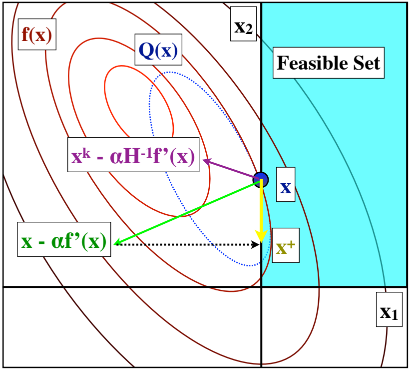
- Proximal-gradient method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha f'(x)].$$

- Can we just plug in the Newton-like step?

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)].$$

- NO!



Naive Proximal Quasi-Newton Method

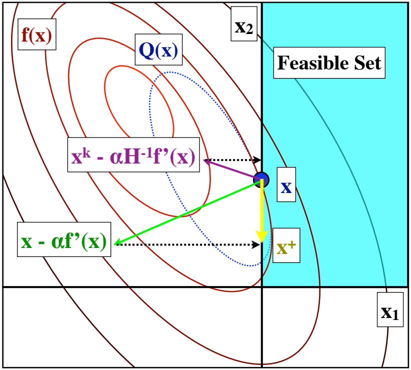
- Proximal-gradient method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha f'(x)].$$

- Can we just plug in the Newton-like step?

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)].$$

- NO!



Naive Proximal Quasi-Newton Method

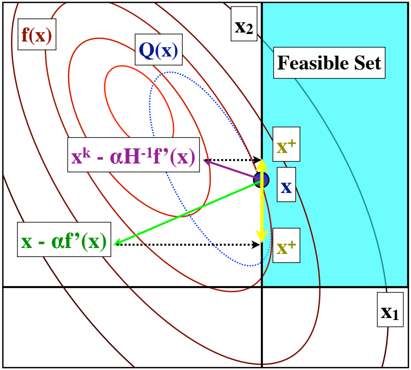
- Proximal-gradient method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha f'(x)].$$

- Can we just plug in the Newton-like step?

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)].$$

- NO!



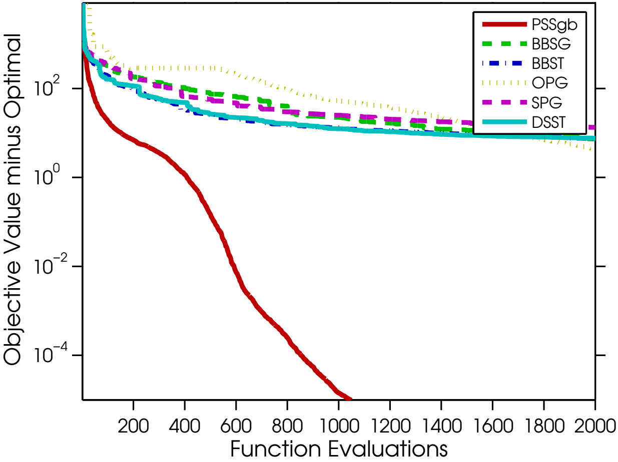
Two-Metric (Sub)Gradient Projection

- In some cases, we can modify H to make this work:
  - Bound constraints.
  - Probability constraints.
  - L1-regularization.
- Two-metric (sub)gradient projection.

  [Gafni & Bertskeas, 1984, Schmidt, 2010].

Comparing to accelerated/spectral/diagonal gradient

Comparing to methods that do not use L-BFGS (sido data):



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

Inexact Proximal-Newton

- The broken proximal-Newton method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)],$$

with the Euclidean proximal operator:

$$\text{prox}_r[y] = \mathop{\arg\min}_{x \in \mathbb{R}^P} \; r(x) + \frac{1}{2}\|x - y\|^2,$$

Inexact Proximal-Newton

- The fixed proximal-Newton method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)]_H,$$

with the Euclidean proximal operator:

$$\text{prox}_r[y] = \operatorname*{arg\,min}_{x \in \mathbb{R}^P} \; r(x) + \frac{1}{2}\|x - y\|^2,$$

Inexact Proximal-Newton

- The fixed proximal-Newton method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)]_H,$$

with the non-Euclidean proximal operator:

$$\text{prox}_r[y]_H = \underset{x \in \mathbb{R}^P}{\arg\min}\ r(x) + \frac{1}{2}\|x - y\|_H^2,$$

where $\|x\|_H^2 = x^T H x$.

Inexact Proximal-Newton

- The fixed proximal-Newton method:

$$x^{+} = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)]_{H},$$

with the non-Euclidean proximal operator:

$$\text{prox}_{r}[y]_{H} = \underset{x \in \mathbb{R}^P}{\arg\min}\ r(x) + \frac{1}{2}\|x - y\|_{H}^{2},$$

where $\|x\|_{H}^{2} = x^{T} H x$.

- Non-smooth Newton-like method
- Same convergence properties as smooth case.

Inexact Proximal-Newton

- The fixed proximal-Newton method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)]_H,$$

with the non-Euclidean proximal operator:

$$\text{prox}_r[y]_H = \underset{x \in \mathbb{R}^P}{\arg\min}\ r(x) + \frac{1}{2}\|x - y\|_H^2,$$

where $\|x\|_H^2 = x^T H x$.

- Non-smooth Newton-like method
- Same convergence properties as smooth case.
- But, the prox is expensive even with a simple regularizer.

