CPSC 540: Machine Learning
Group L1-Regularization, Proximal-Gradient

Mark Schmidt

University of British Columbia

Winter 2017
Assignment 1:
- 1 late day to hand it in tonight.
- 2 late days to hand it in next Monday.

Assignment 2:
- Out soon.
- Due February 6.
We discussed the convex optimization zoo:

- Iteration complexity of algorithms under different assumptions.

<table>
<thead>
<tr>
<th>Assumption</th>
<th>Algorithm</th>
<th>Convex</th>
<th>Strongly-Convex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subgradient bounded</td>
<td>Subgradient</td>
<td>$O\left(\frac{1}{\epsilon^2}\right)$</td>
<td>$O\left(\frac{1}{\epsilon}\right)$</td>
</tr>
<tr>
<td>Gradient is Lipschitz</td>
<td>Gradient</td>
<td>$O\left(\frac{1}{\epsilon}\right)$</td>
<td>$O\left(\frac{L}{\mu} \log\left(\frac{1}{\epsilon}\right)\right)$</td>
</tr>
<tr>
<td>Gradient is Lipschitz</td>
<td>Nesterov</td>
<td>$O\left(\frac{1}{\sqrt{\epsilon}}\right)$</td>
<td>$O\left(\sqrt{\frac{L}{\mu}} \log\left(\frac{1}{\epsilon}\right)\right)$</td>
</tr>
</tbody>
</table>

- **Smoothing** gets faster rate only if you use Nesterov-style algorithms.

- **Asymptotically-Newton** methods get superlinear convergence.
  - Assuming strong-convexity, gradient is Lipschitz, and Hessian is Lipschitz.
  - Not achieved by $O(d)$ time/space practical methods.
Last Time: Weaker Conditions for Linear Convergence

- We argued gradient descent converges linearly under **weaker assumptions**.
  - No need to know $L$, it holds for various step-size strategies.
- No need for “strong-smoothness.”
  - Just need Lipschitz-continuous gradient,
    \[ \| \nabla f(x) - \nabla f(y) \| \leq L \| x - y \|. \]
  - Or just for all $t$ and some $L$ that
    \[ L[f(x^{t+1}) - f(x^t)] \leq -\frac{1}{2} \| \nabla f(x^t) \|^2. \]
- No need for “strong-convexity”, we just need the **PL inequality**, 
  \[ \mu[f(x) - f^*] \leq \frac{1}{2} \| \nabla f(x) \|^2, \]
  or if $f$ is convex we **can make it strongly-convex by adding** L2-regularization.
Last Time: L1-Regularization

- We considered regularization by the $L1$-norm,
  \[
  \arg\min_{x \in \mathbb{R}^d} g(x) + \lambda \|x\|_1.
  \]
  - Encourages solution $x^*$ to be sparse.

- Convex approach to regularization and pruning irrelevant features.
  - Not perfect, but very fast.
  - Could be used as filter, or to initialize NP-hard solver.

- Non-smooth, but non-smooth part is separable,
  \[
  \lambda \|x\|_1 = \sum_{j=1}^{d} \lambda |x_j| = \sum_{j=1}^{d} h_j(x_j).
  \]
  which allows coordinate optimization.
Last Time: Coordinate Optimization

- In coordinate optimization each iteration $t$ only updates one variable.

- More efficient than gradient descent if the iterations are $d$-times cheaper.

- This holds for the problem class

$$f(x) = g(Ax) + \sum_{j=1}^{d} h_j(x_j) + \sum_{i=1}^{d} \sum_{j=1}^{d} g_{ij}(x_i, x_j),$$

for smooth $g$ and $g_{ij}$ (and where $g$ costs $O(n)$).

- We usually analyze it assuming partial derivatives are Lipschitz,

$$|\nabla_j f(x) - \nabla_j f(y)| \leq L|x_j - y_j|,$$

for some $L$ whenever $x$ and $y$ only differ in coordinate $j$.

