# CPSC 540: Machine Learning

## Proximal-Gradient

Mark Schmidt

University of British Columbia

Winter 2019

# Last Time: Projected-Gradient

- We discussed minimizing smooth functions with simple convex constraints,

$$\underset{w \in \mathcal{C}}{\operatorname{argmin}}\, f(w).$$

- For example, we could be solving a non-negative least squares problem,

$$\underset{w \geq 0}{\operatorname{argmin}}\, \frac{1}{2}\|Xw - y\|^2.$$

- With "simple" constraints like this, we can use projected-gradient:

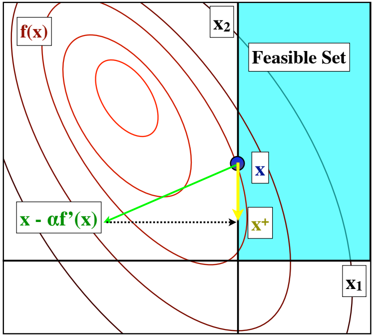
$$w^{k+\frac{1}{2}} = w^k - \alpha_k \nabla f(w^k) \qquad \text{(gradient step)}$$

$$w^{k+1} = \underset{v \in \mathcal{C}}{\operatorname{argmin}}\, \|v - w^{k+\frac{1}{2}}\| \qquad \text{(projection)}$$

# Last Time: Projected-Gradient

$$w^{k+\frac{1}{2}} = w^k - \alpha_k \nabla f(w^k) \qquad \text{(gradient step based on function } f\text{)}$$

$$w^{k+1} = \underset{v \in \mathcal{C}}{\text{argmin}} \|v - w^{k+\frac{1}{2}}\| \qquad \text{(projection onto feasible set } \mathcal{C}\text{)}$$



# Projected-Gradient

- We can view the projected-gradient algorithm as having two steps:
  1. Perform an unconstrained gradient descent step,
  $$w^{k+\frac{1}{2}} = w^k - \alpha_k \nabla f(w^k).$$
  2. Compute the projection onto the set $\mathcal{C}$,
  $$w^{k+1} \in \operatorname*{argmin}_{v \in \mathcal{C}} \|v - w^{k+\frac{1}{2}}\|.$$
- Projection is the closest point that satisfies the constraints.
  - Generalizes "projection onto subspace" from linear algebra.
  - We'll also write projection of $w$ onto $\mathcal{C}$ as
  $$\text{proj}_{\mathcal{C}}[w] = \operatorname*{argmin}_{v \in \mathcal{C}} \|v - w\|,$$
  and for convex $\mathcal{C}$ it's unique.

# Convergence Rate of Projected Gradient

- Projected versions have same complexity as unconstrained versions:

| Assumption | Proj(Grad) | Proj(Subgrad) | Quantity |
|---|---|---|---|
| Convex | $O(1/\epsilon)$ | $O(1/\epsilon^2)$ | $f(w^t) - f^* \leq \epsilon$ |
| Strongly-Convex | $O(\log(1/\epsilon))$ | $O(1/\epsilon)$ | $f(w^t) - f^* \leq \epsilon$ |

- Nice properties in the smooth case:
  - With $\alpha_t < 2/L$, guaranteed to decrease objective.
  - There exist practical step-size strategies as with gradient descent (bonus).
  - For convex $f$ a $w^*$ is optimal iff it's a "fixed point" of the update,

$$w^* = \text{proj}_{\mathcal{C}}[w^* - \alpha \nabla f(w^*)],$$

    for any step-size $\alpha > 0$.

- There exist accelerated versions and Newton-like versions (bonus slides).
  - Acceleration is an obvious modification, Newton is more complicated.

