The Gradient Method: Past and Present

Amir Beck

School of Mathematical Sciences, Tel Aviv University

Workshop in honor of Uri Rothblum,
Tel Aviv University, January 22, 2018

The Gradient Method

The problem.

$$\min\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^n\}$$

f differentiable.

The Gradient Method

The problem.

$$\min\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^n\}$$

f differentiable.

The Gradient Method

$$\mathbf{x}^{k+1} = \mathbf{x}^k - t_k \nabla f(\mathbf{x}^k)$$

$t_k > 0$ - chosen stepsize.

The Gradient Method

The problem.

$$\min\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^n\}$$

f differentiable.

The Gradient Method

$$\mathbf{x}^{k+1} = \mathbf{x}^k - t_k \nabla f(\mathbf{x}^k)$$

$t_k > 0$ - chosen stepsize.

- What is the starting point?
- What stepsize should be taken?
- What is the stopping criteria?

Stepsize Selection Rules

- **constant stepsize** - $t_k = \bar{t}$ for any k.

[1] also referred to as Armijo

Stepsize Selection Rules

- **constant stepsize** - $t_k = \bar{t}$ for any k.
- **exact stepsize** - t_k is a minimizer of f along the ray $\mathbf{x}_k - t\nabla f(\mathbf{x}^k)$:

$$t_k \in \operatorname*{argmin}_{t \geq 0} f(\mathbf{x}^k - t\nabla f(\mathbf{x}^k)).$$

[1] also referred to as Armijo

Stepsize Selection Rules

- **constant stepsize** - $t_k = \bar{t}$ for any k.
- **exact stepsize** - t_k is a minimizer of f along the ray $\mathbf{x}_k - t\nabla f(\mathbf{x}^k)$:

$$t_k \in \underset{t \geq 0}{\operatorname{argmin}} f(\mathbf{x}^k - t\nabla f(\mathbf{x}^k)).$$

- **backtracking**[1] - The method requires three parameters: $s > 0, \alpha \in (0,1), \beta \in (0,1)$. Here we start with an initial stepsize $t_k = s$. While

$$f(\mathbf{x}^k) - f(\mathbf{x}^k - t_k \nabla f(\mathbf{x}^k)) < \alpha t_k \|\nabla f(\mathbf{x}^k)\|^2.$$

  set $t_k := \beta t_k$

Sufficient Decrease Property:

$$f(\mathbf{x}^k) - f(\mathbf{x}^k - t_k \nabla f(\mathbf{x}^k)) \geq \alpha t_k \|\nabla f(\mathbf{x}^k)\|^2.$$

[1] also referred to as Armijo

Gradient Method as Steepest Descent

- $-\nabla f(\mathbf{x}^k)$ is a descent direction:

$$f'(\mathbf{x}^k; -\nabla f(\mathbf{x}^k)) = -\nabla f(\mathbf{x}^k)^T \nabla f(\mathbf{x}^k) = -\|\nabla f(\mathbf{x}^k)\|^2 < 0.$$

- In addition for being a descent direction, minus the gradient is also the steepest descent direction method.

Gradient Method as Steepest Descent

- $-\nabla f(\mathbf{x}^k)$ is a descent direction:

$$f'(\mathbf{x}^k; -\nabla f(\mathbf{x}^k)) = -\nabla f(\mathbf{x}^k)^T \nabla f(\mathbf{x}^k) = -\|\nabla f(\mathbf{x}^k)\|^2 < 0.$$

- In addition for being a descent direction, minus the gradient is also the steepest descent direction method.

Lemma. Let f be a differentiable function and let $\mathbf{x} \in \mathbb{R}^n$ satisfy $\nabla f(\mathbf{x}) \neq \mathbf{0}$. Then an optimal solution of

$$\min_{\mathbf{d}} \{f'(\mathbf{x}; \mathbf{d}) : \|\mathbf{d}\| = 1\}$$

is $\mathbf{d} = -\nabla f(\mathbf{x})/\|\nabla f(\mathbf{x})\|$.

Convergence(?) of the Gradient Method

$$\min\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^n\}$$

Standard conditions:

- f is bounded below and differentiable.
- f is L-smooth meaning that

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\| \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n.$$

Convergence(?) of the Gradient Method

$$\min\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^n\}$$

Standard conditions:

- f is bounded below and differentiable.
- f is L-smooth meaning that

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\| \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n.$$

Can be proved:

- Descent method: $f(\mathbf{x}^{k+1}) < f(\mathbf{x}^k)$
- Accumulation pts. of the sequence generated by GM are stationary points $(\nabla f(\mathbf{x}^*) = 0)$ (constant stepsize $t_k \equiv \bar{t} \in \left(0, \frac{2}{L}\right)$, backtracking or exact minimization)
- If f is convex, convergence to a global optimal solution.

Gradient Method - the Oldest Continuous Optimization Method?

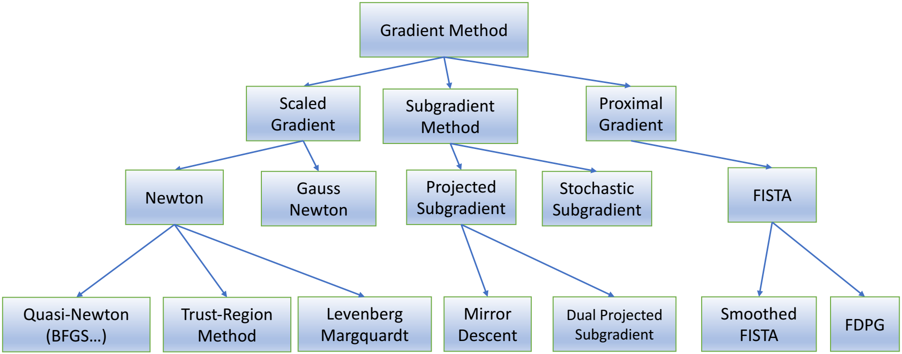
Méthode generales pour la résolution des systèmes d'equations simultanées, 1847

Augustin Louis Cauchy
1789-1857

- Suggested the method for solving sets of nonlinear equations

$$f_i(\mathbf{x}) = 0, i = 1, 2, \ldots, m \Rightarrow \min_{\mathbf{x}} \sum_{i=1}^{m} f_i(\mathbf{x})^2$$

- Not a particularly rigorous paper...
- Modern optimization starts only 100 years afterwards (simplex for LP)



Gradient Method
Scaled Gradient
Subgradient Method
Proximal Gradient
Newton
Gauss Newton
Projected Subgradient
Stochastic Subgradient
FISTA
Quasi-Newton (BFGS...)
Trust-Region Method
Levenberg Margquardt
Mirror Descent
Dual Projected Subgradient
Smoothed FISTA
FDPG

Gradient-Based Algorithms

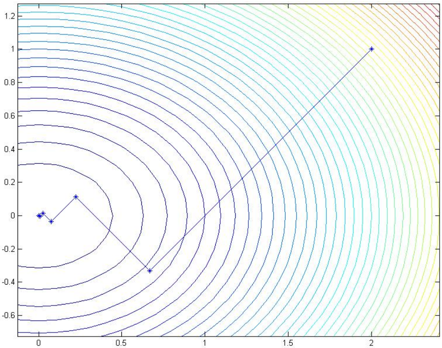
Widely used in applications....

- **Clustering Analysis:** *The k-means algorithm*
- **Neuro-computing:** *The backpropagation algorithm*
- **Statistical Estimation:** *The EM (Expectation-Maximization)* algorithm.
- **Machine Learning:** *SVM, Regularized regression, etc...*
- **Signal and Image Processing:** *Sparse Recovery, Denoising and Deblurring Schemes, Total Variation minimization...*
- **Matrix minimization Problems....and much more...**

The Zig-Zag Property

Zig-Zagging: directions produced by the gradient method with exact minimization are perpendicular.

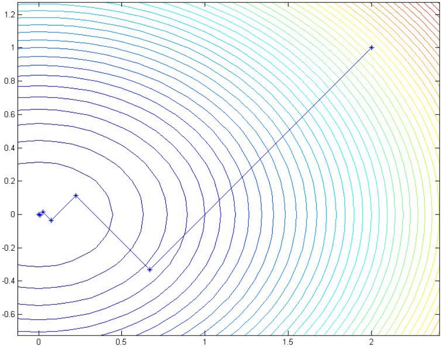
$$\langle \nabla f(\mathbf{x}^k), \nabla f(\mathbf{x}^{k+1}) \rangle = 0$$



The Zig-Zag Property

Zig-Zagging: directions produced by the gradient method with exact minimization are perpendicular.

$$\langle \nabla f(\mathbf{x}^k), \nabla f(\mathbf{x}^{k+1}) \rangle = 0$$

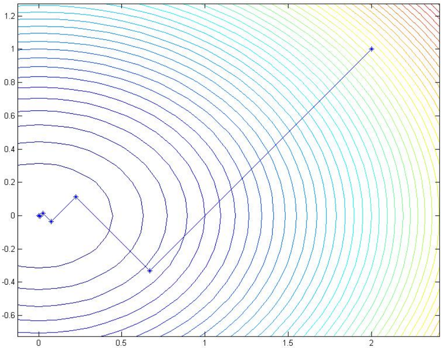


Main disadvantage: gradient method is rather slow.

The Zig-Zag Property

Zig-Zagging: directions produced by the gradient method with exact minimization are perpendicular.

$$\langle \nabla f(\mathbf{x}^k), \nabla f(\mathbf{x}^{k+1}) \rangle = 0$$



Main disadvantage: gradient method is rather slow.
Advantages: requires minimal information $(f, \nabla f)$, "cheap" iterative scheme, suitable for large-scale problems.

The Condition Number

- Rate of convergence of the gradient method depends on the condition number of the matrix $\nabla^2 f(\mathbf{x}^*)$:

$$\kappa(\nabla^2 f(\mathbf{x}^*)) = \frac{\sigma_{\max}(\nabla^2 f(\mathbf{x}^*))}{\sigma_{\min}(\nabla^2 f(\mathbf{x}^*))}$$

- Ill-conditioned problems - high condition number
- Well-conditioned problems - small condition number

A Severely Ill-Condition Function - Rosenbrock

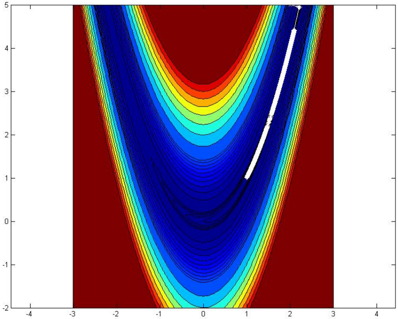
$$\min \left\{ f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \right\}.$$

condition number: 2508

A Severely Ill-Condition Function - Rosenbrock

$$\min \left\{ f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \right\}.$$

condition number: 2508



6890(!!!) iterations.

Improving the Gradient Method - Scaled Gradient

Scaled Gradient Method

$$\mathbf{x}^{k+1} = \mathbf{x}^k - t_k \mathbf{D}_k \nabla f(\mathbf{x}^k)$$

$t_k > 0$ - chosen stepsize. $\mathbf{D}_k \succ \mathbf{0}$

Improving the Gradient Method - Scaled Gradient

Scaled Gradient Method

$$\mathbf{x}^{k+1} = \mathbf{x}^k - t_k \mathbf{D}_k \nabla f(\mathbf{x}^k)$$

$t_k > 0$ - chosen stepsize. $\mathbf{D}_k \succ \mathbf{0}$

- Since $\mathbf{D}_k \succ \mathbf{0}$ - still a descent directions method.