- This is often easier to compute than $L$ for the full gradient.
Convergence Rate of Randomized Coordinate Optimization

- Last time we analyzed coordinate optimization assuming that:
  - Partial derivative are Lipschitz and $f$ satisfies PL inequality.
  - We choose coordinate to update $j_t$ uniformly at random.
  - Given $j_t$, we take a gradient step on $x_{j_t}$ with step-size $\alpha_t = 1/L$.
- We showed that this leads to the bound
  \[
  \mathbb{E}[f(x^{t+1})] - f(x^*) \leq \left(1 - \frac{\mu}{dL}\right) [f(x^t) - f(x^*)].
  \]
- By recursing we get linear convergence rate,
  \[
  \mathbb{E}[\mathbb{E}[f(x^{t+1})]] - f(x^*) \leq \mathbb{E}\left[\left(1 - \frac{\mu}{dL}\right) [f(x^t) - f(x^*)]\right] \quad \text{(expectation wrt $j_{t-1}$)}
  \]
  \[
  \mathbb{E}[f(x^{t+1})] - f(x^*) \leq \left(1 - \frac{\mu}{dL}\right) \mathbb{E}[f(x^t) - f(x^*)] \quad \text{(iterated expectation)}
  \]
  \[
  \leq \left(1 - \frac{\mu}{dL}\right)^2 [f(x^{t-1}) - f(x^*)]
  \]
So our rate for coordinate optimization is

$$\mathbb{E}[f(x^t) - f(x^*)] \leq \left(1 - \frac{\mu}{dL}\right)^t [f(x^0) - f(x^*)],$$

which means we need $O\left(d\frac{L}{\mu} \log\left(\frac{1}{\epsilon}\right)\right)$ iterations.

Remember that gradient descent needs $O\left(\frac{L}{\mu} \log\left(\frac{1}{\epsilon}\right)\right)$ iterations.

So coordinate optimization is slower?

- Yes, but remember we'll assume coordinate optimization steps are $d$-times cheaper.
- So we should divide the coordinate optimization complexity by $d$. 
Randomized Coordinate Optimization vs. Gradient Descent

- So for problems where coordinate steps are \( d \)-times cheaper we have

\[
O\left(\frac{L}{\mu} \log\left(\frac{1}{\epsilon}\right)\right),
\]

for both algorithms in terms of gradient descent iteration costs.

- So why prefer coordinate optimization?

  - The Lipschitz constants are different.
    - Gradient descent uses \( L_f \) and coordinate optimization uses \( L_c \).
    - \( L_c \leq L_f \), so coordinate optimization is faster when steps are \( d \)-times cheaper.
Lipschitz Sampling

- Can we do better than choosing \( j_t \) uniformly at random?

- You can go faster if you have an \( L_j \) for each coordinate:

  \[
  |\nabla_j f(x + \gamma e_j) - \nabla_j f(x)| \leq L_j |\gamma|.
  \]

- Using \( L_{j_t} \) as the step-size and sampling \( j_t \) proportional to \( L_j \) gives

  \[
  \mathbb{E}[f(x^t)] - f(x^*) \leq \left(1 - \frac{\mu}{d\bar{L}}\right)^t [f(x^0) - f(x^*)],
  \]

  where \( \bar{L} \) as the average Lipschitz constant (previously we used the maximum \( L_j \)).

- There are also greedy selection rules...
Gauss-Southwell Selection Rule

- Our bound on the progress if we choose coordinate $j_t$ is

$$f(x^{t+1}) \leq f(x^t) - \frac{1}{2L}|\nabla_j f(x^t)|^2.$$

- The “best” $j_t$ according to the bound is

$$j_t \in \arg\max_j \{|\nabla_j f(x^t)|\},$$

which is called greedy selection or the Gauss-Southwell rule.
Gauss-Southwell Selection Rule

- Our bound on the progress if we choose coordinate $j_t$ is
  \[
  f(x^{t+1}) \leq f(x^t) - \frac{1}{2L} |\nabla_j f(x^t)|^2.
  \]

- The "best" $j_t$ according to the bound is
  \[
  j_t \in \arg\max_j \{|\nabla_j f(x^t)|\},
  \]
  which is called greedy selection or the Gauss-Southwell rule.

