SVAN 2016 Mini-Course
Stochastic Convex Optimization Methods in Machine Learning

Mark Schmidt

University of British Columbia, May 2016

www.cs.ubc.ca/~schmidtm/SVAN16
Last Time: Projected-Gradient

- We can convert the non-smooth problem

  \[
  \arg\min_{x \in \mathbb{R}^d} f(x) + \lambda \sum_{g \in G} \|x_g\|_2,
  \]

  into a smooth problem with simple constraints:

  \[
  \arg\min_{x \in \mathbb{R}^d} f(x) + \lambda \sum_{g \in G} r_g, \text{ subject to } r_g \geq \|x_g\|_2 \text{ for all } g.
  \]
Last Time: Projected-Gradient

We can convert the non-smooth problem

$$\arg\min_{x \in \mathbb{R}^d} f(x) + \lambda \sum_{g \in G} \|x_g\|_2,$$

into a smooth problem with simple constraints:

$$\arg\min_{x \in \mathbb{R}^d} f(x) + \lambda \sum_{g \in G} r_g, \text{ subject to } r_g \geq \|x_g\|_2 \text{ for all } g.$$

With simple constraints, we can use projected-gradient:

$$x^{t+1} = \arg\min_{y \in C} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{L}{2} \|y - x^t\|^2 \right\},$$

or equivalently projection applied to gradient step:

$$x^{t+1} = \arg\min_{y \in C} \{ \|y - x^{GD}_t\| \}, \text{ where } x^{GD}_t = x^t - \alpha_t \nabla f(x^t).$$
Last Time: Projected-Gradient

\[ x^{t+1} = \arg\min_{y \in C} \{ \| y - x_GD^t \| \}, \text{ where } x_GD^t = x^t - \alpha_t \nabla f(x^t). \]

projection of \( x_GD^t \) onto \( C \)

\[ \text{gradient step} \]
Last Time: Projected-Gradient

\[ x^{t+1} = \arg\min_{y \in C} \{ \| y - x_t^{GD} \| \}, \text{ where } x_t^{GD} = x_t - \alpha_t \nabla f(x_t). \]

projection of \( x_t^{GD} \) onto \( C \)

gradient step
Last Time: Projected-Gradient

\[ x^{t+1} = \arg\min_{y \in C} \{ \| y - x_t^{GD} \| \}, \quad \text{where } x_t^{GD} = x_t - \alpha_t \nabla f(x_t). \]

projection of \( x_t^{GD} \) onto \( C \),

\[ f(x) \]

Feasible Set
Last Time: Projected-Gradient

\[ x^{t+1} = \arg\min_{y \in C} \{ \| y - x^{GD}_t \| \}, \text{ where } x^{GD}_t = x^t - \alpha_t \nabla f(x^t). \]

projection of \( x^{GD}_t \) onto \( C \),

gradient step
Last Time: Projected-Gradient

\[ x^{t+1} = \arg\min_{y \in C} \{ \| y - x^{GD}_t \| \}, \quad \text{where } x^{GD}_t = x^t - \alpha_t \nabla f(x^t). \]

projection of \( x^{GD}_t \) onto \( C \)

\( x - \alpha f'(x) \)
Last Time: Projected-Gradient

\[ x^{t+1} = \underset{y \in C}{\text{argmin}} \{ \|y - x^{GD}_t\| \}, \text{ where } x^{GD}_t = x^t - \alpha_t \nabla f(x^t). \]

projection of \(x^{GD}_t\) onto \(C\),

\[ f(x) \]

Feasible Set

\[ x \]

\[ x_1 \]

\[ x_2 \]
Last Time: Projected-Gradient

\[ x^{t+1} = \arg\min_{y \in C} \{ \|y - x^G_{tD}\| \}, \text{ where } x^G_{tD} = x^t - \alpha_t \nabla f(x^t). \]

projection of \( x^G_{tD} \) onto \( C \), where \( x^G_{tD} = x^t - \alpha_t \nabla f(x^t) \).
We can convert non-smooth problem into smooth problems with simple constraints:

But transforming might make problem harder:
  - E.g., transformed problems often lose strong-convexity.