Improving the Gradient Method - Scaled Gradient

Scaled Gradient Method

$$\mathbf{x}^{k+1} = \mathbf{x}^k - t_k \mathbf{D}_k \nabla f(\mathbf{x}^k)$$

$t_k > 0$ - chosen stepsize. $\mathbf{D}_k \succ \mathbf{0}$

- Since $\mathbf{D}_k \succ \mathbf{0}$ - still a descent directions method.
- Same as the gradient method employed after the change of variables $\mathbf{x} = \mathbf{D}_k^{1/2}\mathbf{y}$

Improving the Gradient Method - Scaled Gradient

Scaled Gradient Method

$$\mathbf{x}^{k+1} = \mathbf{x}^k - t_k \mathbf{D}_k \nabla f(\mathbf{x}^k)$$

$t_k > 0$ - chosen stepsize. $\mathbf{D}_k \succ \mathbf{0}$

- Since $\mathbf{D}_k \succ \mathbf{0}$ - still a descent directions method.
- Same as the gradient method employed after the change of variables $\mathbf{x} = \mathbf{D}_k^{1/2}\mathbf{y}$
- Convergence is related to the condition number of $\mathbf{D}_k^{-1/2}\nabla^2 f(\mathbf{x}^k)\mathbf{D}_k^{-1/2}$

Improving the Gradient Method - Scaled Gradient

Scaled Gradient Method

$$\mathbf{x}^{k+1} = \mathbf{x}^k - t_k \mathbf{D}_k \nabla f(\mathbf{x}^k)$$

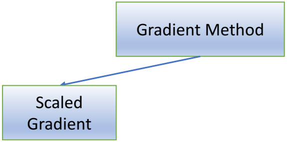
$t_k > 0$ - chosen stepsize. $\mathbf{D}_k \succ \mathbf{0}$

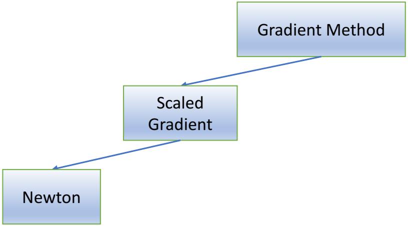
- Since $\mathbf{D}_k \succ \mathbf{0}$ - still a descent directions method.
- Same as the gradient method employed after the change of variables $\mathbf{x} = \mathbf{D}_k^{1/2}\mathbf{y}$
- Convergence is related to the condition number of $\mathbf{D}_k^{-1/2}\nabla^2 f(\mathbf{x}^k)\mathbf{D}_k^{-1/2}$
- "best" choice $\mathbf{D}_k = \nabla^2 f(\mathbf{x}^k)^{-1}$. pure Newton's method:

$$\boxed{\mathbf{x}^{k+1} = \mathbf{x}^k - \nabla^2 f(\mathbf{x}^k)^{-1}\nabla f(\mathbf{x}^k)}$$

- Popular and "cheap" choice: $\mathbf{D}_k$ diagonal (diagonal scaling)

Gradient Method


Gradient Method
Scaled
Gradient


Gradient Method
Scaled Gradient
Newton

The Gauss-Newton Method

Nonlinear least squares:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \sum_{i=1}^{m} (f_i(\mathbf{x}) - c_i)^2$$

$f_1, f_2, \ldots, f_m$ - differentiable.

The Gauss-Newton Method

Nonlinear least squares:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \sum_{i=1}^{m} (f_i(\mathbf{x}) - c_i)^2$$

$f_1, f_2, \ldots, f_m$ - differentiable.
Given the kth iterate $\mathbf{x}^k$, the next iterate is chosen to minimize the sum of squares of the linearized terms, that is,

$$\mathbf{x}^{k+1} = \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^n} \left\{ \sum_{i=1}^{m} \left[f_i(\mathbf{x}^k) + \nabla f_i(\mathbf{x}^k)^T(\mathbf{x}-\mathbf{x}^k) - c_i \right]^2 \right\}.$$

The Gauss-Newton Method

Nonlinear least squares:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \sum_{i=1}^{m} (f_i(\mathbf{x}) - c_i)^2$$

$f_1, f_2, \ldots, f_m$ - differentiable.
Given the kth iterate $\mathbf{x}^k$, the next iterate is chosen to minimize the sum of squares of the linearized terms, that is,

$$\mathbf{x}^{k+1} = \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^n} \left\{ \sum_{i=1}^{m} \left[f_i(\mathbf{x}^k) + \nabla f_i(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) - c_i \right]^2 \right\}.$$

- The general step requires to solve a linear least squares problem at each iteration.

The Gauss-Newton Method

Nonlinear least squares:

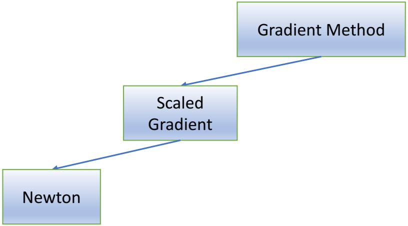
$$\min_{\mathbf{x}\in\mathbb{R}^n} \sum_{i=1}^{m} (f_i(\mathbf{x}) - c_i)^2$$

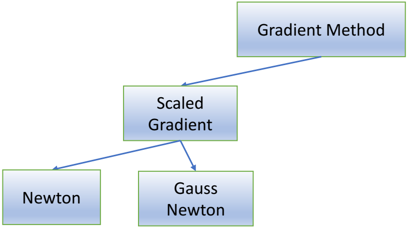
$f_1, f_2, \ldots, f_m$ - differentiable.

Given the kth iterate $\mathbf{x}^k$, the next iterate is chosen to minimize the sum of squares of the linearized terms, that is,

$$\mathbf{x}^{k+1} = \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^n} \left\{ \sum_{i=1}^{m} \left[f_i(\mathbf{x}^k) + \nabla f_i(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) - c_i \right]^2 \right\}.$$

- The general step requires to solve a linear least squares problem at each iteration.
- Actually a scaled gradient method with $\mathbf{D}_k = (\mathbf{J}(\mathbf{x}^k)^T \mathbf{J}(\mathbf{x}^k))^{-1}$ ($\mathbf{J}(\cdot)$ - Jacobian)


Gradient Method
Scaled Gradient
Newton


Gradient Method
Scaled Gradient
Newton
Gauss Newton

Problems with Newton's Method

$$\boxed{\mathbf{x}^{k+1} = \mathbf{x}^k - \nabla^2 f(\mathbf{x}^k)^{-1}\nabla f(\mathbf{x}^k)}$$

- $\nabla^2 f(\mathbf{x}^k)$ difficult to compute and/or problematic to solve the system $\nabla^2 f(\mathbf{x}^k)\mathbf{z} = \nabla f(\mathbf{x}^k)$

Problems with Newton's Method

$$\boxed{\mathbf{x}^{k+1} = \mathbf{x}^k - \nabla^2 f(\mathbf{x}^k)^{-1} \nabla f(\mathbf{x}^k)}$$

- $\nabla^2 f(\mathbf{x}^k)$ difficult to compute and/or problematic to solve the system $\nabla^2 f(\mathbf{x}^k)\mathbf{z} = \nabla f(\mathbf{x}^k)$
- $\nabla^2 f(\mathbf{x}^k)$ might be singular
- $\nabla^2 f(\mathbf{x}^k)$ might not be positive definite (Newton's direction not a descent direction...)

Problems with Newton's Method

$$\boxed{\mathbf{x}^{k+1} = \mathbf{x}^k - \nabla^2 f(\mathbf{x}^k)^{-1} \nabla f(\mathbf{x}^k)}$$

- $\nabla^2 f(\mathbf{x}^k)$ difficult to compute and/or problematic to solve the system $\nabla^2 f(\mathbf{x}^k)\mathbf{z} = \nabla f(\mathbf{x}^k)$
- $\nabla^2 f(\mathbf{x}^k)$ might be singular
- $\nabla^2 f(\mathbf{x}^k)$ might not be positive definite (Newton's direction not a descent direction...)
- Convergence extremely problematic: requires a lot of assumptions that are usually not satisfied.

Problems with Newton's Method

$$\boxed{\mathbf{x}^{k+1} = \mathbf{x}^k - \nabla^2 f(\mathbf{x}^k)^{-1} \nabla f(\mathbf{x}^k)}$$

- $\nabla^2 f(\mathbf{x}^k)$ difficult to compute and/or problematic to solve the system $\nabla^2 f(\mathbf{x}^k)\mathbf{z} = \nabla f(\mathbf{x}^k)$
- $\nabla^2 f(\mathbf{x}^k)$ might be singular
- $\nabla^2 f(\mathbf{x}^k)$ might not be positive definite (Newton's direction not a descent direction...)
- Convergence extremely problematic: requires a lot of assumptions that are usually not satisfied.
- **main advantage:** quadratic rate of convergence (under very restrictive conditions...)

Problems with Newton's Method

$$\boxed{\mathbf{x}^{k+1} = \mathbf{x}^k - \nabla^2 f(\mathbf{x}^k)^{-1}\nabla f(\mathbf{x}^k)}$$

- $\nabla^2 f(\mathbf{x}^k)$ difficult to compute and/or problematic to solve the system $\nabla^2 f(\mathbf{x}^k)\mathbf{z} = \nabla f(\mathbf{x}^k)$
- $\nabla^2 f(\mathbf{x}^k)$ might be singular
- $\nabla^2 f(\mathbf{x}^k)$ might not be positive definite (Newton's direction not a descent direction...)
- Convergence extremely problematic: requires a lot of assumptions that are usually not satisfied.
- **main advantage:** quadratic rate of convergence (under very restrictive conditions...)

Btw, pure Newton's is a utopian method. Better to incorporate a stepsize (damped Newton).

Classics from the 70's - Trying to Mend Newton

- Trust-Region Methods

$$\mathbf{x}^{k+1} \in \operatorname{argmin}\left\{m(\mathbf{x};\mathbf{x}^k) : \|\mathbf{x}-\mathbf{x}^k\| \leq \Delta_k\right\}$$

where $m(\mathbf{x};\mathbf{x}^k)$ is a model of f around $\mathbf{x}^k$, e.g.,
$m(\mathbf{x};\mathbf{x}^k) \equiv f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x}-\mathbf{x}^k) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^k)^T\nabla^2 f(\mathbf{x}^k)(\mathbf{x}-\mathbf{x}^k)$

Classics from the 70's - Trying to Mend Newton

- Trust-Region Methods

$$\mathbf{x}^{k+1} \in \operatorname{argmin}\left\{m(\mathbf{x};\mathbf{x}^k) : \|\mathbf{x}-\mathbf{x}^k\| \leq \Delta_k\right\}$$

where $m(\mathbf{x};\mathbf{x}^k)$ is a model of f around $\mathbf{x}^k$, e.g.,

$$m(\mathbf{x};\mathbf{x}^k) \equiv f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x}-\mathbf{x}^k) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^k)^T\nabla^2 f(\mathbf{x}^k)(\mathbf{x}-\mathbf{x}^k)$$

- Quasi-Newton Try to mimic the Hessian without actually forming it.

e.g., BFGS

$$\boxed{\mathbf{x}^{k+1} = \mathbf{x}^k - t_k\mathbf{D}_k^{-1}\nabla f(\mathbf{x}^k)}$$

$\mathbf{D}_k$ is chosen to satisfy the QN condition

$$\mathbf{D}_k(\mathbf{x}^k - \mathbf{x}^{k-1}) = \nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^{k-1})$$

Classics from the 70's - Trying to Mend Newton

- Trust-Region Methods

$$\mathbf{x}^{k+1} \in \operatorname{argmin}\left\{m(\mathbf{x};\mathbf{x}^k) : \|\mathbf{x}-\mathbf{x}^k\| \leq \Delta_k\right\}$$

where $m(\mathbf{x};\mathbf{x}^k)$ is a model of f around $\mathbf{x}^k$, e.g.,
$m(\mathbf{x};\mathbf{x}^k) \equiv f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x}-\mathbf{x}^k) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^k)^T\nabla^2 f(\mathbf{x}^k)(\mathbf{x}-\mathbf{x}^k)$

- Quasi-Newton Try to mimic the Hessian without actually forming it.