- This gives a faster rate than uniformly at random.
  - You can prove this using that $|\nabla_j f(x^t)| = \|\nabla f(x^t)\|_\infty$.
  - And measuring PL in the $\infty$-norm,
    \[
    \mu[f(x) - f(x^*)] \leq \frac{1}{2} \|f(x)\|_\infty^2.
    \]
  - But typically this can't be implemented $d$ times faster than gradient descent.
    - You need an extra sparsity condition.
Gauss-Southwell-Lipschitz

- Our bound on the progress with an $L_j$ for each coordinate is

$$f(x^{t+1}) \leq f(x^t) - \frac{1}{2L_{j_t}} |\nabla_{j_t} f(x^t)|^2.$$ 

- The best coordinate to update according to this bound is

$$j_t \in \arg\max_j \frac{|\nabla_j f(x^t)|^2}{L_j}$$

which is called the Gauss-Southwell-Lipschitz rule.

- This is the optimal update for quadratic functions.
Numerical Comparison of Coordinate Selection Rules

Comparison on problem where Gauss-Southwell has similar cost to random:
Group Sparsity Projected Gradient Proximal-Gradient

Coordinate Optimization for Non-Smooth Objectives

- Last time we considered problems of the form

\[
\text{argmin}_{x \in \mathbb{R}^d} \ g(x) + \sum_{j=1}^{d} h_j(x_j),
\]

which includes \textbf{L1-regularized least squares}.

- Let's assume that
  - \( g \) is coordinate-wise Lipschitz continuous and \( \mu \)-strongly convex.
  - \( h_j \) are general convex functions (could be non-smooth).
  - You do exact coordinate optimization.

- Then we can show that

\[
\mathbb{E}[f(x^t)] - f(x^*) \leq \left(1 - \frac{\mu}{dL}\right)^t [f(x^0) - f(x^*)],
\]

the same convergence linear rate as if the non-smooth \( h_j \) were not there.

(and faster than the sublinear \( O(1/\epsilon) \) for solving non-smooth strongly-convex problems)
Outline

1. Group Sparsity
2. Projected Gradient
3. Proximal-Gradient
Motivation for Group Sparsity

- Recall that multi-class logistic regression uses
  \[ \hat{y}^i = \arg\max_c \{ w_c^T x^i \}, \]
  where we have a parameter vector \( w_c \) for each class \( c \).
- We typically use softmax loss and write our parameters as a matrix,
  \[ W = \begin{bmatrix} w_1 & w_2 & w_3 & \cdots & w_k \end{bmatrix} \]
- Suppose we want to use L1-regularization for feature selection,
  \[ \arg\min_{W \in \mathbb{R}^{d \times k}} \left\{ f(W) \right\} + \lambda \sum_{c=1}^{k} \| w_c \|_1 \]
  - Unfortunately, setting elements of \( W \) to zero may not select features.
Motivation for Group Sparsity

- Suppose L1-regularization gives a sparse $W$ with a non-zero in each row:

$$W = \begin{bmatrix}
-0.83 & 0 & 0 & 0 \\
0 & 0 & 0.62 & 0 \\
0 & 0 & 0 & -0.06 \\
0 & 0.72 & 0 & 0
\end{bmatrix}.$$ 

- Even though it’s very sparse, it uses all features.
  - Feature 1 is used in $w_1$.
  - Feature 2 is used in $w_3$.
  - Feature 3 is used in $w_4$.
  - Feature 4 is used in $w_2$.

- The classifier multiplies feature $j$ by each value in row $j$.

- In order to remove a feature, we need its entire row to be zero.
Motivation for Group Sparsity

- What we want is **group sparsity**:

  \[
  W = \begin{bmatrix}
  -0.77 & 0.04 & -0.03 & -0.09 \\
  0 & 0 & 0 & 0 \\
  0.04 & -0.08 & 0.01 & -0.06 \\
  0 & 0 & 0 & 0
  \end{bmatrix}.
  \]

- Each row is a group, and we want **groups (rows)** of variables that have all zeroes.
  - If row \( j \) is zero, then \( x_j \) is not used by the model.