Can we apply a method like projected-gradient to the original problem?
Gradient Method

- We want to solve a smooth optimization problem:
  \[
  \arg\min_{x \in \mathbb{R}^d} f(x).
  \]

- Iteration \( x^t \) minimizes with quadratic approximation to ‘f’:
  \[
  f(y) \approx f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{L}{2} \|y - x^t\|^2,
  \]
  \[
  x^{t+1} = \arg\min_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{L}{2} \|y - x^t\|^2 \right\}.
  \]
Gradient Method

We want to solve a smooth optimization problem:

$$\arg\min_{x \in \mathbb{R}^d} f(x).$$

Iteration $x^t$ minimizes with quadratic approximation to $f'$:

$$f(y) \approx f(x^t) + \nabla f(x^t)^T(y - x^t) + \frac{L}{2}\|y - x^t\|^2,$$

$$x^{t+1} = \arg\min_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)^T(y - x^t) + \frac{L}{2}\|y - x^t\|^2 \right\}.$$

We can equivalently write this as the quadratic optimization:

$$x^{t+1} = \arg\min_{y \in \mathbb{R}^d} \left\{ \frac{1}{2}\|y - (x^t - \alpha_t \nabla f(x^t))\|^2 \right\},$$

and the solution is the gradient algorithm:

$$x^{t+1} = x^t - \alpha_t \nabla f(x^t).$$
Proximal-Gradient Method

- We want to solve a smooth plus non-smooth optimization problem:
  \[
  \arg\min_{x \in \mathbb{R}^d} f(x) + r(x).
  \]

- Iteration \( x^t \) minimizes with quadratic approximation to ‘\( f \)’:
  \[
  f(y) \approx f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{L}{2} \| y - x^t \|^2,
  \]
  \[
  x^{t+1} = \arg\min_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{L}{2} \| y - x^t \|^2 \right\}.
  \]
  We can equivalently write this as the quadratic optimization:
  \[
  x^{t+1} = \arg\min_{y \in \mathbb{R}^d} \left\{ \frac{1}{2} \| y - (x^t - \alpha_t \nabla f(x^t)) \|^2 \right\},
  \]
  and the solution is the gradient algorithm:
  \[
  x^{t+1} = x^t - \alpha_t \nabla f(x^t).
  \]
Projective-Gradient Proximal-Gradient Other Proximal Methods

Proximal-Gradient Method

- We want to solve a smooth plus non-smooth optimization problem:

\[
\arg\min_{x \in \mathbb{R}^d} f(x) + r(x).
\]

- Iteration \(x^t\) minimizes with quadratic approximation to \('f'\):

\[
\begin{align*}
  f(y) + r(y) &\approx f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{L}{2} \|y - x^t\|^2 + r(y), \\
  x^{t+1} &= \arg\min_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{L}{2} \|y - x^t\|^2 + r(y) \right\}.
\end{align*}
\]

We can equivalently write this as the quadratic optimization:

\[
\begin{align*}
  x^{t+1} &= \arg\min_{y \in \mathbb{R}^d} \left\{ \frac{1}{2} \|y - (x^t - \alpha_t \nabla f(x^t))\|^2 \right\},
\end{align*}
\]

and the solution is the gradient algorithm:

\[
\begin{align*}
  x^{t+1} &= x^t - \alpha_t \nabla f(x^t).
\end{align*}
\]
Proximal-Gradient Method

- We want to solve a smooth plus non-smooth optimization problem:

\[
\arg\min_{x \in \mathbb{R}^d} f(x) + r(x).
\]

- Iteration \(x^t\) minimizes with quadratic approximation to ‘\(f'\) :

\[
f(y) + r(y) \approx f(x^t) + \nabla f(x^t)^T(y - x^t) + \frac{L}{2}\|y - x^t\|^2 + r(y),
\]

\[
x^{t+1} = \arg\min_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)^T(y - x^t) + \frac{L}{2}\|y - x^t\|^2 + r(y) \right\}.
\]