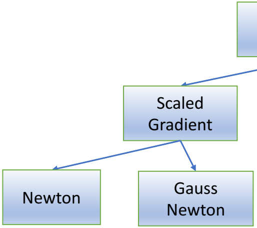
e.g., BFGS

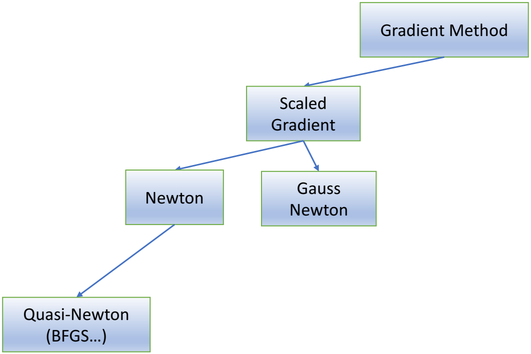
$$\boxed{\mathbf{x}^{k+1} = \mathbf{x}^k - t_k \mathbf{D}_k^{-1}\nabla f(\mathbf{x}^k)}$$

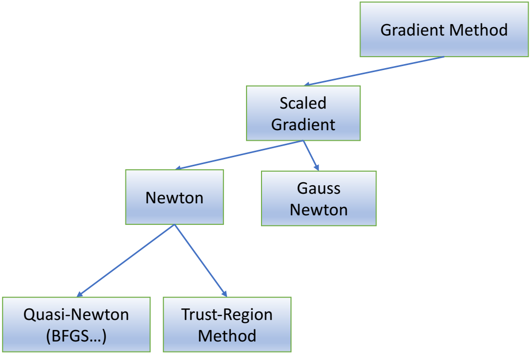
$\mathbf{D}_k$ is chosen to satisfy the QN condition

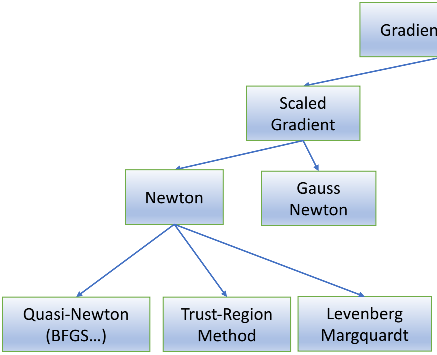
$$\mathbf{D}_k(\mathbf{x}^k - \mathbf{x}^{k-1}) = \nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^{k-1})$$

- $\mathbf{D}_{k+1}$ "simply" constructed from $\mathbf{D}_k$
- Computation of $\mathbf{D}_k^{-1}$ requires only $O(n^2)$ flops (linear algebra tricks)


Gradient Method
Scaled Gradient
Newton
Gauss Newton


Gradient Method
Scaled Gradient
Newton
Gauss Newton
Quasi-Newton (BFGS…)


Gradient Method
Scaled Gradient
Newton
Gauss Newton
Quasi-Newton (BFGS…)
Trust-Region Method


Gradient Method
Scaled Gradient
Newton
Gauss Newton
Quasi-Newton (BFGS…)
Trust-Region Method
Levenberg Margquardt

So far...

Classical algorithms for solving

The problem.

$$\min\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^n\}$$

f differentiable.

So far...

Classical algorithms for solving

The problem.

$$\min\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^n\}$$

f differentiable.

What happens if f is nonsmooth? e.g.,

$$f(\mathbf{x}) = \sum_{i=1}^{m} |\mathbf{a}_i^T \mathbf{x} - b_i|, \max_{i=1,\ldots,m} |\mathbf{a}_i^T \mathbf{x} - b_i| \ldots.$$

Wolfe's Example

False hope: What happens if the method never encounters non-differentiability points?

Wolfe's Example

False hope: What happens if the method never encounters non-differentiability points?

- Let $\gamma > 1$ and consider

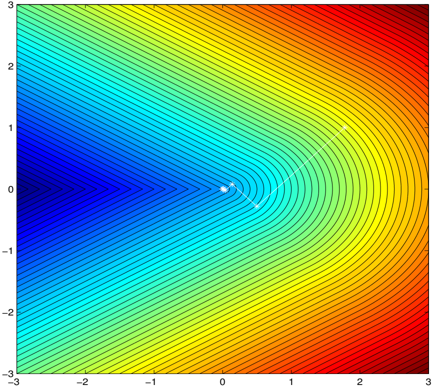
$$f(x_1, x_2) = \begin{cases} \sqrt{x_1^2 + \gamma x_2^2}, & |x_2| \le x_1, \\ \frac{x_1 + \gamma |x_2|}{\sqrt{1+\gamma}}, & \text{else.} \end{cases}$$

- f is differentiable at all points except for the ray $\{(x_1, 0) : x_1 \le 0\}$.

Wolfe's Example

The gradient method with exact line search converges to a non-optimal point.

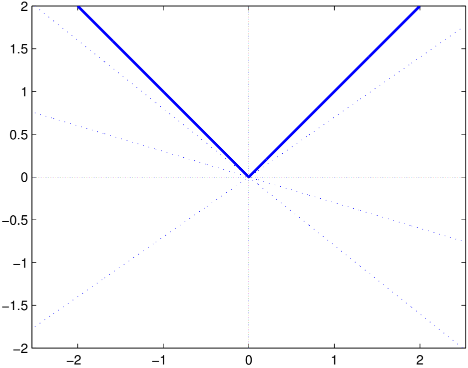
Conclusion: cannot ignore non-differentiability $\rightarrow$ extend the notion of the gradient



The Subgradient Method. Shor (63) Polyak (65)

$$\mathbf{x}^{k+1} = \mathbf{x}^k - t_k f'(\mathbf{x}^k)$$

Replace the gradient $\nabla f(\mathbf{x})$ by a subgradient $f'(\mathbf{x}) \in \partial f(\mathbf{x})$ (vectors that correspond to underestimators of the function)



Projected Subgradient Method

Model: f - convex. C - closed convex

$$\boxed{\min\{f(\mathbf{x}) : \mathbf{x} \in C\}}$$

Projected Subgradient Method

Model: f - convex. C - closed convex

$$\min\{f(\mathbf{x}) : \mathbf{x} \in C\}$$

Projected Subgradient Method (PSM): Shor (63), Polyak (65)

$$\mathbf{x}^k = P_C(\mathbf{x}^{k-1} - t_k f'(\mathbf{x}_{k-1})), \quad f'(\mathbf{x}_{k-1}) \in \partial f(x^{k-1})$$

$t_k > 0$ - stepsize, $P_C(\cdot)$ - orthogonal projection operator.

Orthogonal Projection Operator:

$P_C(\mathbf{x}) =$ closest point in C to $\mathbf{x} = \underset{\mathbf{y} \in C}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{x}\|$.

Projected Subgradient Method

Model: f - convex. C - closed convex

$$\boxed{\min\{f(\mathbf{x}) : \mathbf{x} \in C\}}$$

Projected Subgradient Method (PSM): Shor (63), Polyak (65)

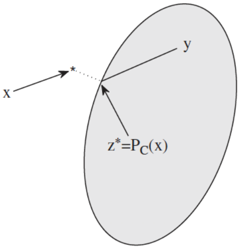
$$\mathbf{x}^k = P_C(\mathbf{x}^{k-1} - t_k f'(\mathbf{x}_{k-1})), \quad f'(\mathbf{x}_{k-1}) \in \partial f(x^{k-1})$$

$t_k > 0$ - stepsize, $P_C(\cdot)$ - orthogonal projection operator.

Orthogonal Projection Operator:

$P_C(\mathbf{x}) =$ closest point in C to $\mathbf{x} = \underset{\mathbf{y}\in C}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{x}\|$.

- SPM is **not a descent method**.
- $t_k \propto \frac{1}{\sqrt{k}} \Rightarrow f^k_{\text{best}} := \min_{1\leq s\leq k} f(\mathbf{x}_s) \to f_{\text{opt}}$



Rate of Convergence of SPM

A typical result: assume C convex compact. Take

$$t_k = \frac{\text{Diam}(C)}{\sqrt{k}}; \ \text{Diam}(C) := \max_{\mathbf{x},\mathbf{y} \in C} \|\mathbf{x} - \mathbf{y}\| < \infty,$$

$$\text{Then,} \qquad \min_{1 \le s \le k} f(\mathbf{x}_s) - f_* \le O(1) M \frac{\text{Diam}(C)}{\sqrt{k}}$$

- Thus, to find an approximate ε solution: $O(1/\varepsilon^2)$

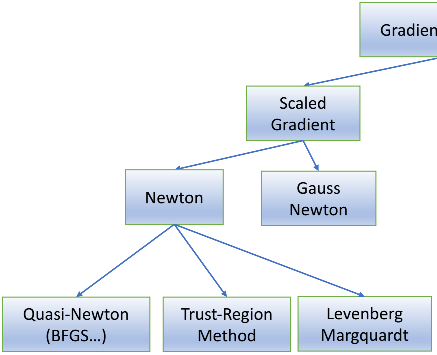
Rate of Convergence of SPM

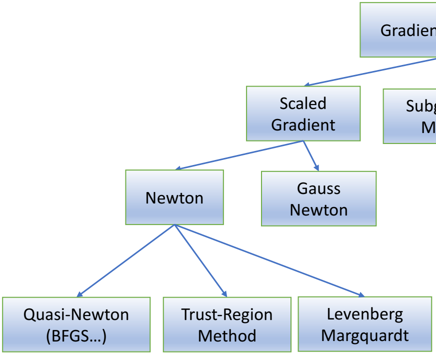
A typical result: assume C convex compact. Take

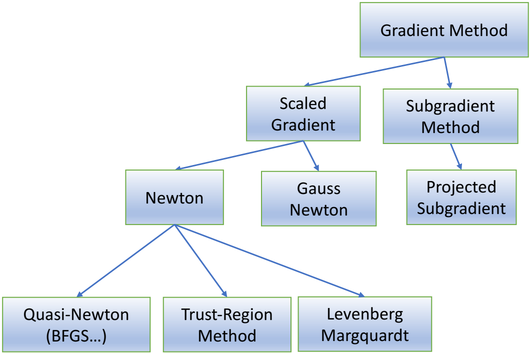
$$t_k = \frac{\mathrm{Diam}(C)}{\sqrt{k}}; \ \mathrm{Diam}(C) := \max_{\mathbf{x},\mathbf{y} \in C} \|\mathbf{x} - \mathbf{y}\| < \infty,$$

Then, $$\min_{1 \le s \le k} f(\mathbf{x}_s) - f_* \le O(1) M \frac{\mathrm{Diam}(C)}{\sqrt{k}}$$

- Thus, to find an approximate ε solution: $O(1/\varepsilon^2)$
- **Key Advantages:** rate nearly *independent* of problem's dimension. Simple, when projections are easy to compute...
- **Main Drawback of SPM:** too slow...needs $k \ge \varepsilon^{-2}$ iterations.
- Can we improve the situation of SPM? by exploiting the structure/geometry of the constraint set C.


Gradient Method
Scaled Gradient
Newton
Gauss Newton
Quasi-Newton (BFGS…)
Trust-Region Method
Levenberg Margquardt


Gradient Method
Scaled Gradient
Subgradient Method
Newton
Gauss Newton
Quasi-Newton (BFGS…)
Trust-Region Method
Levenberg Margquardt


Gradient Method
Scaled Gradient
Subgradient Method
Newton
Gauss Newton
Projected Subgradient
Quasi-Newton (BFGS…)
Trust-Region Method
Levenberg Margquardt

Mirror Descent: Non-Euclidean Version of SD

- Originated from functional analytic arguments in infinite dimensional setting between primal-dual spaces. Nemirovsky and Yudin (83)
- In (B.-Teboulle-2003) it was shown that the (MDA) can be simply viewed as a **Non-Euclidean projected subgradient method**.

The Idea

Another representation of the projected subgradient method:

$$\mathbf{x}^{k+1} = \underset{\mathbf{x} \in C}{\operatorname{argmin}} \left\{ f(\mathbf{x}^k) + \langle f'(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{1}{2t_k} \|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$$

Next iterate is a minimizer of the linear approximation regularized by a prox term.