- Pattern arises in other settings where each row gives parameters for one feature:
  - Multiple regression, multi-label classification, and multi-task classification.
Motivation for Group Sparsity

- **Categorical features** are another setting where group sparsity is needed.

- Consider categorical features encoded as **binary indicator** features:

<table>
<thead>
<tr>
<th>City</th>
<th>Age</th>
<th>Vancouver</th>
<th>Burnaby</th>
<th>Surrey</th>
<th>Age ≤ 20</th>
<th>20 &lt; Age ≤ 30</th>
<th>Age &gt; 30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vancouver</td>
<td>22</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Burnaby</td>
<td>35</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Vancouver</td>
<td>28</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- A linear model would use

\[
\hat{y}^i = w_1 x_{\text{van}} + w_2 x_{\text{bur}} + w_3 x_{\text{sur}} + w_4 x_{\leq 20} + w_5 x_{21-30} + w_6 x_{>30}.
\]

- If we want feature selection of **original categorical variables**, we have 2 groups:
  - \( \{w_1, w_2, w_3\} \) correspond to “City” and \( \{w_4, w_5, w_6\} \) correspond to “Age”. 

---

(There is no need to summarize the content, as it is comprehensive within the text itself.)
Consider a problem with a set of disjoint groups $\mathcal{G}$.

For example, $\mathcal{G} = \{\{1, 2\}, \{3, 4\}\}$.

Minimizing a function $f$ with group L1-regularization:

$$\arg\min_{w \in \mathbb{R}^d} f(w) + \lambda \sum_{g \in \mathcal{G}} \|w_g\|_p,$$

where $g$ refers to individual group indices and $\| \cdot \|_p$ is some norm.

For certain norms, it encourages sparsity in terms of groups $g$.

Variables $x_1$ and $x_2$ will either be both zero or both non-zero.

Variables $x_3$ and $x_4$ will either be both zero or both non-zero.
Group L1-Regularization

Why is it called group L1-regularization?

Consider $G = \{\{1, 2\}, \{3, 4\}\}$ and using L2-norm,

$$
\sum_{g \in G} \|x_g\|_2 = \sqrt{x_1^2 + x_2^2} + \sqrt{x_3^2 + x_4^2}.
$$

If vector $v$ contains the group norms, it’s the L1-norm of $v$:

$$
\text{If } v \equiv \begin{bmatrix} \|x_{12}\|_2 \\ \|x_{34}\|_2 \end{bmatrix} \text{ then } \sum_{g \in G} \|x_g\|_2 = \|x_{12}\|_2 + \|x_{34}\|_2 = v_1 + v_2 = |v_1| + |v_2| = \|v\|_1.
$$

So L1-regularization encourages sparsity in the group norms.

- When the norm of the group is 0, all group elements are 0.
Group L1-Regularization: Choice of Norm

- The group L1-regularizer is sometimes written as a “mixed” norm,
  \[ \| w \|_{1,p} \triangleq \sum_{g \in G} \| w_g \|_p. \]

- The most common choice for the norm is the L2-norm:
  - If \( G = \{\{1, 2\}, \{3, 4\}\} \) we obtain
    \[ \| w \|_{1,2} = \sqrt{w_1^2 + w_2^2} + \sqrt{w_3^2 + w_4^2}. \]

- Another common choice is the L\( \infty \)-norm,
  \[ \| w \|_{1,\infty} = \max\{\|w_1\|, \|w_2\|\} + \max\{\|w_3\|, \|w_4\|\}. \]

- But note that the L1-norm does not give group sparsity,
  \[ \| w \|_{1,1} = |w_1| + |w_2| + |w_3| + |w_4| = \| w \|_1, \]
  as it’s equivalent to non-group L1 regularization.
Sparsity from the L2-Norm?

- Didn't we say sparsity comes from the L1-norm and not the L2-norm?
  - Yes, but we were using the squared L2-norm.

- Squared vs. non-squared L2-norm in 1D:
  
  ![Graphs](image)

  - Non-squared L2-norm is absolute value.
    - It will set $w = 0$ for some finite $\lambda$.