We can equivalently write this as the proximal optimization:

\[
x^{t+1} = \arg\min_{y \in \mathbb{R}^d} \left\{ \frac{1}{2}\|y - (x^t - \alpha_t \nabla f(x^t))\|^2 + \alpha_t r(y) \right\},
\]

and the solution is the gradient algorithm:

\[
x^{t+1} = x^t - \alpha_t \nabla f(x^t).
\]
Proximal-Gradient Method

- We want to solve a smooth plus non-smooth optimization problem:
  \[
  \arg\min_{x \in \mathbb{R}^d} f(x) + r(x).
  \]

- Iteration \( x^t \) minimizes with quadratic approximation to \( f' \):
  \[
  f(y) + r(y) \approx f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{L}{2} \|y - x^t\|^2 + r(y),
  \]

  \[
  x^{t+1} = \arg\min_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{L}{2} \|y - x^t\|^2 + r(y) \right\}.
  \]

We can equivalently write this as the proximal optimization:

\[
    x^{t+1} = \arg\min_{y \in \mathbb{R}^d} \left\{ \frac{1}{2} \|y - (x^t - \alpha_t \nabla f(x^t))\|^2 + \alpha_t r(y) \right\},
\]

and the solution is the proximal-gradient algorithm:

\[
    x^{t+1} = \text{prox}_{\alpha r}[x^t - \alpha_t \nabla f(x^t)].
\]
Proximal-Gradient Method

So proximal-gradient step takes the form:

\[ x_t^{GD} = x^t - \alpha_t \nabla f(x^t), \]

\[ x^{t+1} = \arg\min_{y \in \mathbb{R}^d} \left\{ \frac{1}{2} \| y - x_t^{GD} \|^2 + \alpha_t r(y) \right\}. \]

Second part is called the proximal operator with respect to \( \alpha_t r. \)

Convergence rates are still the same as for minimizing \( f \) alone:

- E.g, if \( \nabla f \) is \( L \)-Lipschitz, \( f \) is \( \mu \)-strongly convex. and \( g \) is convex, then

\[ F(x^t) - F(x^*) \leq \left(1 - \frac{\mu}{L}\right)^t \left[F(x^0) - F(x^*)\right], \]

where \( F(x) = f(x) + r(x). \)
Proximal Operator, Iterative Soft Thresholding

The **proximal operator** is the solution to

$$
\text{prox}_r[x] = \arg\min_{y \in \mathbb{R}^d} \frac{1}{2} \|y - x\|^2 + r(y).
$$
Proximal Operator, Iterative Soft Thresholding

- The **proximal operator** is the solution to

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

- If \( r(y) = \alpha_t \lambda \| y \|_1 \), proximal operator is **soft-threshold**:
  - Apply \( x_j = \text{sign}(x_j) \max\{0, |x_j| - \alpha_t \lambda\} \) element-wise.
  - E.g., if \( \alpha_t \lambda = 1 \):

<table>
<thead>
<tr>
<th>Input</th>
<th>Threshold</th>
<th>Soft-Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6715</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1.2075</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.7172</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.6302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.4889</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Proximal Operator, Iterative Soft Thresholding**

- The **proximal operator** is the solution to

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

- If $r(y) = \alpha_t \lambda \| y \|_1$, proximal operator is **soft-threshold**:
  - Apply $x_j = \text{sign}(x_j) \max\{0, |x_j| - \alpha_t \lambda\}$ element-wise.
  - E.g., if $\alpha_t \lambda = 1$:

<table>
<thead>
<tr>
<th>Input</th>
<th>Threshold</th>
<th>Soft-Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6715</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-1.2075</td>
<td>-1.2075</td>
<td></td>
</tr>
<tr>
<td>0.7172</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1.6302</td>
<td>1.6302</td>
<td></td>
</tr>
<tr>
<td>0.4889</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Proximal Operator, Iterative Soft Thresholding

- The **proximal operator** is the solution to

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

- If \( r(y) = \alpha_t \lambda \|y\|_1 \), proximal operator is **soft-threshold**:
  - Apply \( x_j = \text{sign}(x_j) \max\{0, |x_j| - \alpha_t \lambda\} \) element-wise.
  - E.g., if \( \alpha_t \lambda = 1 \):