The Idea

Another representation of the projected subgradient method:

$$\mathbf{x}^{k+1} = \underset{\mathbf{x}\in C}{\operatorname{argmin}} \left\{ f(\mathbf{x}^k) + \langle f'(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{1}{2t_k}\|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$$

Next iterate is a minimizer of the linear approximation regularized by a prox term.

The Idea: Replace the Euclidean distance by a non-Euclidean function:

$$\mathbf{x}^{k+1} = \underset{\mathbf{x}\in C}{\operatorname{argmin}} \left\{ f(\mathbf{x}) + \langle f'(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{1}{t_k} D(\mathbf{x}, \mathbf{x}^k) \right\}$$

The Idea

Another representation of the projected subgradient method:

$$\mathbf{x}^{k+1} = \underset{\mathbf{x}\in C}{\operatorname{argmin}} \left\{ f(\mathbf{x}^k) + \langle f'(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{1}{2t_k}\|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$$

Next iterate is a minimizer of the linear approximation regularized by a prox term.

The Idea: Replace the Euclidean distance by a non-Euclidean function:

$$\mathbf{x}^{k+1} = \underset{\mathbf{x}\in C}{\operatorname{argmin}} \left\{ f(\mathbf{x}) + \langle f'(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{1}{t_k} D(\mathbf{x}, \mathbf{x}^k) \right\}$$

What should we expect from $D(\cdot,\cdot)$?

- Take into account the structure of the constraints and “easy to compute”.
- ”distance-like”: $D(\mathbf{u},\mathbf{v}) \geq 0$ and equal zero iff $\mathbf{u} = \mathbf{v}$.
- Popular choice: Bregman distance $D(\mathbf{u},\mathbf{v}) = B_\omega(\mathbf{u},\mathbf{v}) = \omega(\mathbf{u}) - \omega(\mathbf{v}) - \nabla\omega(\mathbf{v})^T(\mathbf{u}-\mathbf{v})$ strongly convex w.r.t. to an arbitrary norm.

Demo - Trust Topology Design

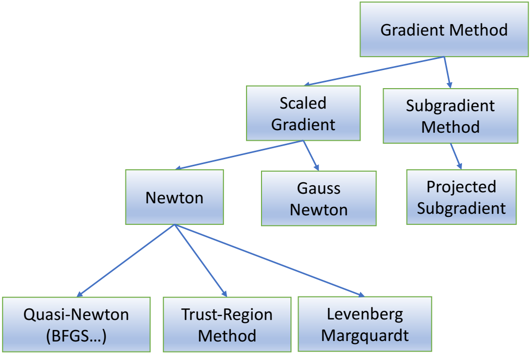
Design a truss of a given total weight capable to withstand a collection of forces acting on the nodes. Simplex constraints.

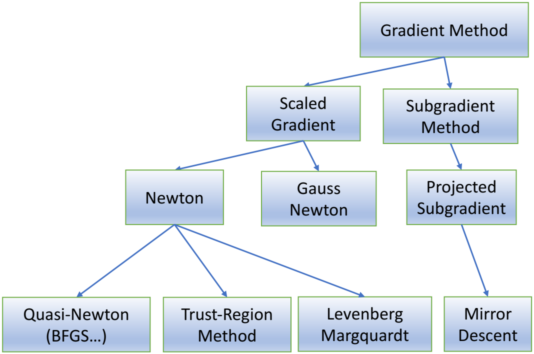
$$\min_{\mathbf{t}}\{\mathbf{f}^T A^{-1}(\mathbf{t})\mathbf{f} : \mathbf{t} \in \Delta_n\}$$

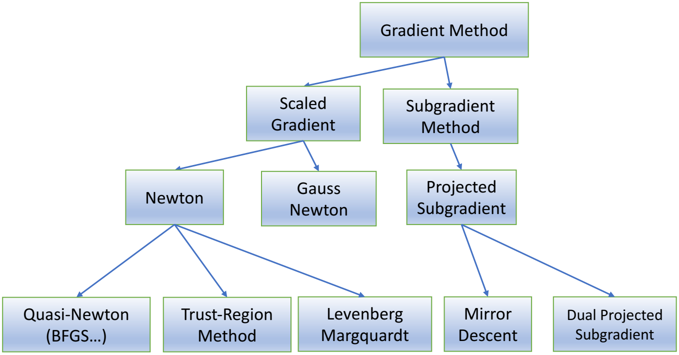
Comparing PSM with mirror descant $(\omega(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|_2^2, \sum_{i=1}^n x_i \log x_i)$

$$x_i^{k+1} = \frac{x_i^k e^{-t_k f_i'(\mathbf{x}^k)}}{\sum_{j=1}^n x_j^k e^{-t_k f_j'(\mathbf{x}^k)}}, \quad i = 1, 2, \ldots, n.$$

Theoretically the efficiency estimate is still of the order $O(1/\sqrt{k})$ but the constants can be improved by using non-Euclidean distances.


Gradient Method
Scaled Gradient
Subgradient Method
Newton
Gauss Newton
Projected Subgradient
Quasi-Newton (BFGS…)
Trust-Region Method
Levenberg Margquardt


Gradient Method
Scaled Gradient
Subgradient Method
Newton
Gauss Newton
Projected Subgradient
Quasi-Newton (BFGS…)
Trust-Region Method
Levenberg Margquardt
Mirror Descent


Gradient Method
Scaled Gradient
Subgradient Method
Newton
Gauss Newton
Projected Subgradient
Quasi-Newton (BFGS…)
Trust-Region Method
Levenberg Margquardt
Mirror Descent
Dual Projected Subgradient

Dual Projected Subgradient Method

Model:

$$\begin{aligned} f_{\rm opt} = \quad & \min \quad f(\mathbf{x}) \\ & \text{s.t.} \quad \mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \\ & \qquad\ \ \mathbf{x} \in X. \end{aligned}$$

Assumptions:

(A) $X \subseteq \mathbb{R}^n$ is convex.

(B) $f : \mathbb{R}^n \to \mathbb{R}$ is convex.

(C) $\mathbf{g}(\cdot) = (g_1(\cdot), g_2(\cdot), \ldots, g_m(\cdot))^T$, where $g_1, g_2, \ldots, g_m : \mathbb{R}^n \to \mathbb{R}$ are convex.

(D) For any $\boldsymbol{\lambda} \in \mathbb{R}^m_+$, the problem $\min_{\mathbf{x} \in X}\{f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x})\}$ attains an optimal solution.

The Lagrangian of the problem is given by

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x}).$$

The Dual Problem

$$\text{(D)} \quad q_{\rm opt} \equiv \max\{q(\boldsymbol{\lambda}) : \boldsymbol{\lambda} \in \mathbb{R}^m_+\},$$

where

$$q(\boldsymbol{\lambda}) = \min_{\mathbf{x}\in X} f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x}).$$

- Under the assumptions, strong duality holds, meaning that $f_{\rm opt} = q_{\rm opt}$ and the optimal solution of the dual problem is attained.

The Method

Main Observation: To compute a subgradient of $-q$ at $\boldsymbol{\lambda}$:

- Find $\mathbf{x}_\lambda \in \operatorname{argmin}_{\mathbf{x}\in X} L(\mathbf{x}, \boldsymbol{\lambda})$.
- $-\mathbf{g}(\mathbf{x}_\lambda) \in \partial(-q)(\boldsymbol{\lambda})$.

The Dual Projected Subgradient Method

Initialization: pick $\boldsymbol{\lambda}^0 \in \mathbb{R}^m_+$ arbitrarily.

General step: for any $k = 0, 1, 2, \ldots$,

(a) pick a positive number γ_k.

(b) compute $\mathbf{x}^k \in \operatorname{argmin}_{\mathbf{x}\in X} \left\{ f(\mathbf{x}) + (\boldsymbol{\lambda}^k)^T \mathbf{g}(\mathbf{x}) \right\}$.

(c) if $\mathbf{g}(\mathbf{x}^k) = \mathbf{0}$, then **terminate** with an output $\mathbf{x}^k$; otherwise,

$$\boldsymbol{\lambda}^{k+1} = \left[\boldsymbol{\lambda}^k + \gamma_k \frac{\mathbf{g}(\mathbf{x}^k)}{\|\mathbf{g}(\mathbf{x}^k)\|_2} \right]_+ .$$

$O(1/\sqrt{k})$ rate of convergence can be shown

$O(1/\varepsilon^2)$ Rate of Convergence in Nonsmooth Convex Optimization

- SPM,MD and dual projected subgradient are all $O(1/\varepsilon^2), O(1/\sqrt{k})$ methods. Can we do better?
- According to lower complexity bounds, the answer is No!
- However, by exploiting the structure of the functions, we can do better. For example, if assuming some smoothness properties...

Polynomial versus Gradient-Based Methods (80's and 90's)

- Rise of Polynomial methods for convex programming: ellipsoid, interior point methods.
- Convex problems are polynomially solvable within ε accuracy:

$$\text{Running Time} \leq \text{P}oly(\text{Problem's size}, \# \text{ of accuracy digits}).$$

- **Theoretically:** large scale problems can be solved to high accuracy with polynomial methods, such as IPM.
- **Practically:** Running time is **dimension-dependent** and grows **nonlinearly** with problem's dimension. For IPM which are Newton's type methods: $\sim O(n^3)$.
- **Thus, a "single iteration" of IPM can last forever!**
- 2000-... Gradient-based method have become popular again due to increasing size of applications.

Dealing with the Size - Decomposition

- One way to deal with the large or even huge-scale size of the new arising applications is use decomposition. For example...

Dealing with the Size - Decomposition

- One way to deal with the large or even huge-scale size of the new arising applications is use decomposition. For example...
- Consider the problem

$$\min_{\mathbf{x}} \left\{ f(\mathbf{x}) \equiv \sum_{i=1}^{m} f_i(\mathbf{x}) \right\}$$

where $f_1, f_2, \ldots, f_m$ are all convex functions. Suppose that m **is huge**.

Dealing with the Size - Decomposition

- One way to deal with the large or even huge-scale size of the new arising applications is use decomposition. For example...
- Consider the problem

$$\min_{\mathbf{x}} \left\{ f(\mathbf{x}) \equiv \sum_{i=1}^{m} f_i(\mathbf{x}) \right\}$$

  where $f_1, f_2, \ldots, f_m$ are all convex functions. Suppose that m **is huge**.
- The subgradient method is very expansive to execute:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - t_k \left(\sum_{i=1}^{m} f_i'(\mathbf{x}^k) \right).$$

Dealing with the Size - Decomposition

- One way to deal with the large or even huge-scale size of the new arising applications is use decomposition. For example...
- Consider the problem

$$\min_{\mathbf{x}} \left\{ f(\mathbf{x}) \equiv \sum_{i=1}^{m} f_i(\mathbf{x}) \right\}$$

where $f_1, f_2, \ldots, f_m$ are all convex functions. Suppose that m **is huge**.