Inexact Proximal-Newton

- The fixed proximal-Newton method:

$$x^+ = \text{prox}_{\alpha r}[x - \alpha H^{-1} f'(x)]_H,$$

with the non-Euclidean proximal operator:

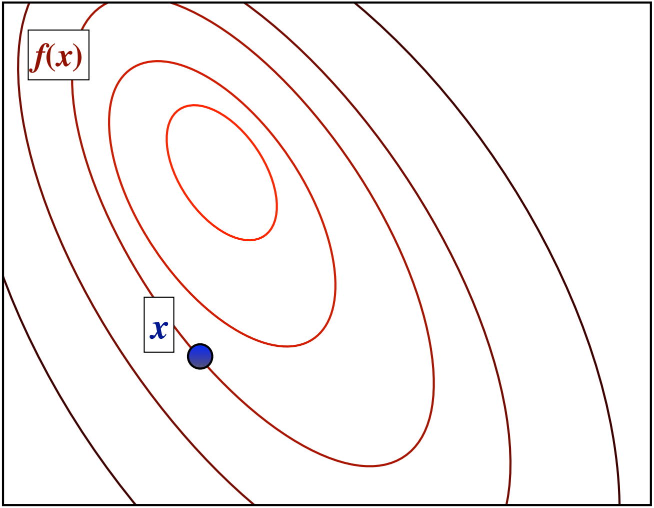
$$\text{prox}_r[y]_H = \underset{x \in \mathbb{R}^P}{\arg\min}\ r(x) + \frac{1}{2}\|x - y\|_H^2,$$

where $\|x\|_H^2 = x^T H x$.

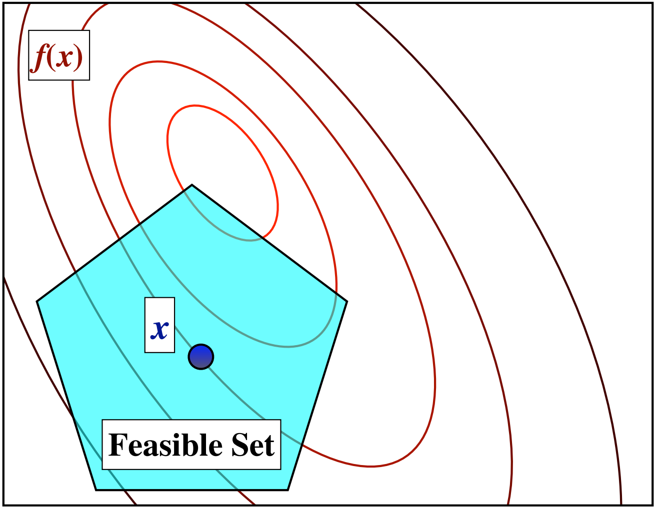
- Non-smooth Newton-like method
- Same convergence properties as smooth case.
- But, the prox is expensive even with a simple regularizer.
- Solution: use a cheap approximate solution.

(e.g., spectral proximal-gradient)