# Why the Projected Gradient?

- We want to optimize $f$ (smooth but possibly non-convex) over some convex set $\mathcal{C}$,

$$\operatorname*{argmin}_{w \in \mathcal{C}} f(w).$$

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

$$w^{k+1} \in \operatorname*{argmin}_{v} \left\{ f(w^k) + \nabla f(w^k)(v - w^k) + \frac{1}{2\alpha_k}\|v - w^k\|^2 \right\},$$

where we've written it with a general step-size $\alpha_k$ instead of $1/L$.
  - Solving the convex quadratic argmin gives $w^{k+1} = w^k - \alpha_k \nabla f(w^k)$.

- We could minimize quadratic approximation to $f$ subject to the constraints,

$$w^{k+1} \in \operatorname*{argmin}_{v \in \mathcal{C}} \left\{ f(w^k) + \nabla f(w^k)^\top (v - w^k) + \frac{1}{2\alpha_k}\|v - w^k\|^2 \right\},$$

# Why the Projected Gradient?

- We write this "minimize quadratic approximation over the set $\mathcal{C}$" iteration as

$$
\begin{aligned}
w^{k+1} &\in \operatorname*{argmin}_{v \in \mathcal{C}} \left\{ f(w^k) + \nabla f(w^k)^\top (v - w^k) + \frac{1}{2\alpha_k}\|v - w^k\|^2 \right\} \\
&\equiv \operatorname*{argmin}_{v \in \mathcal{C}} \left\{ \alpha_k f(w^k) + \alpha_k \nabla f(w^k)^\top (v - w^k) + \frac{1}{2}\|v - w^k\|^2 \right\} \quad \text{(multiply by } \alpha_k\text{)} \\
&\equiv \operatorname*{argmin}_{v \in \mathcal{C}} \left\{ \frac{\alpha_k^2}{2}\|\nabla f(w^k)\|^2 + \alpha_k \nabla f(w^k)^\top (v - w^k) + \frac{1}{2}\|v - w^k\|^2 \right\} \quad (\pm \text{ const.}) \\
&\equiv \operatorname*{argmin}_{v \in \mathcal{C}} \left\{ \|(v - w^k) + \alpha_k \nabla f(w^k)\|^2 \right\} \quad \text{(complete the square)} \\
&\equiv \operatorname*{argmin}_{v \in \mathcal{C}} \left\{ \|v - \underbrace{(w^k - \alpha_k \nabla f(w^k))}_{\text{gradient descent}}\| \right\},
\end{aligned}
$$

which gives the projected-gradient algorithm: $w^{k+1} = \text{proj}_{\mathcal{C}}[w^k - \alpha_k \nabla f(w^k)]$.

# Simple Convex Sets

- Projected-gradient is only efficient if the projection is cheap.
- We say that $\mathcal{C}$ 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 we want $w \geq 0$ then projection sets negative values to 0.
  - Non-negative constraints are "simple".
- Another example is $w \geq 0$ and $w^\top 1 = 1$, the probability simplex.
  - There are $O(d)$ algorithm to compute this projection (similar to "select" algorithm)

# 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, $a^\top x = b$, or a small number of them.
  - Having a half-space constraint, $a^\top x \leq b$, 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 $\tau$).
- It's easy to minimize smooth functions with these constraints.

# Intersection of Simple Convex Sets: Dykstra's Algorithm

- Often our set $\mathcal{C}$ is the intersection of simple convex set,

$$\mathcal{C} \equiv \cap_i \mathcal{C}_i.$$

- For example, we could have a large number linear constraints:

$$\mathcal{C} \equiv \{w \mid a_i^T w \leq b_i, \forall_i\}.$$

- Dykstra's algorithm can compute the projection in this case.
  - On each iteration, it projects a vector onto one of the sets $\mathcal{C}_i$.
  - Requires $O(\log(1/\epsilon))$ such projections to get within $\epsilon$.

(This is not the shortest path algorithm of "Dijkstra".)