<table>
<thead>
<tr>
<th>Input</th>
<th>Threshold</th>
<th>Soft-Threshold</th>
</tr>
</thead>
</table>
| \[
\begin{bmatrix}
0.6715 \\
-1.2075 \\
0.7172 \\
1.6302 \\
0.4889
\end{bmatrix}
\] | \[
\begin{bmatrix}
0 \\
-1.2075 \\
0 \\
1.6302 \\
0
\end{bmatrix}
\] | \[
\begin{bmatrix}
0 \\
-0.2075 \\
0 \\
0.6302 \\
0
\end{bmatrix}
\] |
Special case of Projected-Gradient Methods

- **Projected-gradient** methods are another special case:

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

  (indicator function for convex set $C$)
Special case of Projected-Gradient Methods

- Projected-gradient methods are another special case:

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

(indicator function for convex set \( C \))

gives

\[
x^{t+1} = \arg\min_{y \in \mathbb{R}^d} \frac{1}{2} \| y - x \|^2 + r(y) = \arg\min_{y \in C} \frac{1}{2} \| y - x \|^2 = \arg\min_{y \in C} \| y - x \|.
\]
Proximal-Gradient for Group L1-Regularization

- The proximal operator for L1-regularization,

\[
\arg\min_{y \in \mathbb{R}^d} \left\{ \frac{1}{2} \| y - x \|^2 + \alpha_t \lambda \| y \|_1 \right\},
\]

applies soft-threshold element-wise,

\[
x_j = \frac{x_j}{|x_j|} \max\{0, |x_j| - \alpha t \lambda\}.
\]

- The proximal operator for group L1-regularization,

\[
\arg\min_{y \in \mathbb{R}^d} \left\{ \frac{1}{2} \| y - x \|^2 + \alpha_t \lambda \sum_{g \in G} \| y \|_2 \right\},
\]

applies group soft-threshold group-wise,

\[
x_g = \frac{x_g}{\| x_g \|_2} \max\{0, \| x_g \|_2 - \alpha t \lambda\}.
\]
The proximal operator for L1-regularization,

\[
\argmin_{y \in \mathbb{R}^d} \left\{ \frac{1}{2} \| y - x \|^2 + \alpha_t \lambda \| y \|_1 \right\},
\]

applies soft-threshold element-wise,

\[
x_j = \frac{x_j}{|x_j|} \max\{0, |x_j| - \alpha_t \lambda\}.
\]

The proximal operator for group L1-regularization,

\[
\argmin_{y \in \mathbb{R}^d} \left\{ \frac{1}{2} \| y - x \|^2 + \alpha_t \lambda \sum_{g \in G} \| y \|_2 \right\},
\]

applies a group soft-threshold group-wise,

\[
x_g = \frac{x_g}{\| x_g \|_2} \max\{0, \| x_g \|_2 - \alpha_t \lambda\}.
\]
Exact Proximal-Gradient Methods

We can efficiently compute the proximity operator for:

1. L1-Regularization and most separable regularizers.
2. Group $\ell_1$-Regularization (disjoint) and most group-separable regularizers.
We can efficiently compute the proximity operator for:

1. L1-Regularization and most separable regularizers.
2. Group $\ell_1$-Regularization (disjoint) and most group-separable regularizers.
3. Lower and upper bounds.
4. Small number of linear constraint.
5. Probability constraints.
7. A few other simple regularizers/constraints.
Exact Proximal-Gradient Methods

- We can efficiently compute the proximity operator for:
  1. L1-Regularization and most separable regularizers.
  2. Group $\ell_1$-Regularization (disjoint) and most group-separable regularizers.
  3. Lower and upper bounds.
  4. Small number of linear constraint.
  5. Probability constraints.
  7. A few other simple regularizers/constraints.