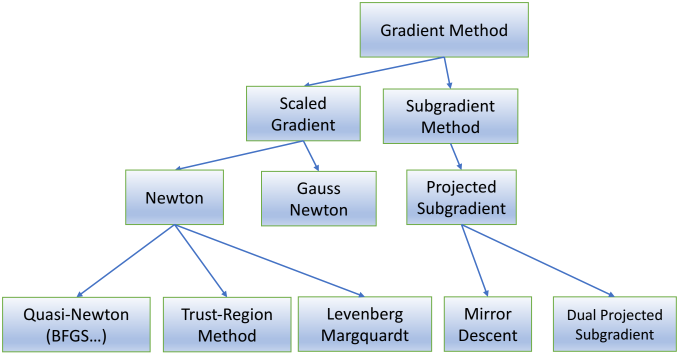
- The subgradient method is very expansive to execute:

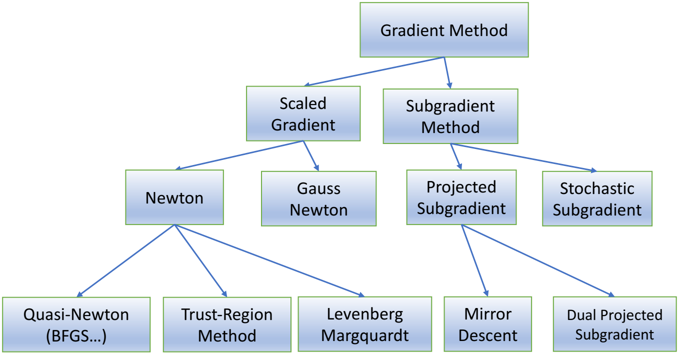
$$\mathbf{x}^{k+1} = \mathbf{x}^k - t_k \left(\sum_{i=1}^{m} f_i'(\mathbf{x}^k) \right).$$

- Instead, we can use the stochastic projected subgradient method that exploits only one randomlly chosen subgradient at each iteration (**decomposition**)

$$\mathbf{x}^{k+1} = \mathbf{x}^k - t_k f_{i_k}'(\mathbf{x}^k)$$

i_k - randomly chosen


Gradient Method
Scaled Gradient
Subgradient Method
Newton
Gauss Newton
Projected Subgradient
Quasi-Newton (BFGS...)
Trust-Region Method
Levenberg Margquardt
Mirror Descent
Dual Projected Subgradient


Gradient Method
Scaled Gradient
Subgradient Method
Newton
Gauss Newton
Projected Subgradient
Stochastic Subgradient
Quasi-Newton (BFGS...)
Trust-Region Method
Levenberg Margquardt
Mirror Descent
Dual Projected Subgradient

The General Composite Model

We will be interested in the following model:

$$(P) \qquad \min\{F(\mathbf{x}) \equiv f(\mathbf{x}) + g(\mathbf{x}) : \mathbf{x} \in \mathbb{E}\}.$$

- $f : \mathbb{R}^n \to \mathbb{R}$ is an L_f-smooth convex functin:

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L_f \|\mathbf{x} - \mathbf{y}\| \quad \text{for every } \mathbf{x}, \mathbf{y} \in \mathbb{R}^n,$$

- $g : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ extended valued convex function which is **nonsmooth**.
- Problem (P) is solvable, i.e., $X_* := \operatorname{argmin} f \neq \emptyset$, and for $\mathbf{x}^* \in X_*$ we set $F_{\mathrm{opt}} := F(\mathbf{x}^*)$.

Special Cases of the General Model

- $g = 0$ - smooth unconstrained convex minimization.

$$\min_{\mathbf{x}} f(\mathbf{x})$$

Special Cases of the General Model

- $g = 0$ - smooth unconstrained convex minimization.

$$\min_{\mathbf{x}} f(\mathbf{x})$$

- $g = \delta_C(\cdot)$ - constrained smooth convex minimization.

$$\min_{\mathbf{x}} \{f(\mathbf{x}) : \mathbf{x} \in C\}$$

Special Cases of the General Model

- $g = 0$ - smooth unconstrained convex minimization.

$$\min_{\mathbf{x}} f(\mathbf{x})$$

- $g = \delta_C(\cdot)$ - constrained smooth convex minimization.

$$\min_{\mathbf{x}} \{f(\mathbf{x}) : \mathbf{x} \in C\}$$

- $g = \|\cdot\|_1$ - l_1-regularized convex minimization.

$$\min_{\mathbf{x}} \{f(\mathbf{x}) + \lambda \|\mathbf{x}\|_1\}$$

The Proximal Gradient Method

The derivation of the proximal gradient method is similar to the one of the projected subgradient method.

- For any $L \geq L_f$, and a given iterate $\mathbf{x}^k$:

$$Q_L(\mathbf{x}, \mathbf{x}^k) := f(\mathbf{x}^k) + \langle \mathbf{x} - \mathbf{x}^k, \nabla f(\mathbf{x}^k) \rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{x}^k\|^2 + \underbrace{g(\mathbf{x})}_{\text{untouched}}$$

The Proximal Gradient Method

The derivation of the proximal gradient method is similar to the one of the projected subgradient method.

- For any $L \geq L_f$, and a given iterate $\mathbf{x}^k$:

$$Q_L(\mathbf{x}, \mathbf{x}^k) := f(\mathbf{x}^k) + \langle \mathbf{x} - \mathbf{x}^k, \nabla f(\mathbf{x}^k) \rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{x}^k\|^2 + \underbrace{g(\mathbf{x})}_{\text{untouched}}$$

- **Algorithm:**

$$\begin{aligned}
\mathbf{x}^{k+1} &:= \operatorname*{argmin}_{\mathbf{x}} Q_L(\mathbf{x}, \mathbf{x}^k) \\
&= \operatorname*{argmin}_{\mathbf{x}} \left\{ g(\mathbf{x}) + \frac{L}{2}\left\| \mathbf{x} - (\mathbf{x}^k - \frac{1}{L}\nabla f(\mathbf{x}^k)) \right\|^2 \right\} \\
&= \operatorname{prox}_{\frac{1}{L}g}\left(\mathbf{x}^k - \frac{1}{L}\nabla f(\mathbf{x}^k) \right) \equiv p_L(\mathbf{x}^k).
\end{aligned}$$

prox operator: $\operatorname{prox}_g(\mathbf{x}) := \operatorname*{argmin}_{\mathbf{u}} \left\{ g(\mathbf{u}) + \frac{1}{2}\|\mathbf{u} - \mathbf{x}\|^2 \right\}.$

Special Cases

The general method: $\mathbf{x}^{k+1} = \mathrm{prox}_{\frac{1}{L}g}\left(\mathbf{x}^k - \frac{1}{L}\nabla f(\mathbf{x}^k)\right)$.

- $g \equiv 0 \Rightarrow$ the gradient method.

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{1}{L}\nabla f(\mathbf{x}^k)$$

Special Cases

The general method: $\mathbf{x}^{k+1} = \mathrm{prox}_{\frac{1}{L}g}\left(\mathbf{x}^k - \frac{1}{L}\nabla f(\mathbf{x}^k)\right).$

- $g \equiv 0 \Rightarrow$ the gradient method.

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{1}{L}\nabla f(\mathbf{x}^k)$$

- $g = \delta_C(\cdot) \Rightarrow$ the gradient projection method

$$\mathbf{x}^{k+1} = P_C\left(\mathbf{x}^k - \frac{1}{L}\nabla f(\mathbf{x}^k)\right)$$

Special Cases

The general method: $\mathbf{x}^{k+1} = \mathrm{prox}_{\frac{1}{L}g}\left(\mathbf{x}^k - \frac{1}{L}\nabla f(\mathbf{x}^k)\right).$

- $g \equiv 0 \Rightarrow$ the gradient method.

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{1}{L}\nabla f(\mathbf{x}^k)$$

- $g = \delta_C(\cdot) \Rightarrow$ the gradient projection method

$$\mathbf{x}^{k+1} = P_C\left(\mathbf{x}^k - \frac{1}{L}\nabla f(\mathbf{x}^k)\right)$$

- $g(\mathbf{x}) := \lambda\|\mathbf{x}\|_1 \Rightarrow$ Iterative shrinkage/thresholding algorithm

$$\mathbf{x}^{k+1} = \mathcal{T}_{\lambda/L}\left(\mathbf{x}^k - \frac{1}{L}\nabla f(\mathbf{x}^k)\right)$$

and $\mathcal{T}_\alpha : \mathbb{R}^n \to \mathbb{R}^n$ is the shrinkage operator defined by

$$\mathcal{T}_\alpha(\mathbf{x})_i = (|x_i| - \alpha)_+ \mathrm{sgn}\,(x_i).$$

Special Case: LASSO

- $g(\mathbf{x}) := \lambda\|\mathbf{x}\|_1,\ f(\mathbf{x}) := \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$ (prox=shrinkage).

$$\mathbf{x}_{k+1} = \mathcal{T}_{\lambda/L}\left(\mathbf{x}_k - \frac{2}{L}\mathbf{A}^T(\mathbf{A}\mathbf{x}_k - \mathbf{b})\right)$$

ISTA - Iterative Shrinkage/Thresholding Algorithm

In SP literature: Chambolle (98); Figueiredo-Nowak (03, 05); Daubechies et al. (04),Elad et al. (06), Hale et al. (07)...

In Optimization: can be viewed as the Proximal forward backward Splitting Method (Passty (79))

Prox Computations

There are a quite a few “simple” functions for which the prox can be easily computed