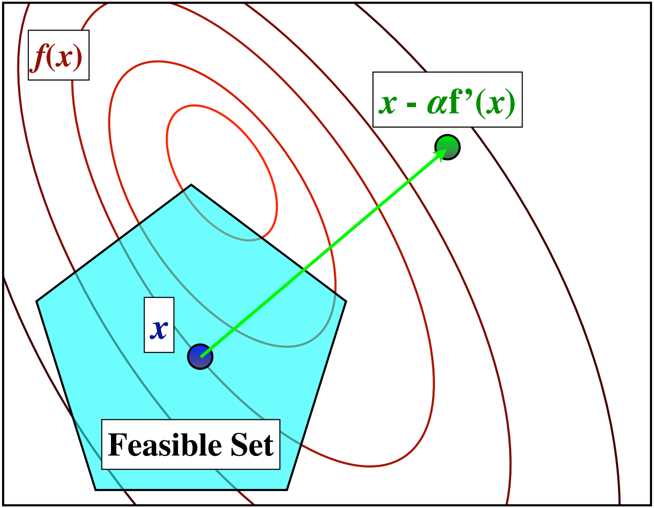
Inexact Projected Newton



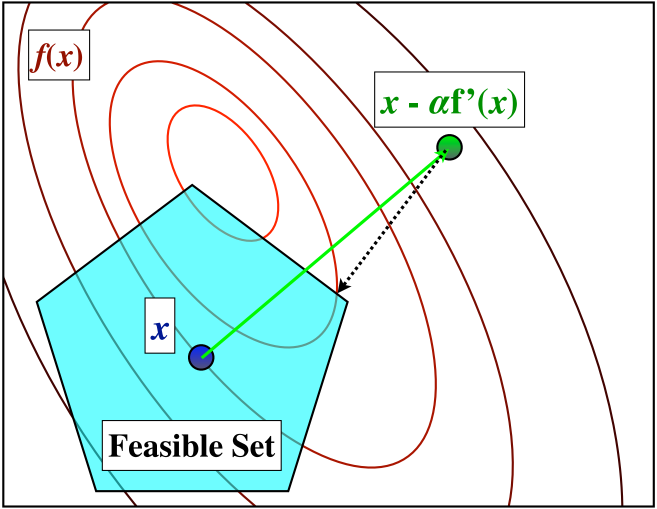
Inexact Projected Newton



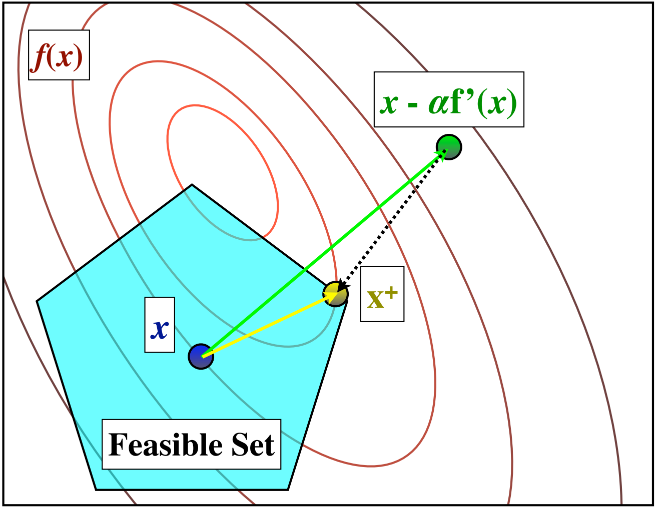
Inexact Projected Newton



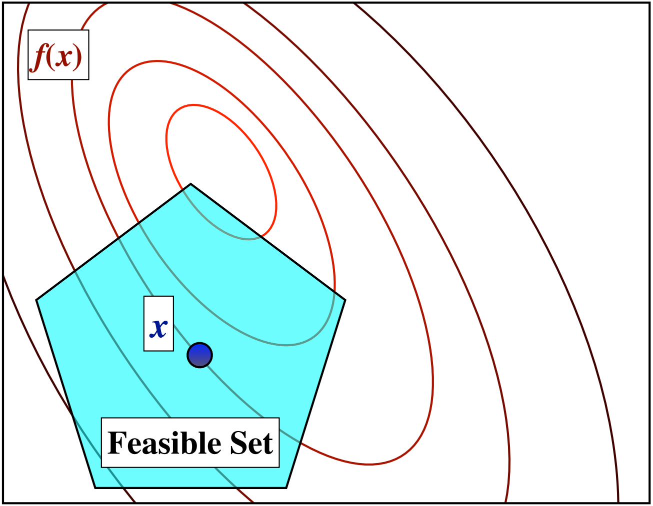
Inexact Projected Newton



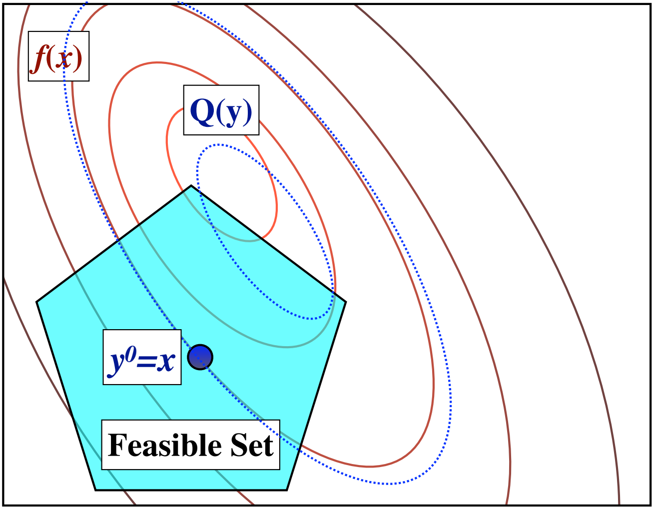
Inexact Projected Newton



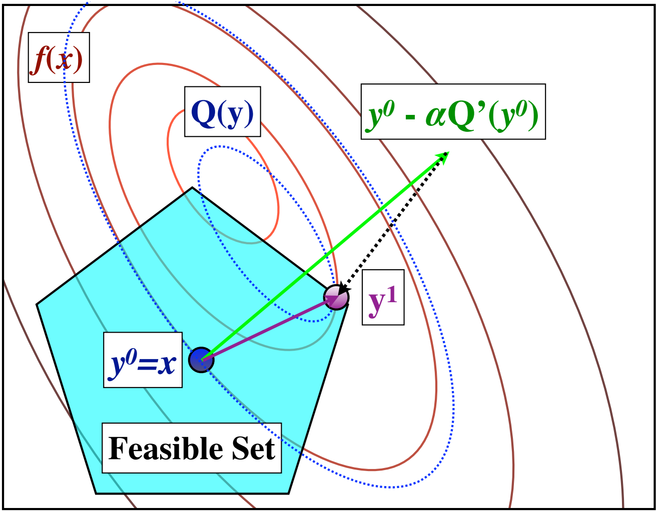
Inexact Projected Newton



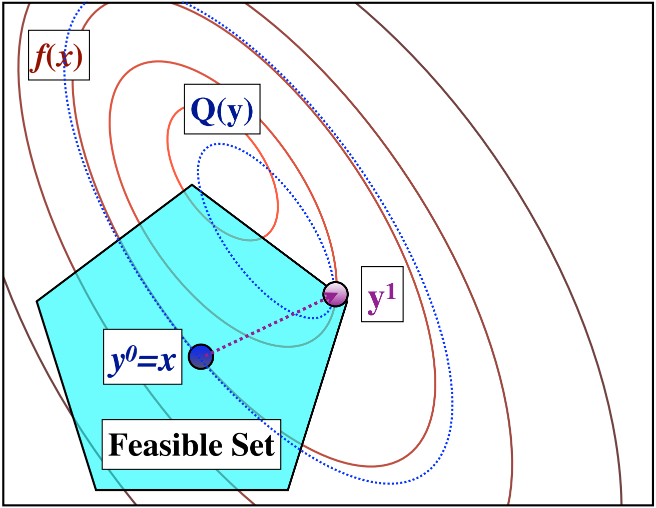
Inexact Projected Newton



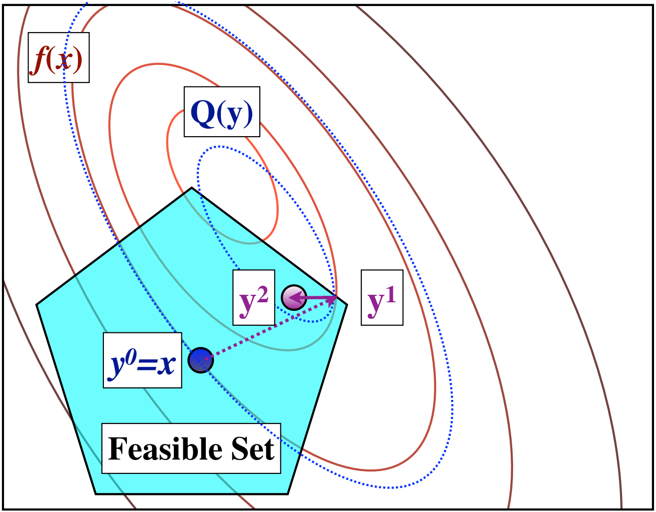
Inexact Projected Newton



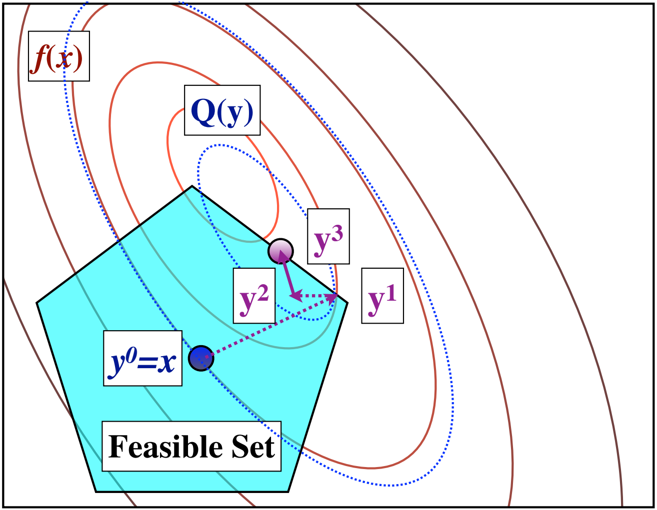
Inexact Projected Newton



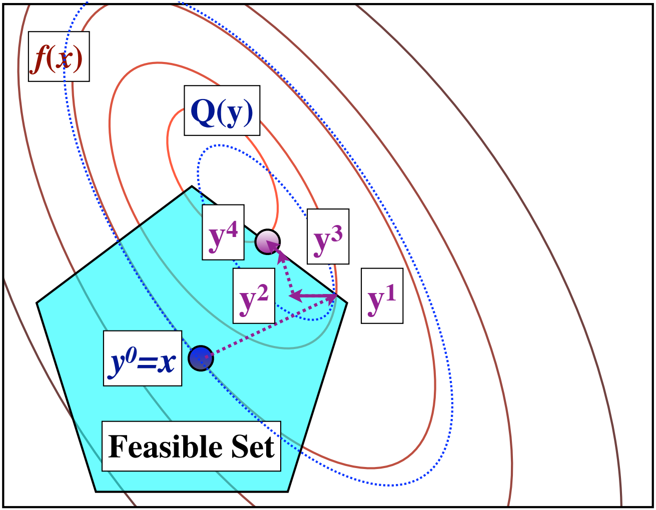
Inexact Projected Newton



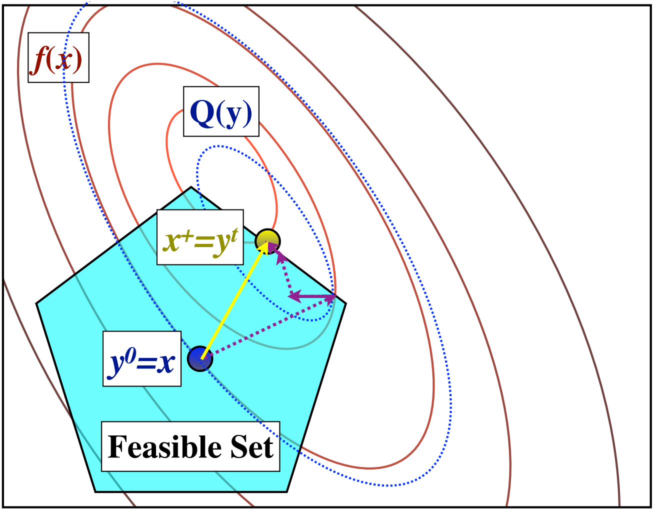
Inexact Projected Newton



Inexact Projected Newton



Inexact Projected Newton



Projected Quasi-Newton (PQN) Algorithm

- A proximal quasi-Newton (PQN) algorithm:

  [Schmidt et al., 2009, Schmidt, 2010]

Projected Quasi-Newton (PQN) Algorithm

- A proximal quasi-Newton (PQN) algorithm:

  [Schmidt et al., 2009, Schmidt, 2010]

  - Outer: evaluate $f(x)$ and $f'(x)$, use L-BFGS to update H.

Projected Quasi-Newton (PQN) Algorithm

- A proximal quasi-Newton (PQN) algorithm:

  [Schmidt et al., 2009, Schmidt, 2010]