- Can solve these non-smooth problems as fast as smooth problems.
- But what if we can’t efficiently compute proximal operator?
Inexact Proximal-Gradient Methods

- We can efficiently approximate the proximal operator for:
  - Overlapping group L1-regularization.
  - Total-variation regularization.
  - Nuclear-norm regularization.
  - Sums of ‘simple’ functions (proximal-Dykstra).
Inexact Proximal-Gradient Methods

- We can efficiently **approximate** the proximal operator for:
  - Overlapping group L1-regularization.
  - Total-variation regularization.
  - Nuclear-norm regularization.
  - Sums of ‘simple’ functions (proximal-Dykstra).

- **Inexact proximal-gradient** methods:
  - Use an approximation to the proximal operator.
  - If approximation error decreases fast enough, same convergence rate:
    - To get $O(\rho^t)$ rate, error must be in $o(\rho^t)$. 
Discussion of Proximal-Gradient

- Solution $x^*$ is a fixed-point:

$$x^* = \text{prox}_{\alpha r}[x^* - \alpha f(x^*)], \text{ for any } \alpha.$$
Discussion of Proximal-Gradient

- Solution $x^*$ is a fixed-point:
  \[ x^* = \text{prox}_{\alpha r}[x^* - \alpha f(x^*)], \text{ for any } \alpha. \]

- With $\alpha_t < 2/L$, guaranteed to decrease objective.
  - Can still use adaptive step-size to estimate ‘L’.
Discussion of Proximal-Gradient

- Solution $x^*$ is a fixed-point:
  \[
  x^* = \text{prox}_{\alpha r}[x^* - \alpha f(x^*)], \text{ for any } \alpha.
  \]

- With $\alpha_t < 2/L$, guaranteed to decrease objective.
  - Can still use adaptive step-size to estimate ‘L’.
- With any $\alpha_t$, proximal–gradient generates a feasible descent direction:
  - If $\bar{x}^t = \text{prox}_{\alpha_t r}[x^t - \alpha_t \nabla f(x^t)]$, then the step
    \[
    x^{t+1} = x^t + \gamma_t (\bar{x}^t - x^t),
    \]
    decreases $f$ and satisfies constraints for $\gamma_t$ small enough.
Discussion of Proximal-Gradient

- Solution $x^*$ is a fixed-point:
  \[ x^* = \text{prox}_{\alpha r}[x^* - \alpha f(x^*)], \text{ for any } \alpha. \]

- With $\alpha_t < 2/L$, guaranteed to decrease objective.
  - Can still use adaptive step-size to estimate ‘L’.

- With any $\alpha_t$, proximal–gradient generates a feasible descent direction:
  - If $\bar{x}^t = \text{prox}_{\alpha_t r}[x^t - \alpha_t \nabla f(x^t)]$, then the step
    \[ x^{t+1} = x^t + \gamma_t (\bar{x}^t - x^t), \]
  decreases $f$ and satisfies constraints for $\gamma_t$ small enough.

- If proximal operator is expensive, can do Armijo line-search for $\gamma_t$ instead of $\alpha_t$:
  - Fix $\alpha_t$ and decrease $\gamma_t$: “backtracking along the feasible direction”.
    - Iterations tend to be in interior.
  - Fix $\gamma_t$ and decrease $\alpha_t$: “backtracking along the projection arc”.
    - Iterations tend to be at boundary.
Faster Proximal-Gradient Methods

- **Accelerated** proximal-gradient method:

\[
x^{t+1} = \operatorname{prox}_{\alpha_t r}[y^t - \alpha_t \nabla f(x^t)]
\]

\[
y^{t+1} = x^t + \beta_t (x^{t+1} - x^t).
\]

- Convergence properties same as smooth version.
Faster Proximal-Gradient Methods

- **Accelerated** proximal-gradient method:

  \[ x^{t+1} = \text{prox}_{\alpha_t r}[y^t - \alpha_t \nabla f(x^t)] \]

  \[ y^{t+1} = x^t + \beta_t (x^{t+1} - x^t) . \]

- Convergence properties same as smooth version.