Prox Computations

f	$\mathrm{dom}(f)$	$\mathrm{prox}_f(\mathbf{x})$	assumptions	reference
$\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$	$\mathbb{R}^n$	$(\mathbf{A}+\mathbf{I})^{-1}(\mathbf{x}-\mathbf{b})$	$\mathbf{A} \in \mathbb{S}^n_+$, $\mathbf{b} \in \mathbb{R}^n$, $c \in \mathbb{R}$	Section 6.2.3
λx^3	$\mathbb{R}_+$	$\frac{-1+\sqrt{1+12\lambda[x]_+}}{6\lambda}$	$\lambda > 0$	Lemma 6.5
μx	$[0,\alpha]$	$\min\{\max\{x-\mu,0\},\alpha\}$	$\mu \in \mathbb{R}$, $\alpha \in \mathbb{R}_+$	Example 6.14
$\lambda\\|\mathbf{x}\\|$	$\mathbb{E}$	$\left(1-\frac{\lambda}{\max\{\\|\mathbf{x}\\|,\lambda\}}\right)\mathbf{x}$	$\\|\cdot\\|$ - Euclidean norm, $\lambda > 0$	Example 6.19
$-\lambda\\|\mathbf{x}\\|$	$\mathbb{E}$	$\left(1+\frac{\lambda}{\\|\mathbf{x}\\|}\right)\mathbf{x}, \quad \mathbf{x} \neq \mathbf{0}$, $\{\mathbf{u} : \\|\mathbf{u}\\| = \lambda\}, \quad \mathbf{x} = \mathbf{0}$.	$\\|\cdot\\|$ - Euclidean norm, $\lambda > 0$	Example 6.21
$\lambda\\|\mathbf{x}\\|_1$	$\mathbb{R}^n$	$\mathcal{T}_\lambda(\mathbf{x}) = [\lvert\mathbf{x}\rvert - \lambda\mathbf{e}]_+ \odot \mathrm{sgn}(\mathbf{x})$	$\lambda > 0$	Example 6.8
$\\|\boldsymbol{\omega} \odot \mathbf{x}\\|_1$	$\mathrm{Box}[-\boldsymbol{\alpha},\boldsymbol{\alpha}]$	$\mathcal{S}_{\boldsymbol{\omega},\boldsymbol{\alpha}}(\mathbf{x})$	$\boldsymbol{\alpha} \in [0,\infty]^n$, $\boldsymbol{\omega} \in \mathbb{R}^n_{++}$	Example 6.23
$\lambda\\|\mathbf{x}\\|_\infty$	$\mathbb{R}^n$	$\mathbf{x} - \lambda P_{B_{\\|\cdot\\|_1}[\mathbf{0},1]}(\mathbf{x}/\lambda)$	$\lambda > 0$	Example 6.48
$\lambda\\|\mathbf{x}\\|_a$	$\mathbb{E}$	$\mathbf{x} - \lambda P_{B_{\\|\cdot\\|_{a,*}}[\mathbf{0},1]}(\mathbf{x}/\lambda)$	$\\|\mathbf{x}\\|_a$ – norm, $\lambda > 0$	Example 6.47
$\lambda\\|\mathbf{x}\\|_0$	$\mathbb{R}^n$	$H_{\sqrt{2\lambda}}(x_1) \times \cdots \times H_{\sqrt{2\lambda}}(x_n)$	$\lambda > 0$	Example 6.10
$\lambda\\|\mathbf{x}\\|^3$	$\mathbb{E}$	$\frac{2}{1+\sqrt{1+12\lambda\\|\mathbf{x}\\|}}\mathbf{x}$	$\\|\cdot\\|$ - Euclidean norm, $\lambda > 0$,	Example 6.20
$-\lambda\sum_{j=1}^n \log x_j$	$\mathbb{R}^n_{++}$	$\left(\frac{x_j+\sqrt{x_j^2+4\lambda}}{2}\right)_{j=1}^n$	$\lambda > 0$	Example 6.9
$\delta_C(\mathbf{x})$	$\mathbb{E}$	$P_C(\mathbf{x})$	$\emptyset \neq C \subseteq \mathbb{E}$	Theorem 6.24
$\lambda\sigma_C(\mathbf{x})$	$\mathbb{E}$	$\mathbf{x} - \lambda P_C(\mathbf{x}/\lambda)$	$\lambda > 0$, $C \neq \emptyset$ closed convex	Theorem 6.46
$\lambda\max\{x_i\}$	$\mathbb{R}^n$	$\mathbf{x} - P_{\Delta_n}(\mathbf{x}/\lambda)$	$\lambda > 0$	Example 6.49
$\lambda\sum_{i=1}^k x_{[i]}$	$\mathbb{R}^n$	$\mathbf{x} - \lambda P_C(\mathbf{x}/\lambda)$, $C = H_{\mathbf{e},k} \cap \mathrm{Box}[\mathbf{0},\mathbf{e}]$	$\lambda > 0$	Example 6.50
$\lambda\sum_{i=1}^k \lvert x_{\langle i\rangle}\rvert$	$\mathbb{R}^n$	$\mathbf{x} - \lambda P_C(\mathbf{x}/\lambda)$, $C = B_{\\|\cdot\\|_1}[\mathbf{0},k] \cap \mathrm{Box}[-\mathbf{e},\mathbf{e}]$	$\lambda > 0$	Example 6.51
$\lambda M_f^\mu(\mathbf{x})$	$\mathbb{E}$	$\mathbf{x} + \frac{\lambda}{\mu+\lambda}\left(\mathrm{prox}_{(\mu+\lambda)f}(\mathbf{x}) - \mathbf{x}\right)$	$\lambda,\mu > 0$, f proper closed convex	Corollary 6.63
$\lambda d_C(\mathbf{x})$	$\mathbb{E}$	$\mathbf{x} + \min\left\{\frac{\lambda}{d_C(\mathbf{x})},1\right\}(P_C(\mathbf{x}) - \mathbf{x})$	C nonempty closed convex, $\lambda > 0$	Lemma 6.43
$\frac{\lambda}{2}d_C^2(\mathbf{x})$	$\mathbb{E}$	$\frac{\lambda}{\lambda+1}P_C(\mathbf{x}) + \frac{1}{\lambda+1}\mathbf{x}$	C nonempty closed convex, $\lambda > 0$	Example 6.64
$\lambda H_\mu(\mathbf{x})$	$\mathbb{E}$	$\left(1-\frac{\lambda}{\max\{\\|\mathbf{x}\\|,\mu+\lambda\}}\right)$	$\lambda,\mu > 0$	Example 6.65
$\rho\\|\mathbf{x}\\|_1^2$	$\mathbb{R}^n$	$\left(\frac{v_i x_i}{v_i+2\rho}\right)_{i=1}^n$, $\mathbf{v} = \left[\sqrt{\frac{\rho}{\mu}}\lvert\mathbf{x}\rvert - 2\rho\right]_+$, $\mathbf{e}^T\mathbf{v} = 1$ ($\mathbf{0}$ when $\mathbf{x} = \mathbf{0}$)	$\rho > 0$	Lemma 6.69
$\\|\mathbf{A}\mathbf{x}\\|_2$	$\mathbb{R}^n$	$\mathbf{x} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \alpha^*\mathbf{I})^{-1}\mathbf{A}\mathbf{x}$, $\alpha^* = 0$ if $\\|\mathbf{v}_0\\|_2 \leq \lambda$; otherwise, $\\|\mathbf{v}_{\alpha^*}\\|_2 = \lambda$; $\mathbf{v}_\alpha \equiv (\mathbf{A}\mathbf{A}^T + \alpha\mathbf{I})^{-1}\mathbf{A}\mathbf{x}$	$\mathbf{A} \in \mathbb{R}^{m\times n}$ with full row rank	Lemma 6.67

Prox of Symmetric Spectral Functions over $\mathbb{S}^n$ (From Example 7.19)

$F(\mathbf{X})$	$\mathrm{dom}(F)$	$\mathrm{prox}_F(\mathbf{X})$
$\alpha\|\mathbf{X}\|_F^2$	$\mathbb{S}^n$	$\frac{1}{1+2\alpha}\mathbf{X}$
$\alpha\|\mathbf{X}\|_F$	$\mathbb{S}^n$	$\left(1 - \frac{\alpha}{\max\{\|\mathbf{X}\|_F,\alpha\}}\right)\mathbf{X}$
$\alpha\|\mathbf{X}\|_{S_1}$	$\mathbb{S}^n$	$\mathbf{U}\mathcal{T}_\alpha(\boldsymbol{\lambda}(\mathbf{X}))\mathbf{U}^T$
$\alpha\|\mathbf{X}\|_{2,2}$	$\mathbb{S}^n$	$\mathbf{U}\mathrm{diag}(\boldsymbol{\lambda}(\mathbf{X}) - \alpha P_{B_{\|\cdot\|_1}[\mathbf{0},1]}(\boldsymbol{\lambda}(\mathbf{X})/\alpha))\mathbf{U}^T$
$-\alpha\det(\mathbf{X})$	$\mathbb{S}^n_{++}$	$\mathbf{U}\mathrm{diag}\left(\frac{\lambda_j(\mathbf{X})+\sqrt{\lambda_j(\mathbf{X})^2+4\alpha}}{2}\right)\mathbf{U}^T$
$\alpha\lambda_1(\mathbf{X})$	$\mathbb{S}^n$	$\mathbf{U}\mathrm{diag}(\boldsymbol{\lambda}(\mathbf{X}) - P_{\Delta_n}(\boldsymbol{\lambda}(\mathbf{X})/\alpha))\mathbf{U}^T$
$\alpha\sum_{i=1}^k \lambda_i(\mathbf{X})$	$\mathbb{S}^n$	$\mathbf{X} - \alpha\mathbf{U}P_C(\boldsymbol{\lambda}(\mathbf{X})/\alpha)\mathbf{U}^T$, $C = H_{\mathbf{e},k} \cap \mathrm{Box}[\mathbf{0},\mathbf{e}]$

Prox of Symmetric Spectral Functions over $\mathbb{R}^{m\times n}$ (From Example 7.30)

$F(\mathbf{X})$	$\mathrm{prox}_F(\mathbf{X})$
$\alpha\|\mathbf{X}\|_F^2$	$\frac{1}{1+2\alpha}\mathbf{X}$
$\alpha\|\mathbf{X}\|_F$	$\left(1 - \frac{\alpha}{\max\{\|\mathbf{X}\|_F,\alpha\}}\right)\mathbf{X}$
$\alpha\|\mathbf{X}\|_{S_1}$	$\mathbf{U}\mathcal{T}_\alpha(\boldsymbol{\sigma}(\mathbf{X}))\mathbf{V}^T$
$\alpha\|\mathbf{X}\|_{S_\infty}$	$\mathbf{X} - \alpha\mathbf{U}\mathrm{diag}(P_{B_{\|\cdot\|_1}[\mathbf{0},1]}(\boldsymbol{\sigma}(\mathbf{X})/\alpha))\mathbf{V}^T$
$\alpha\|\mathbf{X}\|_{\langle k\rangle}$	$\mathbf{X} - \alpha\mathbf{U}P_C(\boldsymbol{\sigma}(\mathbf{X})/\alpha)\mathbf{V}^T$, $C = B_{\|\cdot\|_1}[\mathbf{0},k] \cap B_{\|\cdot\|_\infty}[\mathbf{0},1]$

Orthogonal Projections

set (C)	$P_C(\mathbf{x})$	assumptions	reference
$\mathbb{R}^n_+$	$[\mathbf{x}]_+$	$-$	Lemma 6.26
$\text{Box}[\boldsymbol{\ell}, \mathbf{u}]$	$P_C(\mathbf{x})_i = \min\{\max\{x_i, \ell_i\}, u_i\}$	$\ell_i \le u_i$	Lemma 6.26
$B_{\lVert\cdot\rVert_2}[\mathbf{c}, r]$	$\mathbf{c} + \frac{r}{\max\{\lVert\mathbf{x}-\mathbf{c}\rVert_2, r\}}(\mathbf{x} - \mathbf{c})$	$\mathbf{c} \in \mathbb{R}^n, r > 0$	Lemma 6.26
$\{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}\}$	$\mathbf{x} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}(\mathbf{A}\mathbf{x} - \mathbf{b})$	$\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{A}$ full row rank	Lemma 6.26
$\{\mathbf{x} : \mathbf{a}^T\mathbf{x} \le b\}$	$\mathbf{x} - \frac{[\mathbf{a}^T\mathbf{x}-b]_+}{\lVert\mathbf{a}\rVert^2}\mathbf{a}$	$\mathbf{0} \ne \mathbf{a} \in \mathbb{R}^n, b \in \mathbb{R}$	Lemma 6.26
Δ_n	$[\mathbf{x} - \mu^*\mathbf{e}]_+$ where $\mu^* \in \mathbb{R}$ satisfies $\mathbf{e}^T[\mathbf{x} - \mu^*\mathbf{e}]_+ = 1$		Corollary 6.29
$H_{\mathbf{a},b} \cap \text{Box}[\boldsymbol{\ell}, \mathbf{u}]$	$P_{\text{Box}(\boldsymbol{\ell},\mathbf{u})}(\mathbf{x} - \mu^*\mathbf{a})$ where $\mu^* \in \mathbb{R}$ satisfies $\mathbf{a}^T P_{\text{Box}[\boldsymbol{\ell},\mathbf{u}]}(\mathbf{x} - \mu^*\mathbf{a}) = b$	$\mathbf{a} \in \mathbb{R}^n \backslash \{\mathbf{0}\}, b \in \mathbb{R}$	Theorem 6.27
$H^-_{\mathbf{a},b} \cap \text{Box}[\boldsymbol{\ell}, \mathbf{u}]$	$\begin{cases} P_{\text{Box}[\boldsymbol{\ell},\mathbf{u}]}(\mathbf{x}), & \mathbf{a}^T\mathbf{v}_{\mathbf{x}} \le b, \\ P_{\text{Box}[\boldsymbol{\ell},\mathbf{u}]}(\mathbf{x} - \lambda^*\mathbf{a}), & \mathbf{a}^T\mathbf{v}_{\mathbf{x}} > b, \end{cases}$ $\mathbf{v}_{\mathbf{x}} = P_{\text{Box}[\boldsymbol{\ell},\mathbf{u}]}(\mathbf{x})$, $\mathbf{a}^T P_{\text{Box}[\boldsymbol{\ell},\mathbf{u}]}(\mathbf{x} - \lambda^*\mathbf{a}) = b, \lambda^* > 0$	$\mathbf{a} \in \mathbb{R}^n \backslash \{\mathbf{0}\}, b \in \mathbb{R}$	Example 6.32
$B_{\lVert\cdot\rVert_1}[\mathbf{0}, \alpha]$	$\begin{cases} \mathbf{x}, & \lVert\mathbf{x}\rVert_1 \le \alpha, \\ \mathcal{T}_{\lambda^*}(\mathbf{x}), & \lVert\mathbf{x}\rVert_1 > \alpha, \end{cases}$ $\lVert\mathcal{T}_{\lambda^*}(\mathbf{x})\rVert_1 = \alpha$, $\lambda^* > 0$	$\alpha > 0$	Example 6.33
$\{\mathbf{x} : \boldsymbol{\omega}^T\lvert\mathbf{x}\rvert \le \beta,$ $-\boldsymbol{\alpha} \le \mathbf{x} \le \boldsymbol{\alpha}\}$	$\begin{cases} \mathbf{v}_{\mathbf{x}}, & \boldsymbol{\omega}^T\lvert\mathbf{v}_{\mathbf{x}}\rvert \le \beta, \\ \mathcal{S}_{\lambda^*\boldsymbol{\omega},\boldsymbol{\alpha}}(\mathbf{x}), & \boldsymbol{\omega}^T\lvert\mathbf{v}_{\mathbf{x}}\rvert > \beta, \end{cases}$ $\mathbf{v}_{\mathbf{x}} = P_{\text{Box}[-\boldsymbol{\alpha},\boldsymbol{\alpha}]}(\mathbf{x})$, $\boldsymbol{\omega}^T\lvert\mathcal{S}_{\lambda^*\boldsymbol{\omega},\boldsymbol{\alpha}}(\mathbf{x})\rvert = \beta, \lambda^* > 0$	$\boldsymbol{\omega} \in \mathbb{R}^n_{++}$, $\boldsymbol{\alpha} \in [0, \infty]^n$, $\beta \in \mathbb{R}_{++}$	Example 6.34
$\{\mathbf{x} > \mathbf{0} : \Pi x_i \ge \alpha\}$	$\begin{cases} \mathbf{x}, & \mathbf{x} \in C, \\ \left(\frac{x_j + \sqrt{x_j^2 + 4\lambda^*}}{2}\right)_{j=1}^n, & \mathbf{x} \notin C, \end{cases}$, $\Pi_{j=1}^n \left((x_j + \sqrt{x_j^2 + 4\lambda^*})/2\right) = \alpha, \lambda^* > 0$	$\alpha > 0$	Example 6.35
$\{(\mathbf{x}, s) : \lVert\mathbf{x}\rVert_2 \le s\}$	$\left(\frac{\lVert\mathbf{x}\rVert_2 + s}{2\lVert\mathbf{x}\rVert_2}\mathbf{x}, \frac{\lVert\mathbf{x}\rVert_2 + s}{2}\right)$ if $\lVert\mathbf{x}\rVert_2 \ge \lvert s\rvert$ $(\mathbf{0}, 0)$ if $s < \lVert\mathbf{x}\rVert_2 < -s$, $(\mathbf{x}, s)$ if $\lVert\mathbf{x}\rVert_2 \le s$.		Example 6.37
$\{(\mathbf{x}, s) : \lVert\mathbf{x}\rVert_1 \le s\}$	$\begin{cases} (\mathbf{x}, s), & \lVert\mathbf{x}\rVert_1 \le s, \\ (\mathcal{T}_{\lambda^*}(\mathbf{x}), s + \lambda^*), & \lVert\mathbf{x}\rVert_1 > s, \end{cases}$ $\lVert\mathcal{T}_{\lambda^*}(\mathbf{x})\rVert_1 - \lambda^* - s = 0$, $\lambda^* > 0$		Example 6.38