  - Outer: evaluate $f(x)$ and $f'(x)$, use L-BFGS to update H.
  - Inner: spectral proximal-gradient to approximate proximal operator:
    - Requires multiplication by H (linear-time for L-BFGS).
    - Requires proximal operator of r (cheap for simple constraints).

Projected Quasi-Newton (PQN) Algorithm

- A proximal quasi-Newton (PQN) algorithm:

  [Schmidt et al., 2009, Schmidt, 2010]

  - Outer: evaluate $f(x)$ and $f'(x)$, use L-BFGS to update H.
  - Inner: spectral proximal-gradient to approximate proximal operator:
    - Requires multiplication by H (linear-time for L-BFGS).
    - Requires proximal operator of r (cheap for simple constraints).
  - For small α, one iteration is sufficient to give descent.

Projected Quasi-Newton (PQN) Algorithm

- A proximal quasi-Newton (PQN) algorithm:

  [Schmidt et al., 2009, Schmidt, 2010]

  - Outer: evaluate $f(x)$ and $f'(x)$, use L-BFGS to update H.
  - Inner: spectral proximal-gradient to approximate proximal operator:
    - Requires multiplication by H (linear-time for L-BFGS).
    - Requires proximal operator of r (cheap for simple constraints).
  - For small α, one iteration is sufficient to give descent.
- Cheap inner iterations lead to fewer expensive outer iterations.