- The naive Newton-like methods,

  \[ x^{t+1} = \text{prox}_{\alpha r}[x^t - \alpha_t d^t] , \text{ where } d^t \text{ solves } \nabla^2 f(x^t)d^t = \nabla f(x^t) , \]

  does NOT work.
Naive Projected-Newton

The figure illustrates the Naive Projected-Newton method. The function $f(x)$ is shown as a contour plot, and the feasible set is represented by the blue area. The point $x$ is the current iterate, and $x^+$ is the projection of $x - \alpha f'(x)$ onto the feasible set. The line segment $x - \alpha f'(x)$ shows the direction of the descent step, and the projection onto the feasible set is shown as $x^+$. The process iterates until a point within the feasible set is found where the gradient is sufficiently small, indicating a local minimum.
Naive Projected-Newton
Naive Projected-Newton

\[ f(x) \]

Feasible Set

\[ x, x_1, x_2 \]

\[ x - \alpha f'(x) \]

\[ x^k - \alpha H^{-1}f'(x) \]

\[ Q(x) \]
Naive Projected-Newton

\[ f(x) \]

Feasible Set

\[ x_1 \]

\[ x_2 \]

\[ x - \alpha f'(x) \]

\[ x^k - \alpha H^{-1} f'(x) \]

\[ x^+ \]
Naive Projected-Newton

\[ f(x) \]\n
Feasible Set

\[ x^k - \alpha H^{-1} f'(x) \]

\[ x - \alpha f'(x) \]

\[ x^+ \]

\[ x \]

\[ x^+ \]

\[ x_1 \]

\[ x_2 \]
Projected-Newton Method

- Projected-gradient minimizes quadratic approximation,

\[ x^{t+1} = \arg\min_{y \in C} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{1}{2\alpha_t} \| y - x^t \|^2 \right\}. \]
Projected-Newton Method

- Projected-gradient minimizes quadratic approximation,

\[ x^{t+1} = \arg\min_{y \in C} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{1}{2\alpha_t} \|y - x^t\|^2 \right\}. \]

- Newton’s method can be viewed as quadratic approximation (with \( H^t \approx \nabla^2 f(x^t) \)):

\[ x^{t+1} = \arg\min_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{1}{2\alpha_t} (y - x^t)H^t(y - x^t) \right\}. \]
Projected-Newton Method

- Projected-gradient minimizes quadratic approximation,

\[ x^{t+1} = \arg\min_{y \in C} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{1}{2\alpha_t} \|y - x^t\|^2 \right\}. \]

- Newton’s method can be viewed as quadratic approximation (wth \( H^t \approx \nabla^2 f(x^t) \)):

\[ x^{t+1} = \arg\min_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{1}{2\alpha_t} (y - x^t)H^t(y - x^t) \right\}. \]

- Projected Newton minimizes constrained quadratic approximation:

\[ x^{t+1} = \arg\min_{y \in C} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{1}{2\alpha_t} (y - x^t)H^t(y - x^t) \right\}. \]
Projected-Newton Method

- Projected-gradient minimizes quadratic approximation,
  \[ x^{t+1} = \arg\min_{y \in C} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{1}{2\alpha_t} \|y - x^t\|^2 \right\}. \]

- Newton’s method can be viewed as quadratic approximation (with \( H^t \approx \nabla^2 f(x^t) \)):
  \[ x^{t+1} = \arg\min_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{1}{2\alpha_t} (y - x^t)H^t(y - x^t) \right\}. \]

- Projected Newton minimizes constrained quadratic approximation:
  \[ x^{t+1} = \arg\min_{y \in C} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{1}{2\alpha_t} (y - x^t)H^t(y - x^t) \right\}. \]

- Equivalently, we project Newton step under different Hessian-defined norm,
  \[ x^{t+1} = \arg\min_{y \in C} \|y - (x^t - \alpha_t[H^t]^{-1}\nabla f(x^t))\|_{H^t}, \]

where general “quadratic norm” is \( \|z\|_A = \sqrt{z^TAz} \) for \( A > 0 \).
Discussion of Proximal-Newton

- **Proximal-Newton** is defined similarly,

\[
    x^{t+1} = \operatorname{argmin}_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{L}{2}(y - x^t)H^t(y - x^t) + r(y) \right\}.
\]