Orthogonal Projections onto Symmetric Spectral Sets in $\mathbb{S}^n$

set (C)	$P_C(\mathbf{X})$	assumptions
$\mathbb{S}^n_+$	$\mathbf{U}\mathrm{diag}([\boldsymbol{\lambda}(\mathbf{X})]_+)\mathbf{U}^T$	–
$\{\mathbf{X} : \ell\mathbf{I} \preceq \mathbf{X} \preceq u\mathbf{I}\}$	$\mathbf{U}\mathrm{diag}(\mathbf{v})\mathbf{U}^T$, $v_i = \min\{\max\{\lambda_i(\mathbf{X}), \ell\}, u\}$	$\ell \leq u$
$B_{\\|\cdot\\|_F}[\mathbf{0}, r]$	$\frac{r}{\max\{\\|\mathbf{X}\\|_F, r\}}\mathbf{X}$	$r > 0$
$\{\mathbf{X} : \mathrm{Tr}(\mathbf{X}) \leq b\}$	$\mathbf{U}\mathrm{diag}(\mathbf{v})\mathbf{U}^T$, $\mathbf{v} = \boldsymbol{\lambda}(\mathbf{X}) - \frac{[\mathbf{e}^T\boldsymbol{\lambda}(\mathbf{X}) - b]_+}{n}\mathbf{e}$	$b \in \mathbb{R}$
Υ_n	$\mathbf{U}\mathrm{diag}(\mathbf{v})\mathbf{U}^T$, $\mathbf{v} = [\boldsymbol{\lambda}(\mathbf{X}) - \mu^*\mathbf{e}]_+$ where $\mu^* \in \mathbb{R}$ satisfies $\mathbf{e}^T[\boldsymbol{\lambda}(\mathbf{X}) - \mu^*\mathbf{e}]_+ = 1$	-
$B_{\\|\cdot\\|_{S_1}}[\mathbf{0}, \alpha]$	$\begin{cases} \mathbf{X}, & \\|\mathbf{X}\\|_{S_1} \leq \alpha, \\ \mathbf{U}\mathcal{T}_{\lambda^*}(\boldsymbol{\lambda}(\mathbf{X}))\mathbf{U}^T, & \\|\mathbf{X}\\|_{S_1} > \alpha, \end{cases}$ $\\|\mathcal{T}_{\lambda^*}(\boldsymbol{\lambda}(\mathbf{X}))\\|_1 = \alpha$, $\lambda^* > 0$	$\alpha > 0$

Orthogonal Projection onto Symmetric Spectral Sets in $\mathbb{R}^{m\times n}$ (From Example 7.31)

set (C)	$P_C(\mathbf{X})$	assumptions
$B_{\\|\cdot\\|_{S_\infty}}[\mathbf{0}, \alpha]$	$\mathbf{U}\mathrm{diag}(\mathbf{v})\mathbf{U}^T$, $v_i = \min\{\sigma_i(\mathbf{X}), \alpha\}$	$\alpha > 0$
$B_{\\|\cdot\\|_F}[\mathbf{0}, r]$	$\frac{r}{\max\{\\|\mathbf{X}\\|_F, r\}}\mathbf{X}$	$r > 0$
$B_{\\|\cdot\\|_{S_1}}[\mathbf{0}, \alpha]$	$\begin{cases} \mathbf{X}, & \\|\mathbf{X}\\|_{S_1} \leq \alpha, \\ \mathbf{U}\mathcal{T}_{\lambda^*}(\sigma(\mathbf{X}))\mathbf{U}^T, & \\|\mathbf{X}\\|_{S_1} > \alpha, \end{cases}$ $\\|\mathcal{T}_{\lambda^*}(\boldsymbol{\sigma}(\mathbf{X}))\\|_1 = \alpha$, $\lambda^* > 0$	$\alpha > 0$

Rate of Convergence of Prox-Grad

Theorem - [Rate of Convergence of Prox-Grad]
Let $\{\mathbf{x}^k\}$ be the sequence generated by the proximal gradient method.

$$F(\mathbf{x}^k) - F(\mathbf{x}^*) \leq \frac{L\|\mathbf{x}^0 - \mathbf{x}^*\|^2}{2k}$$

for any optimal solution $\mathbf{x}^*$.

- Thus, to solve (M), the proximal gradient method converges at a *sublinear rate* in function values.
- # iterations for $F(\mathbf{x}^k) - F(\mathbf{x}^*) \leq \varepsilon$ is $O(1/\varepsilon)$.

Rate of Convergence of Prox-Grad

Theorem - [Rate of Convergence of Prox-Grad]
Let $\{\mathbf{x}^k\}$ be the sequence generated by the proximal gradient method.

$$F(\mathbf{x}^k) - F(\mathbf{x}^*) \leq \frac{L\|\mathbf{x}^0 - \mathbf{x}^*\|^2}{2k}$$

for any optimal solution $\mathbf{x}^*$.

- Thus, to solve (M), the proximal gradient method converges at a *sublinear rate* in function values.
- # iterations for $F(\mathbf{x}^k) - F(\mathbf{x}^*) \leq \varepsilon$ is $O(1/\varepsilon)$.
- **Note:** The sequence $\{\mathbf{x}^k\}$ can be proven to *converge* to solution $\mathbf{x}^*$.
- No need to know the Lipschitz constant (backtracking).

Towards a Faster Algorithm

- An $O(1/k)$ rate of convergence is rather slow.
- Can we find a faster methods?

Towards a Faster Algorithm

- An $O(1/k)$ rate of convergence is rather slow.
- Can we find a faster methods?
- The answer is YES!.

FISTA - [B., Teboulle 2009]

An equally simple algorithm as prox-grad. (Here L_f is known).

FISTA with constant stepsize

Input: $L \geq L_f$ - A Lipschitz constant of ∇f.

Step 0. Take $\mathbf{y}^1 = \mathbf{x}^0 \in \mathbb{E},\ t_1 = 1$.

Step k. $(k \geq 1)$ Compute

$$\mathbf{x}^k \equiv \operatorname{prox}_{\frac{1}{L}g}\left(\mathbf{y}^k - \frac{1}{L}\nabla f(\mathbf{y}^k)\right), \quad \hookleftarrow \text{ main computation}$$

$$\bullet \quad t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2},$$

$$\bullet\bullet \quad \mathbf{y}^{k+1} = \mathbf{x}^k + \left(\frac{t_k - 1}{t_{k+1}}\right)(\mathbf{x}^k - \mathbf{x}^{k-1}).$$

Additional computation for FISTA in (•) and (••) is clearly marginal.

Theorem - Global Rate of Convergence FISTA

Theorem Let $\{\mathbf{x}_k\}$ be generated by FISTA. Then for any $k \geq 1$

$$F(\mathbf{x}_k) - F(\mathbf{x}^*) \leq \frac{2\alpha L(f)\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{(k+1)^2},$$

where $\alpha = 1$ for the constant stepsize setting and $\alpha = \eta$ for the backtracking stepsize setting.

- $\#$ of iterations to reach $F(\tilde{\mathbf{x}}) - F_* \leq \varepsilon$ is $\sim O(1/\sqrt{\varepsilon})$.
- Clearly improves ISTA by **a square root factor**.

Theorem - Global Rate of Convergence FISTA

Theorem Let $\{\mathbf{x}_k\}$ be generated by FISTA. Then for any $k \geq 1$

$$F(\mathbf{x}_k) - F(\mathbf{x}^*) \leq \frac{2\alpha L(f)\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{(k+1)^2},$$

where $\alpha = 1$ for the constant stepsize setting and $\alpha = \eta$ for the backtracking stepsize setting.