Projected Quasi-Newton (PQN) Algorithm

- A proximal quasi-Newton (PQN) algorithm:

  [Schmidt et al., 2009, Schmidt, 2010]

  - Outer: evaluate $f(x)$ and $f'(x)$, use L-BFGS to update H.
  - Inner: spectral proximal-gradient to approximate proximal operator:
    - Requires multiplication by H (linear-time for L-BFGS).
    - Requires proximal operator of r (cheap for simple constraints).
  - For small α, one iteration is sufficient to give descent.
- Cheap inner iterations lead to fewer expensive outer iterations.
- "Optimizing costly functions with simple constraints".

Projected Quasi-Newton (PQN) Algorithm

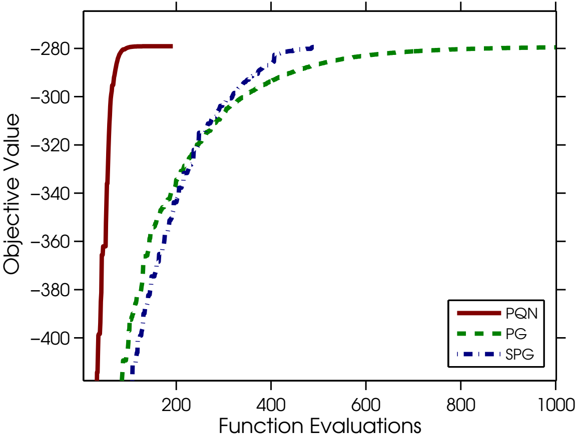
- A proximal quasi-Newton (PQN) algorithm:

  [Schmidt et al., 2009, Schmidt, 2010]

  - Outer: evaluate $f(x)$ and $f'(x)$, use L-BFGS to update H.
  - Inner: spectral proximal-gradient to approximate proximal operator:
    - Requires multiplication by H (linear-time for L-BFGS).
    - Requires proximal operator of r (cheap for simple constraints).
  - For small α, one iteration is sufficient to give descent.
- Cheap inner iterations lead to fewer expensive outer iterations.
- “Optimizing costly functions with simple constraints”.
- “Optimizing costly functions with simple regularizers”.

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].



Inexact Proximal Newton

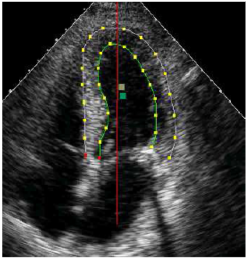
- The proximal quasi-Newton (PQN) approach:
  - "The projected quasi-Newton (PQN) algorithm [19, 20] is perhaps the most elegant and logical extension of quasi-Newton methods, but it involves solving a sub-iteration." [Becker and Fadili, 2012].
  - "PQN is an implementation that uses a limited-memory quasi-Newton update and has both excellent empirical performance and theoretical properties." [Lee et al., 2012].

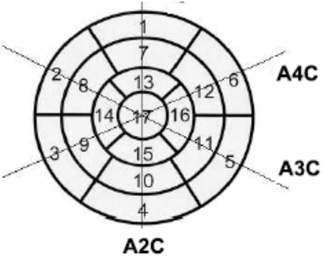
Inexact Proximal Newton

- The proximal quasi-Newton (PQN) approach:
  - "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].
  - Proximal-Newton methods are becoming optimization workhorse, e.g. NIPS 2012:
    - Becker & Fadili, Hsieh et al., Lee et al., Olsen et al., Pacheco & Sudderth.
  - `http://www.di.ens.fr/~mschmidt/Software/PQN.html`

Motivation: Structure Learning in CRFs

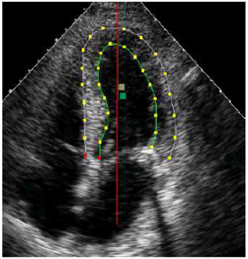
- Task: early detection of coronoary heart disease.

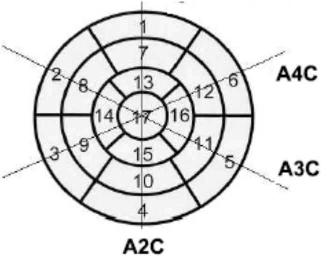




Motivation: Structure Learning in CRFs

- Task: early detection of coronoary heart disease.

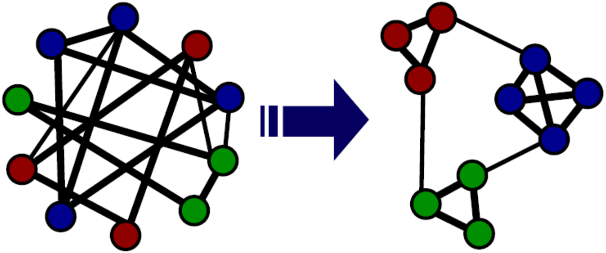




- Assess motion of heart segments using structured prediction.
- Data-fitting function is dynamic program.