  - Squaring the L2-norm gives a smooth function and destroys sparsity.
Sparsity from the L2-Norm?

- Squared vs. non-squared L2-norm in 2D:

![Diagram showing the difference between squared and non-squared L2-norms](image)

- The squared L2-norm is smooth and has no sparsity.

- For some finite $\lambda$, non-squared L2-norm simultaneously sets all variables to zero.
L1-Regularization vs. L2-Regularization

- Last time we looked at sparsity using our constraint trick,

\[
\arg\min_{w \in \mathbb{R}^d} f(w) + \lambda \|w\|_p \quad \Leftrightarrow \quad \arg\min_{w \in \mathbb{R}^d, \tau \in \mathbb{R}} f(w) + \lambda \tau \text{ with } \tau \geq \|w\|_p.
\]

- Note that we’re also minimizing the radius \(\tau\).
  - If \(\tau\) shrinks to zero, all \(w\) are set to zero.
  - But if \(\tau\) is squared there is virtually no penalty for having \(\tau\) non-zero.
The regularization path is the set of $w$ values as $\lambda$ varies,

$$w^\lambda = \arg\min_{w \in \mathbb{R}^d} f(w) + \lambda r(w),$$

Squared L2-regularization path vs. L1-regularization path:

- With $r(w) = \|w\|^2$, each $w_j$ gets close to 0 but is never exactly 0.
- With $r(w) = \|w\|_1$, each $w_j$ gets set to exactly zero for a finite $\lambda$. 
L2^2 and L2 Regularization Paths

- The regularization path is the set of \( w \) values as \( \lambda \) varies,
  \[
  w^\lambda = \arg\min_{w \in \mathbb{R}^d} f(w) + \lambda r(w),
  \]

- Squared L2-regularization path vs. non-squared path:

  - With \( r(w) = \|w\|^2 \), each \( w_j \) gets close to 0 but is never exactly 0.
  - With \( r(w) = \|w\|_2 \), all \( w_j \) get set to exactly zero for same finite \( \lambda \).
Group L1-Regularization

- Minimizing a function $f$ with **group L1-regularization**, 

$$\text{argmin}_{w \in \mathbb{R}^d} f(w) + \lambda \|w\|_{1,p} \iff \text{argmin}_{w \in \mathbb{R}^d, \tau \in \mathbb{R}^{\mid G\mid}} f(w) + \lambda \sum_{g=1}^{\mid G\mid} \tau_g \text{ with } \tau_g \geq \|w\|_p.$$ 

- We’re minimizing $f(w)$ plus the radiuses $\tau_g$ for each group $g$.
  - If $\tau_g$ shrinks to zero, all $w_g$ are set to zero.
Group L1-Regularization Paths

- The regularization path for group L1-regularization for different $p$ values:

- With $p = 1$ there is no grouping effect.
- With $p = 2$ the groups become zero at the same time.
The regularization path for group L1-regularization for different $p$ values:

- With $p = 1$ there is **no grouping effect**.
- With $p = 2$ the groups become zero at the same time.
- With $p = \infty$ the groups converge to same magnitude which then goes to 0.
Outline

1. Group Sparsity
2. Projected Gradient
3. Proximal-Gradient
Solving Group L1-Regularization Problems

- The group L1-regularizer is non-differentiable for any norm.
- It’s also non-separable, so we can’t apply coordinate optimization.
  - You can do block coordinate optimization, but that won’t work for other problems.

- A different problem structure we can use is

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

that it’s the sum of a smooth function and a “simple” function.
  - We’ll define “simple” later, but simple functions can be non-smooth.

- We can efficiently solve such problems with proximal-gradient methods.
  - A generalization of projected gradient methods.
We used projected gradient in 340 for NMF to find non-negative solutions,

$$\arg\min_{x \geq 0} f(x).$$

In this case the algorithm has a simple form,

$$x^{t+1} = \max\{0, x^t - \alpha_t \nabla f(x^t)\},$$

where the $\max$ is taken element-wise.