- But this is expensive even when \( r \) is simple.
Discussion of Proximal-Newton

- **Proximal-Newton** is defined similarly,

\[ x^{t+1} = \arg\min_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{L}{2} (y - x^t)H^t(y - x^t) + r(y) \right\}. \]

- But this is expensive even when \( r \) is simple.
- There are a variety of practical ways to approximate this:
  - Use Barzilai-Borwein or diagonal \( H^t \).
  - Two-metric projection: special method for separable \( r \).
  - **Inexact proximal-Newton**: solve the above approximately.
    - Useful when \( f \) is very expensive but \( r \) is simple.
    - “Costly functions with simple regularizers”.
Alternating Direction Method of Multipliers

- Alternating direction method of multipliers (ADMM) solves:

\[
\min_{Ax + By = c} \; f(x) + r(y).
\]

- Alternate between prox-like operators with respect to \( f \) and \( r \).
Alternating Direction Method of Multipliers

- Alternating direction method of multipliers (ADMM) solves:
  
  \[
  \min_{Ax + By = c} f(x) + r(y).
  \]

- Alternate between prox-like operators with respect to \( f \) and \( r \).
- Can introduce constraints to convert to this form:

  \[
  \min_x f(Ax) + r(x) \iff \min_{x = Ay} f(x) + r(y),
  \]
Alternating Direction Method of Multipliers

- **Alternating direction method of multipliers (ADMM)** solves:

  \[
  \min_{Ax+By=c} f(x) + r(y).
  \]

- Alternate between prox-like operators with respect to \(f\) and \(r\).
- Can **introduce constraints** to convert to this form:

  \[
  \min_x f(Ax) + r(x) \iff \min_{x=Ay} f(x) + r(y),
  \]

  \[
  \min_x f(x) + r(Bx) \iff \min_{y=Ax} f(x) + r(y).
  \]
Alternating Direction Method of Multipliers

- **Alternating direction method of multipliers (ADMM)** solves:
  \[
  \min_{Ax+By=c} f(x) + r(y).
  \]

- Alternate between prox-like operators with respect to \(f\) and \(r\).
- Can **introduce constraints** to convert to this form:
  \[
  \min_x f(Ax) + r(x) \iff \min_y f(x) + r(y),
  \]
  \[
  \min_x f(x) + r(Bx) \iff \min_y f(x) + r(y).
  \]

- If prox can not be computed exactly: **Linearized ADMM.**
Frank-Wolfe Method

In some cases the projected gradient step

\[ x^{t+1} = \arg\min_{y \in C} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{1}{2\alpha_t} \|y - x^t\|^2 \right\}, \]

may be hard to compute.
Frank-Wolfe Method

- In some cases the projected gradient step

\[
x^{t+1} = \arg\min_{y \in C} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{1}{2\alpha_t} \|y - x^t\|^2 \right\},
\]

may be hard to compute.

- Frank-Wolfe method simply uses:

\[
x^{t+1} = \arg\min_{y \in C} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) \right\},
\]

requires compact \( C \), takes convex combination of \( x^t \) and \( x^{t+1} \).

- \( O(1/t) \) rate for smooth convex objectives, some linear convergence results for strongly-convex [Jaggi, 2013].
Summary

- No black-box method can beat subgradient methods
- For most objectives, you can beat subgradient methods.
No black-box method can beat subgradient methods

For most objectives, you can beat subgradient methods.

You just need a long list of tricks:

- Smoothing.
- Chambolle-Pock.
- Projected-gradient.
- Two-metric projection.
- Proximal-gradient.
- Proximal-Newton.
- ADMM
- Frank-Wolfe.
- Mirror descent.
- Incremental surrogate optimization.
- Solving smooth dual.
Summary

- **Group L1-Regularization**: encourages sparsity in variable groups.
- **Structured sparsity**: encourages other patterns in variables.
- **Projected-Gradient**: allows optimization with simple constraints.
- **Proximal-gradient**: linear rates for sum of smooth and non-smooth.
- **Proximal-Newton**: even faster rates in special cases.

- Next time: faster stochastic methods, and kernels for exponential/infinite bases.