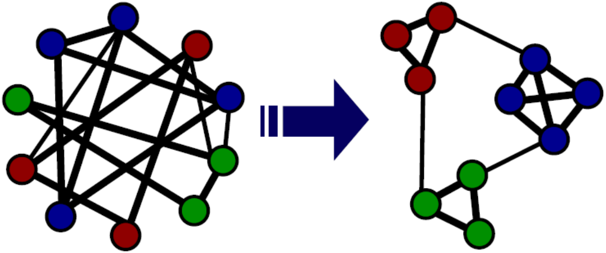
Example: Learning Variable Groupings

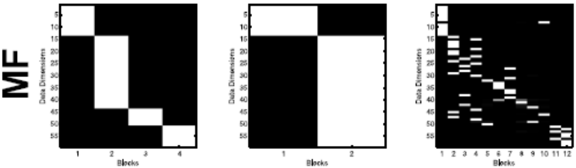
Discovering variable groupings:



Example: Learning Variable Groupings

Discovering variable groupings:





Known GL12 GL1

Example: Modeling Interventional Data

Conditioning by observation vs. conditioning by intervention:

Example: Modeling Interventional Data

Conditioning by observation vs. conditioning by intervention:

- If I see that my watch says 11:55, then it's almost lunch time

Example: Modeling Interventional Data

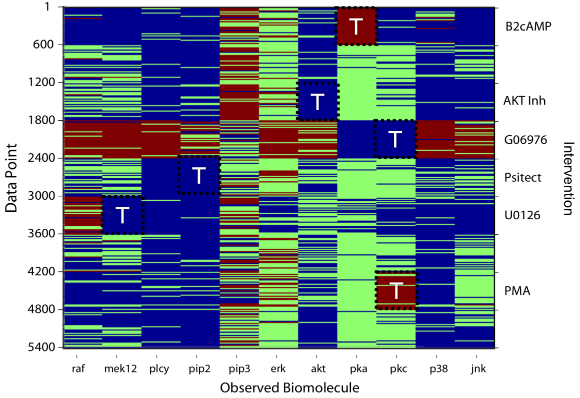
Conditioning by observation vs. conditioning by intervention:

- If I see that my watch says 11:55, then it's almost lunch time
- If I set my watch so it says 11:55, it doesn't help

Example: Modeling Interventional Data

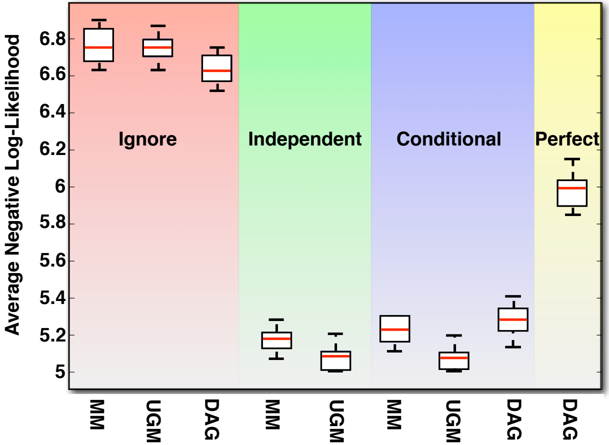
Conditioning by observation vs. conditioning by intervention:

- If I see that my watch says 11:55, then it's almost lunch time
- If I set my watch so it says 11:55, it doesn't help



Example: Modeling Interventional Data

Using structured prediction to model interventions:



Outline

1. Structured sparsity (inexact proximal-gradient method)
2. Learning dependencies (costly models with simple constraints)
3. Fitting a huge dataset (stochastic average gradient)

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 least-squares,

$$f_i(x) := (a_i^T x - b_i)^2.$$

- Other examples:
  - 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]:

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

  - Only requires $O(\log(1/\epsilon))$ iterations.
  - 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]:

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

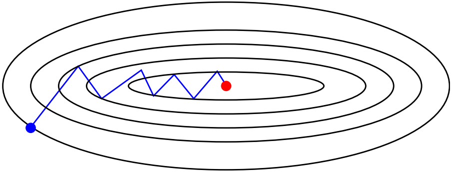
  - Only requires $O(\log(1/\epsilon))$ iterations.
  - Iteration cost is linear in N.
  - Quasi-Newton methods still require $O(N)$.
- Stochastic gradient method [Robbins & Monro, 1951]:
  - Random selection of $i(t)$ from $\{1, 2, \ldots, N\}$.

$$x_{t+1} = x_t - \alpha_t f'_{i(t)}(x_t).$$

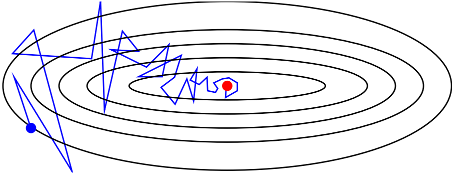
  - Iteration cost is independent of N.
  - Requires $O(1/\epsilon)$ iterations.

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]:

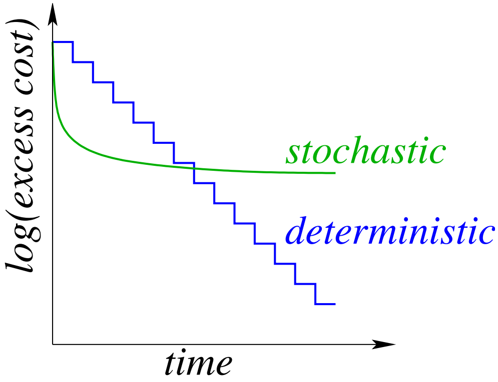


- Stochastic gradient method [Robbins & Monro, 1951]:



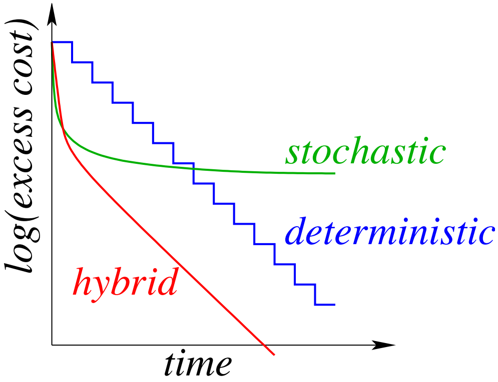
Motivation for New Methods

- DG method requires $O(\log(1/\epsilon))$ with $O(N)$.
- SG method requires $O(1/\epsilon)$ iterations with $O(1)$.