- “Do a gradient descent step, set negative values to 0.”

An obvious algorithm to try, and works as well as unconstrained gradient descent.
Broken “Projected-Gradient” Algorithms

- Based on our intuition, maybe we can go faster using a Newton-like step,

\[ x^{t+1} = \max\{0, x^t - \alpha_t [\nabla^2 f(x^t)]^{-1} \nabla f(x^t)\}, \]

- We might also think that if we want \( x \) to be a probability

\[ \arg\min_{x \geq 0, \ 1^T x = 1} f(x), \]

we could take a gradient step, set negative values to zero, and divide by the sum.

- Both of the above algorithms will NOT work.
Optimization with Simple Constraints

- Recall that we can view gradient descent as a minimizing quadratic approximation

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

where we have a general step-size \( \alpha_t \) instead of \( 1/L \).

- Now we want to optimize \( x \) over some convex set \( C \),

\[ \arg\min_{x \in C} f(x). \]

- We could minimize quadratic approximation to \( f \) subject to the constraints,

\[ x^{t+1} \in \arg\min_{y \in C} \left\{ f(x^t) + \nabla f(x^t)^T(y - x^t) + \frac{1}{2\alpha_t} \|y - x^t\|^2 \right\}, \]
We can re-write this iteration as

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

\[ \equiv \arg\min_{y \in C} \left\{ \alpha_t f(x^t) + \alpha_t \nabla f(x^t)^T (y - x^t) + \frac{1}{2} \|y - x^t\|^2 \right\} \quad \text{(multiply by } \alpha_t) \]

\[ \equiv \arg\min_{y \in C} \left\{ \frac{\alpha_t^2}{2} \|\nabla f(x^t)\|^2 + \alpha_t \nabla f(x^t)^T (y - x^t) + \frac{1}{2} \|y - x^t\|^2 \right\} \quad \text{(add constant)} \]

\[ \equiv \arg\min_{y \in C} \left\{ \| (y - x^t) + \alpha_t \nabla f(x^t) \|^2 \right\} \quad \text{(complete the square)} \]

\[ \equiv \arg\min_{y \in C} \left\{ \|y - \left( x^t - \alpha_t \nabla f(x^t) \right) \|^2 \right\}, \quad \text{gradient descent} \]

and this is called the \textit{projected-gradient} algorithm.
We can view the projected-gradient algorithm as having two steps:

1. Perform an unconstrained gradient descent step,
\[ x^{t+\frac{1}{2}} = x^t - \alpha_t \nabla f(x^t). \]

2. Compute the projection onto the set \( \mathcal{C} \),
\[ x^{t+1} \in \arg\min_{y \in \mathcal{C}} \| y - x^{t+\frac{1}{2}} \| . \]

Projection is the closest point that satisfies the constraints.

- Generalizes “projection” from linear algebra.
- We’ll also write projection of \( x \) onto \( \mathcal{C} \) as
\[ \text{proj}_\mathcal{C}[x] = \arg\min_{y \in \mathcal{C}} \| y - x \| , \]

and for convex \( \mathcal{C} \) it’s unique.
Projected-Gradient

\[ x^{t+1} \in \operatorname{argmin}_{y \in C} \| y - x^{t+\frac{1}{2}} \|, \quad x^{t+\frac{1}{2}} = x^t - \alpha_t \nabla f(x^t). \]

\[ \text{projection} \]

\[ \text{gradient} \]
Projected-Gradient

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

projection

Feasible Set

\[ f(x) \]
Projected-Gradient

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

projection

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

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

projection
gradient
Convergence Rate of Projected Gradient

- **Iteration complexity of projection-gradient:**

<table>
<thead>
<tr>
<th>Assumption</th>
<th>Algorithm</th>
<th>Convex</th>
<th>Strongly-Convex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subgradient bounded</td>
<td>Subgradient</td>
<td>$O(1/\epsilon^2)$</td>
<td>$O(1/\epsilon)$</td>
</tr>
<tr>
<td>Gradient is Lipschitz</td>
<td>Gradient</td>
<td>$O(1/\epsilon)$</td>
<td>$O\left(\frac{L}{\mu} \log(1/\epsilon)\right)$</td>
</tr>
<tr>
<td>Gradient is Lipschitz</td>
<td>Nesterov</td>
<td>$O(1/\sqrt{\epsilon})$</td>
<td>$O\left(\sqrt{\frac{L}{\mu}} \log(1/\epsilon)\right)$</td>
</tr>
</tbody>
</table>

- These are the same rates we had for unconstrained optimization.