- $\#$ of iterations to reach $F(\tilde{\mathbf{x}}) - F_* \leq \varepsilon$ is $\sim O(1/\sqrt{\varepsilon})$.
- Clearly improves ISTA by **a square root factor**.
- **Do we practically achieve this theoretical rate? Yes**

LASSO (Penalized Version)

- Consider the problem

$$\text{(P)} \quad \min\left\{f(\mathbf{x}) \equiv \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_1\right\}$$

$\mathbf{A} \in \mathbb{R}^{100\times 200}, \mathbf{b} \in \mathbb{R}^{100}, \lambda > 0$

LASSO (Penalized Version)

- Consider the problem

$$\text{(P)} \quad \min \left\{ f(\mathbf{x}) \equiv \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_1 \right\}$$

$\mathbf{A} \in \mathbb{R}^{100\times 200}, \mathbf{b} \in \mathbb{R}^{100}, \lambda > 0$

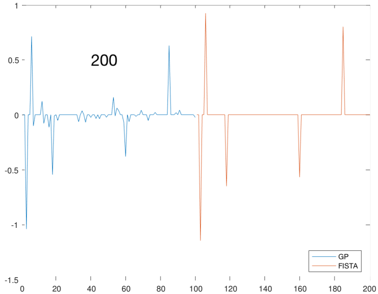
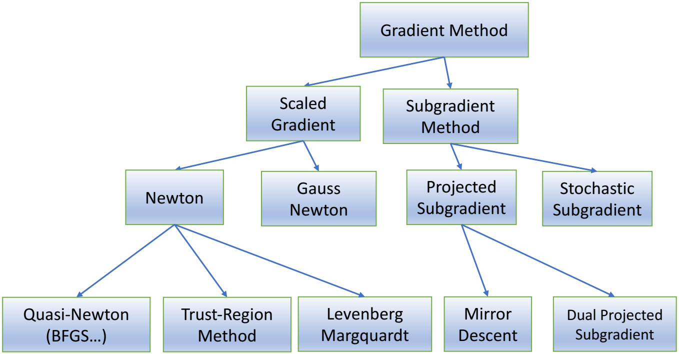
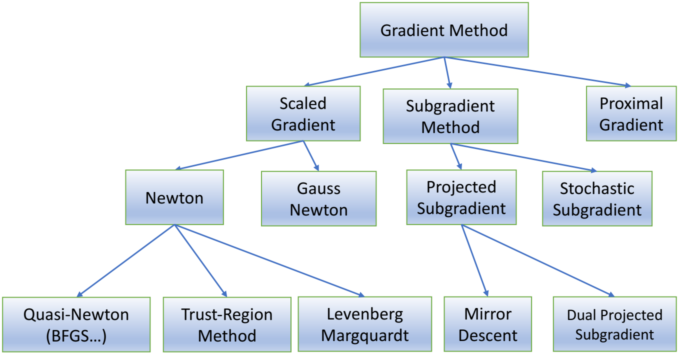
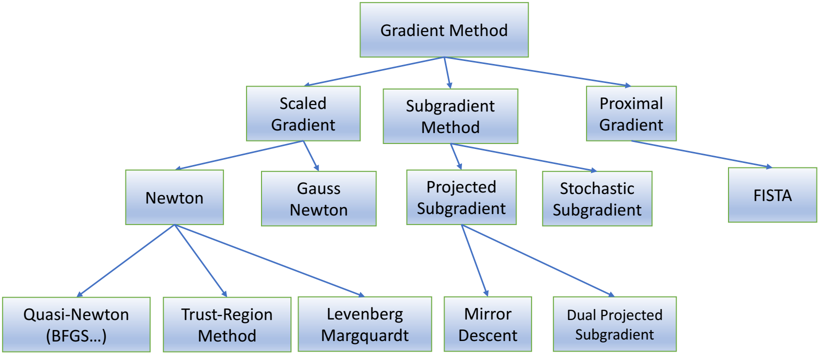


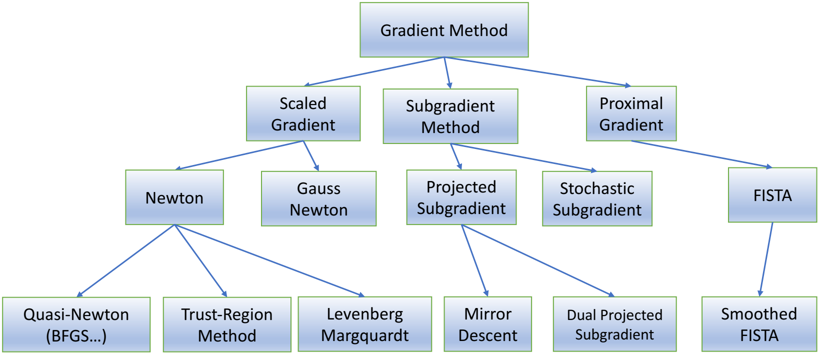
Illustration $(\lambda = 1)$


Gradient Method
Scaled Gradient
Subgradient Method
Newton
Gauss Newton
Projected Subgradient
Stochastic Subgradient
Quasi-Newton (BFGS...)
Trust-Region Method
Levenberg Margquardt
Mirror Descent
Dual Projected Subgradient



Gradient Method
Scaled Gradient
Subgradient Method
Proximal Gradient
Newton
Gauss Newton
Projected Subgradient
Stochastic Subgradient
Quasi-Newton (BFGS…)
Trust-Region Method
Levenberg Margquardt
Mirror Descent
Dual Projected Subgradient


Gradient Method
Scaled Gradient
Subgradient Method
Proximal Gradient
Newton
Gauss Newton
Projected Subgradient
Stochastic Subgradient
FISTA
Quasi-Newton (BFGS...)
Trust-Region Method
Levenberg Margquardt
Mirror Descent
Dual Projected Subgradient


Gradient Method
Scaled Gradient
Subgradient Method
Proximal Gradient
Newton
Gauss Newton
Projected Subgradient
Stochastic Subgradient
FISTA
Quasi-Newton (BFGS…)
Trust-Region Method
Levenberg Margquardt
Mirror Descent
Dual Projected Subgradient
Smoothed FISTA

Smoothed FISTA

- Revisit nonsmooth problems:

$$\min_{\mathbf{x}} \{f(\mathbf{x}) : \mathbf{x} \in C\}$$

f - convex nonsmooth, C - convex

Smoothed FISTA

- Revisit nonsmooth problems:

$$\min_{\mathbf{x}}\{f(\mathbf{x}) : \mathbf{x} \in C\}$$

  f - convex nonsmooth, C - convex
- "standard" nonsmooth algorithms solve it in $O(1/\varepsilon^2)$ complexity.

Smoothed FISTA

- Revisit nonsmooth problems:

$$\min_{\mathbf{x}} \{ f(\mathbf{x}) : \mathbf{x} \in C \}$$

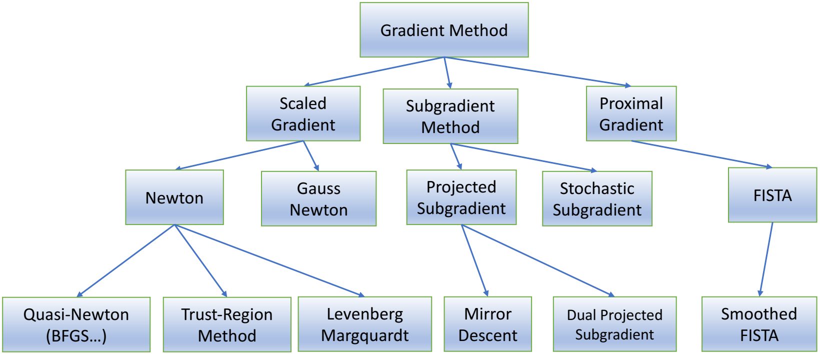
  f - convex nonsmooth, C - convex
- "standard" nonsmooth algorithms solve it in $O(1/\varepsilon^2)$ complexity.
- Another approach: consider a smoothed version of the problem: $\min_{\mathbf{x} \in C} f_\eta(\mathbf{x})$ and solve it using FISTA.
- Example: $\sum_{i=1}^m |\mathbf{a}_i^T \mathbf{x} - b_i| \to \sum_{i=1}^m \sqrt{(\mathbf{a}_i^T \mathbf{x} - b_i)^2 + \eta^2}$

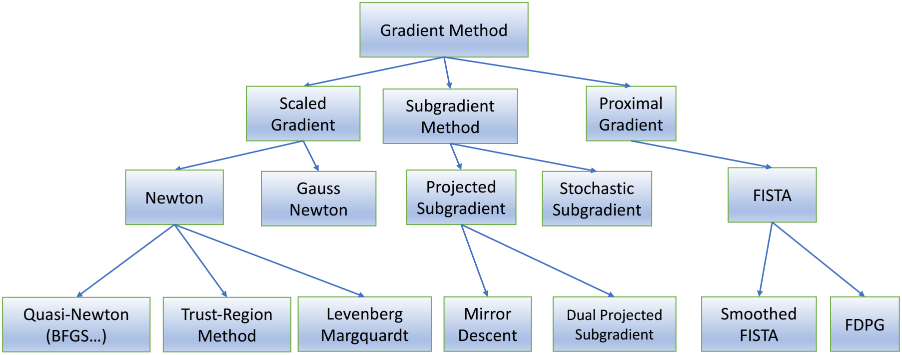
Smoothed FISTA

- Revisit nonsmooth problems:

$$\min_{\mathbf{x}}\{f(\mathbf{x}) : \mathbf{x} \in C\}$$

  f - convex nonsmooth, C - convex
- "standard" nonsmooth algorithms solve it in $O(1/\varepsilon^2)$ complexity.
- Another approach: consider a smoothed version of the problem: $\min_{\mathbf{x}\in C} f_\eta(\mathbf{x})$ and solve it using FISTA.
- Example: $\sum_{i=1}^m |\mathbf{a}_i^T\mathbf{x} - b_i| \rightarrow \sum_{i=1}^m \sqrt{(\mathbf{a}_i^T\mathbf{x} - b_i)^2 + \eta^2}$
- Carefully choosing the smoothing parameter, $O(1/\varepsilon)$ complexity can be shown.


Gradient Method
Scaled Gradient
Subgradient Method
Proximal Gradient
Newton
Gauss Newton
Projected Subgradient
Stochastic Subgradient
FISTA
Quasi-Newton (BFGS...)
Trust-Region Method
Levenberg Margquardt
Mirror Descent
Dual Projected Subgradient
Smoothed FISTA



Gradient Method
Scaled Gradient
Subgradient Method
Proximal Gradient
Newton
Gauss Newton
Projected Subgradient
Stochastic Subgradient
FISTA
Quasi-Newton (BFGS…)
Trust-Region Method
Levenberg Margquardt
Mirror Descent
Dual Projected Subgradient
Smoothed FISTA
FDPG

Dual FISTA - FDPG

- Model:

$$\boxed{\min_{\mathbf{x}} f(\mathbf{x}) + g(\mathbf{A}\mathbf{x})}$$

Dual FISTA - FDPG

- Model:

$$\boxed{\min_{\mathbf{x}} f(\mathbf{x}) + g(\mathbf{A}\mathbf{x})}$$

- Dual model:

$$\boxed{\max_{\mathbf{y}} -f^*(\mathbf{A}^T\mathbf{y}) - g^*(-\mathbf{y})}$$

f^*, g^* - convex conjugates ($h^*(\mathbf{y}) \equiv \max_{\mathbf{x}}\{\mathbf{x}^T\mathbf{y} - h(\mathbf{x})\}$)

Dual FISTA - FDPG

- Model:

$$\boxed{\min_{\mathbf{x}} f(\mathbf{x}) + g(\mathbf{A}\mathbf{x})}$$

- Dual model:

$$\boxed{\max_{\mathbf{y}} -f^*(\mathbf{A}^T\mathbf{y}) - g^*(-\mathbf{y})}$$

  f^*, g^* - convex conjugates ($h^*(\mathbf{y}) \equiv \max_{\mathbf{x}}\{\mathbf{x}^T\mathbf{y} - h(\mathbf{x})\}$)
- Apply FISTA on the dual.
- Can deal with many different types of problems...

FDPG

The Fast Dual Proximal Gradient (FDPG) Method - primal representation

Initialization: $L \geq L_F = \frac{\|\mathbf{A}\|^2}{\sigma}, \mathbf{w}^0 = \mathbf{y}^0 \in \mathbb{R}^m, t_0 = 1.$
General step $(k \geq 0)$**:**

(a) $\mathbf{u}^k = \underset{\mathbf{u}}{\operatorname{argmax}} \left\{ \langle \mathbf{u}, \mathbf{A}^T(\mathbf{w}^k) \rangle - f(\mathbf{u}) \right\}.$

(b) $\mathbf{y}^{k+1} = \mathbf{w}^k - \frac{1}{L}\mathbf{A}(\mathbf{u}^k) + \frac{1}{L}\mathrm{prox}_{Lg}(\mathbf{A}(\mathbf{u}^k) - L\mathbf{w}^k)$

(c) $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$

(d) $\mathbf{w}^{k+1} = \mathbf{y}^{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right)(\mathbf{y}^{k+1} - \mathbf{y}^k).$

Present and Future?

The scale of problems is becoming huge. Emphasis of current and probably near future research:

- Decomposition
- Randomization
- Distributed

Thank You

Any Questions????