Motivation for New Methods

- DG method requires $O(\log(1/\epsilon))$ with $O(N)$.
- SG method requires $O(1/\epsilon)$ iterations with $O(1)$.



- Goal is requiring $O(\log(1/\epsilon))$ iterations with $O(1)$ cost.

Prior Work on Speeding up SG Methods

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

- **Step-size strategies, momentum, gradient/iterate averaging**
  - Polyak & Juditsky (1992), Tseng (1998), Kushner & Yin (2003) Nesterov (2009), Xiao (2010), Hazan & Kale (2011), Rakhlin et al. (2012)
- **Stochastic versions of accelerated and Newton 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:

- **Step-size strategies, momentum, gradient/iterate averaging**
  - Polyak & Juditsky (1992), Tseng (1998), Kushner & Yin (2003) Nesterov (2009), Xiao (2010), Hazan & Kale (2011), Rakhlin et al. (2012)
- **Stochastic versions of accelerated and Newton 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 $O(1/\epsilon)$ rate

Prior Work on Speeding up SG Methods

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

- **Step-size strategies, momentum, gradient/iterate averaging**
  - Polyak & Juditsky (1992), Tseng (1998), Kushner & Yin (2003) Nesterov (2009), Xiao (2010), Hazan & Kale (2011), Rakhlin et al. (2012)
- **Stochastic versions of accelerated and Newton 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 $O(1/\epsilon)$ rate
- **Constant step-size SG, accelerated SG**
  - Kesten (1958), Delyon and Juditsky (1993), Nedic and Bertsekas (2000)
  - $O(\log(1/\epsilon)$ iterations to reach a fixed tolerance

Prior Work on Speeding up SG Methods

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

- **Step-size strategies, momentum, gradient/iterate averaging**
  - Polyak & Juditsky (1992), Tseng (1998), Kushner & Yin (2003) Nesterov (2009), Xiao (2010), Hazan & Kale (2011), Rakhlin et al. (2012)
- **Stochastic versions of accelerated and Newton 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 $O(1/\epsilon)$ rate
- **Constant step-size SG, accelerated SG**
  - Kesten (1958), Delyon and Juditsky (1993), Nedic and Bertsekas (2000)
  - $O(\log(1/\epsilon)$ iterations to reach a fixed tolerance
- **Hybrid methods, incremental average gradient**
  - Bertsekas (1997), Blatt et al. (2007), Friedlander and Schmidt (2012)
  - $O(\log(1/\epsilon))$ iterations but eventually requires full passes.

Stochastic Average Gradient

- **Can we have $O(1)$ cost but only require $O(\log(1/\epsilon))$ iterations?**

Stochastic Average Gradient

- **Can we have** $O(1)$ **cost but only require** $O(\log(1/\epsilon))$ **iterations?**
  - YES!

Stochastic Average Gradient

- **Can we have $O(1)$ cost but only require $O(\log(1/\epsilon))$ iterations?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select $i(t)$ from $\{1, 2, \dots, n\}$ and compute $f'_{i(t)}(x^t)$.

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

Stochastic Average Gradient

- **Can we have** $O(1)$ **cost but only require** $O(\log(1/\epsilon))$ **iterations?**
  - 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

- **Can we have $O(1)$ cost but only require $O(\log(1/\epsilon))$ iterations?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select $i(t)$ from $\{1, 2, \ldots, n\}$ and compute $f'_{i(t)}(x^t)$.

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} y_i^t$$

    - **Memory**: $y_i^t = f'_i(x^t)$ from the last t where i was selected.

Stochastic Average Gradient

- **Can we have $O(1)$ cost but only require $O(\log(1/\epsilon))$ iterations?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select $i(t)$ from $\{1, 2, \ldots, n\}$ and compute $f'_{i(t)}(x^t)$.

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} y_i^t$$

    - **Memory**: $y_i^t = f_i'(x^t)$ from the last t where i was selected.
  - Stochastic variant of increment average gradient (IAG).

    [Blatt et al., 2007]

Stochastic Average Gradient

- **Can we have $O(1)$ cost but only require $O(\log(1/\epsilon))$ iterations?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select $i(t)$ from $\{1, 2, \ldots, n\}$ and compute $f'_{i(t)}(x^t)$.

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} y_i^t$$

    - **Memory**: $y_i^t = f'_i(x^t)$ from the last t where i was selected.
  - Stochastic variant of increment average gradient (IAG).

    [Blatt et al., 2007]
  - Assumes gradients of non-selected examples don't change.
  - Assumption becomes accurate as $||x^{t+1} - x^t|| \to 0$.
  - Memory requirements reduced to $O(N)$ for many problems.

Convergence Rate of SAG

Theorem *[Schmidt et al., 2013] The expected number of SAG iterations to reach an accuracy of* ϵ *is* $O(\max\{\kappa, N\} \log(1/\epsilon))$.

Convergence Rate of SAG

Theorem *[Schmidt et al., 2013] The expected number of SAG iterations to reach an accuracy of* ϵ *is* $O(\max\{\kappa, N\}\log(1/\epsilon))$.

- Proof is ‘infamous’, but the constants are good.

Convergence Rate of SAG

Theorem *[Schmidt et al., 2013] The expected number of SAG iterations to reach an accuracy of ϵ is $O(\max\{\kappa, N\} \log(1/\epsilon))$.*

- Proof is 'infamous', but the constants are good.
- Number of f_i' evaluations to reach ϵ:

Convergence Rate of SAG

Theorem *[Schmidt et al., 2013] The expected number of SAG iterations to reach an accuracy of ϵ is $O(\max\{\kappa, N\} \log(1/\epsilon))$.*

- Proof is 'infamous', but the constants are good.
- Number of f_i' evaluations to reach ϵ:
  - Stochastic: $O(\kappa(1/\epsilon))$.