- **Other nice properties:**
  - With $\alpha_t < 2/L$, guaranteed to decrease objective.
  - For convex $f$ the only “fixed points” are optimal solutions,

  $$x^* = \text{proj}_C[x^* - \alpha \nabla f(x^*)],$$

  for any step-size $\alpha > 0$: 
Projecting gradient is only efficient if the projection is cheap.

We say that $C$ is simple if the projection is cheap.
- For example, if it costs $O(d)$ then it adds no cost to the algorithm.

For example, if want $x \geq 0$ then projection sets negative values to 0.
- Non-negative constraints are “simple”.

Another example if $x \geq 0$ and $x^T1 = 1$, the probability simplex.
- There are $O(d)$ algorithm to compute this projection.
Simple Convex Sets

- Other examples of simple convex sets:
  - Having upper and lower bounds on the variables, $LB \leq x \leq UB$.
  - Having a linear equality constraint, $a^T x = b$, or a small number of them.
  - Having a half-space constraint, $a^T x \leq b$, or a small number of them.
  - Having a norm-ball constraint, $\|x\|_p \leq \tau$, for $p = 1, 2, \infty$ (fixed $\tau$).
  - Having a norm-cone constraint, $\|x\|_p \leq \tau$, for $p = 1, 2, \infty$ (variable $\tau$).
Group L1-Regularization

- We can convert the non-smooth group L1-regularization problem,

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

into a smooth problem with simple constraints:

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

- Here the constraints are separable:
  - We can project onto each norm-cone separately.

- Since norm-cones are simple we can solve this with projected-gradient,
Faster Projected-Gradient Methods

- **Accelerated** projected-gradient method has the form
  \[ x^{t+1} = \text{proj}_C[y^t - \alpha_t \nabla f(x^t)] \]
  \[ y^{t+1} = x^t + \beta_t (x^{t+1} - x^t). \]

- We could alternately use the Barzilai-Borwein step-size.
  - Known as **spectral projected-gradient**.

- The naive Newton-like methods with Hessian approximation \( H_t \),
  \[ x^{t+1} = \text{proj}_C[x^t - \alpha_t [H_t]^{-1} \nabla f(x^t)], \]
  does NOT work.
Naive Projected-Newton
Naive Projected-Newton
Naive Projected-Newton

\[ f(x) \]

Feasible Set

\[ x_1 \]

\[ x_2 \]
Naive Projected-Newton

\[ f(x) \]

Feasible Set
Naive Projected-Newton

\[ f(x) \]

Feasible Set

\[ x - \alpha f'(x) \]
Naive Projected-Newton

\[ f(x) \]

Feasible Set

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

\[ x_1 \]

\[ x_2 \]
Naive Projected-Newton

\[ f(x) \]

Feasible Set

\[ x \]

\[ x_1 \]

\[ x_2 \]

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

\[ x^+ \]
Naive Projected-Newton

\[ f(x) \]

**Feasible Set**

\[ \mathbf{x} \rightarrow f'(x) \mathbf{x} + f(x) \]

\[ \mathbf{x} - \alpha f'(x) \mathbf{x} \]
Naive Projected-Newton

\[ f(x) \]

Feasible Set

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

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

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

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

Feasible Set

\[ f(x) \]

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

\[ f(x) \]

Feasible Set

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

\[ Q(x) \]

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

\[ x^+ \]

\[ x \]

\[ x^+ \]

\[ X_1 \]

\[ X_2 \]
Projected-Newton Method

- Projected-gradient minimizes quadratic approximation,

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

- 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 \( \|z\|_A = \sqrt{z^T A z} \) for \( A \succeq 0 \).
Discussion of Projected-Newton

- Projected-Newton iteration is given by

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

- But this is expensive even when \( C \) is simple.

There are a variety of practical alternatives:

- If \( H_t \) is diagonal then this is typically simple to solve.

- Two-metric projection methods are special algorithms for upper/lower bounds.
  - Fix problem of naive method in this case by making \( H_t \) partially diagonal.

- Inexact projected-Newton: solve the above approximately.
  - Useful when \( f \) is very expensive but \( H_t \) and \( C \) are simple.
  - “Costly functions with simple constraints”.


Outline

1 Group Sparsity
2 Projected Gradient
3 Proximal-Gradient
Should we use projected-gradient for non-smooth problems?

- We converted non-smooth problem into smooth with simple constraints.

- But transforming might make problem harder:
  - For L1-regularization least squares,
    \[
    \arg\min_{w \in \mathbb{R}^d} \frac{1}{2}\|Xw - y\|^2 + \lambda\|w\|_1,
    \]
    we can re-write as a smooth problem with bound constraints,
    \[
    \arg\min_{w_+ \geq 0, w_- \geq 0}\|X(w_+ - w_-) - y\|^2 + \lambda \sum_{j=1}^d (w_+ + w_-).
    \]
    - Transformed problem is not strongly convex even if the original was.

- **Proximal-gradient** methods apply to analogous non-smooth problems,
  \[
  \arg\min_{w \in \mathbb{R}^d} g(w) + r(w).
  \]
Gradient Method

- We want to solve a smooth optimization problem:

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

- Iteration \( x^t \) works with a quadratic approximation to \( f \):

  \[
  f(y) \approx f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{1}{2\alpha_t} \|y - x^t\|^2,
  \]

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

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

- Iteration \(x^t\) works with a quadratic approximation to \(f\):
  \[
  f(y) + r(y) \approx f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{1}{2\alpha_t} \|y - x^t\|^2 + r(y),
  \]
  \[
  x^{t+1} = \argmin_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{1}{2\alpha_t} \|y - x^t\|^2 + r(y) \right\}.
  \]

We can equivalently write this as the **proximal** optimization:

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

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:

\[
x_{t+\frac{1}{2}} = x^t - \alpha_t \nabla f(x^t)
\]

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

- 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 \(r\) is convex, then

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

where \(F(x) = f(x) + r(x)\).
Special case of Projected-Gradient Methods

- **Projected-gradient** methods are a 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 Operator, Iterative Soft Thresholding

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

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

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

<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>0</td>
</tr>
<tr>
<td>-1.2075</td>
<td>-1.2075</td>
<td>-0.2075</td>
</tr>
<tr>
<td>0.7172</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.6302</td>
<td>1.6302</td>
<td>0.6302</td>
</tr>
<tr>
<td>0.4889</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Has the nice property that iterations \( x^t \) are sparse.
Proximal-Gradient for L1-Regularization

- The proximal operator for L1-regularization when using step-size $\alpha_t$,

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

applies soft-threshold element-wise,

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

- $w_j$ with absolute values below $\alpha_t \lambda$ get set to 0.
- $w_j$ with absolute values above $\alpha_t \lambda$ get shrunk by $\alpha_t \lambda$. 
**Proximal-Gradient for Group L1-Regularization**

- The proximal operator for *group* L1-regularization,

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

applies a soft-threshold *group*-wise,

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

- So we can *solve* group L1-regularization problems as fast as smooth problems.
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 a soft-threshold group-wise,

$$
x_g = \frac{x_g}{\|x_g\|_2} \max\{0, \|x_g\|_2 - \alpha_t \lambda\}.
$$

So we can solve group L1-regularization problems as fast as smooth problems.
Proximal-Gradient for Group L1-Regularization

- 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 a soft-threshold group-wise,

$$x_g = \frac{x_g}{\|x_g\|_2} \max\{0, \|x_g\|_2 - \alpha_t \lambda\}.$$ 

So we can solve group L1-regularization problems as fast as smooth problems.
Summary

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

Next time: what if the number of training examples $n$ is huge?