Convergence Rate of SAG

Theorem *[Schmidt et al., 2013] The expected number of SAG iterations to reach an accuracy of ϵ is* $O(\max\{\kappa, N\} \log(1/\epsilon))$.

- Proof is 'infamous', but the constants are good.
- Number of f_i' evaluations to reach ϵ:
  - Stochastic: $O(\kappa(1/\epsilon))$.
  - Gradient: $O(N\kappa \log(1/\epsilon))$.

Convergence Rate of SAG

Theorem *[Schmidt et al., 2013] The expected number of SAG iterations to reach an accuracy of ϵ is $O(\max\{\kappa, N\}\log(1/\epsilon))$.*

- Proof is 'infamous', but the constants are good.
- Number of f_i' evaluations to reach ϵ:
  - Stochastic: $O(\kappa(1/\epsilon))$.
  - Gradient: $O(N\kappa\log(1/\epsilon))$.
  - Accelerated: $O(N\sqrt{\kappa}\log(1/\epsilon))$.

Convergence Rate of SAG

Theorem *[Schmidt et al., 2013] The expected number of SAG iterations to reach an accuracy of ϵ is $O(\max\{\kappa, N\} \log(1/\epsilon))$.*

- Proof is 'infamous', but the constants are good.
- Number of f_i' evaluations to reach ϵ:
  - Stochastic: $O(\kappa(1/\epsilon))$.
  - Gradient: $O(N\kappa \log(1/\epsilon))$.
  - Accelerated: $O(N\sqrt{\kappa} \log(1/\epsilon))$.
  - SAG: $O(\max\{N, \kappa\} \log(1/\epsilon))$.

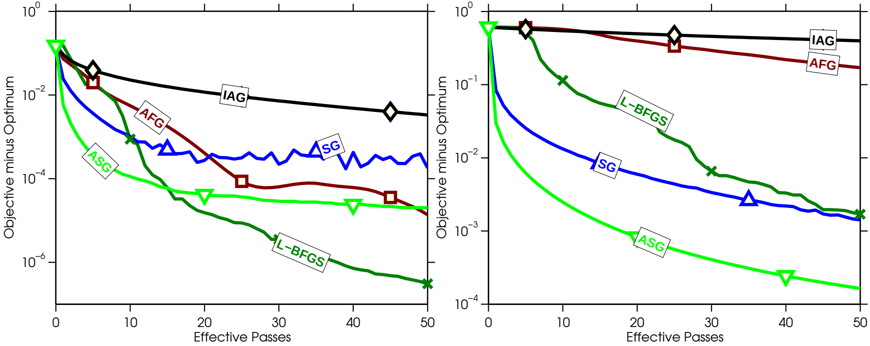
Convergence Rate of SAG

Theorem *[Schmidt et al., 2013] The expected number of SAG iterations to reach an accuracy of ϵ is $O(\max\{\kappa, N\}\log(1/\epsilon))$.*

- Proof is 'infamous', but the constants are good.
- Number of f_i' evaluations to reach ϵ:
  - Stochastic: $O(\kappa(1/\epsilon))$.
  - Gradient: $O(N\kappa\log(1/\epsilon))$.
  - Accelerated: $O(N\sqrt{\kappa}\log(1/\epsilon))$.
  - SAG: $O(\max\{N, \kappa\}\log(1/\epsilon))$.
- SAG beats two lower bounds:
  - Stochastic gradient bound of $O(1/\epsilon)$.
  - Deterministic gradient bound of $O(N\sqrt{\kappa}\log(1/\epsilon))$ (large N and κ).

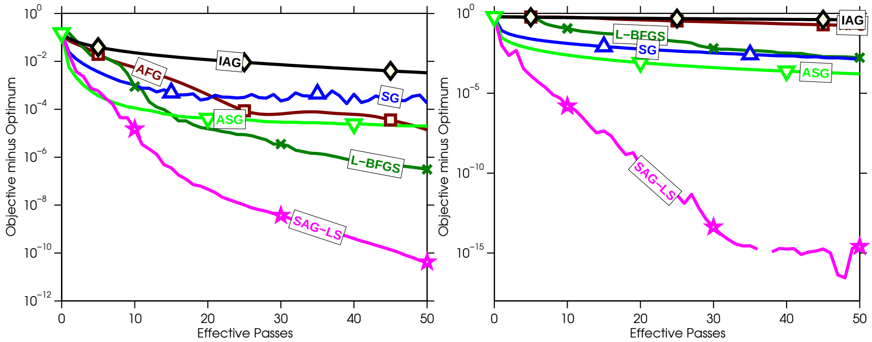
Comparing FG and SG Methods

- 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 ($n = 697641$, $p = 47236$)



Discussion

- Faster theoretical convergence using only the ‘sum’ structure.

Discussion

- Faster theoretical convergence using only the 'sum' structure.
- Simple algorithm, empirically better than theory predicts.

Discussion

- Faster theoretical convergence using only the 'sum' structure.
- Simple algorithm, empirically better than theory predicts.
- Robust stochastic gradient algorithm:
  - Adaptive step-size, termination criterion.

Discussion

- Faster theoretical convergence using only the 'sum' structure.
- Simple algorithm, empirically better than theory predicts.
- Robust stochastic gradient algorithm:
  - Adaptive step-size, termination criterion.
- Various extensions:
  - Non-uniform sampling.

    [Schmidt et al., 2013]
  - Non-smooth problems.

    [Mairal, 2013, Wong et al., 2013, Mairal, 2014, Xiao and Zhang, 2014, Defazio et al., 2014]
  - Memory-free methods.

    [Mahdavi et al., 2013, Johnson and Zhang, 2013, Zhang et al., 2013, Konecny and Richtarik, 2013, Xiao and Zhang, 2014]
  - Quasi-Newton methods.

    [Sohl-Dickstein et al., 2014]