# Outline

1 Proximal-Gradient

2 Group Sparsity

# Solving Problems with Simple Regularizers

- We were discussing how to solve non-smooth L1-regularized objectives like

$$\operatorname*{argmin}_{w\in\mathbb{R}^d} \frac{1}{2}\|Xw - y\|^2 + \lambda\|w\|_1.$$

- Use our trick to formulate as a quadratic program?
    - $O(d^2)$ or worse.
- Make a smooth approximation to the L1-norm?
    - Destroys sparsity (we'll again just have one subgradient at zero).
- Use a subgradient method?
    - Needs $O(1/\epsilon)$ iterations even in the strongly-convex case.
- Transform to "smooth $f$ with simple constraints" and use projected-gradient?
    - Works well (bonus), but increases problem size and destroys strong-convexity.

- For "simple" regularizers, proximal-gradient methods don't have these drawbacks

# Quadratic Approximation View of Gradient Method

- We want to solve a smooth optimization problem:

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} f(w).$$

- Iteration $w^k$ works with a quadratic approximation to $f$:

$$f(v) \approx f(w^k) + \nabla f(w^k)^\top (v - w^k) + \frac{1}{2\alpha_k}\|v - w^k\|^2,$$

$$w^{k+1} \in \operatorname*{argmin}_{v \in \mathbb{R}^d} \left\{ f(w^k) + \nabla f(w^k)^\top (v - w^k) + \frac{1}{2\alpha_k}\|v - w^k\|^2 \right\}.$$

We can equivalently write this as the quadratic optimization:

$$w^{k+1} \in \operatorname*{argmin}_{v \in \mathbb{R}^d} \left\{ \frac{1}{2}\|v - (w^k - \alpha_k \nabla f(w^k))\|^2 \right\},$$

and the solution is the gradient algorithm:

$$w^{k+1} = w^k - \alpha_k \nabla f(w^k).$$

# Quadratic Approximation View of Proximal-Gradient Method

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

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} f(w) + r(w).$$

- Iteration $w^k$ works with a quadratic approximation to $f$:

$$f(v) + r(v) \approx f(w^k) + \nabla f(w^k)^\top (v - w^k) + \frac{1}{2\alpha_k}\|v - w^k\|^2 + r(v),$$

$$w^{k+1} \in \operatorname*{argmin}_{v \in \mathbb{R}^d} \left\{ f(w^k) + \nabla f(w^k)^\top (v - w^k) + \frac{1}{2\alpha_k}\|v - w^k\|^2 + r(v) \right\}.$$

We can equivalently write this as the proximal optimization:

$$w^{k+1} \in \operatorname*{argmin}_{v \in \mathbb{R}^d} \left\{ \frac{1}{2}\|v - (w^k - \alpha_k \nabla f(w^k))\|^2 + \alpha_k r(v) \right\},$$

and the solution is the proximal-gradient algorithm:

$$w^{k+1} = \text{prox}_{\alpha_k r}[w^k - \alpha_k \nabla f(w^k)].$$

# Proximal-Gradient for L1-Regularization

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

$$\text{prox}_{\alpha_k\lambda\|\cdot\|_1}[w^{k+\frac{1}{2}}] \in \underset{v\in\mathbb{R}^d}{\text{argmin}}\left\{\frac{1}{2}\|v - w^{k+\frac{1}{2}}\|^2 + \alpha_k\lambda\|v\|_1\right\},$$

involves solving a simple (strongly-convex) 1D problem for each variable $j$:

$$w_j^{k+1} \in \underset{v_j\in\mathbb{R}}{\text{argmin}}\left\{\frac{1}{2}(v_j - w_j^{k+\frac{1}{2}})^2 + \alpha_k\lambda|v_j|\right\}.$$

- We can find the argmin by finding the unique $v_j$ with $0$ in the sub-differential.
- The solution is given by applying "soft-threshold" operation:
  1. If $|w_j^{k+\frac{1}{2}}| \leq \alpha_k\lambda$, set $w_j^{k+1} = 0$.
  2. Otherwise, shrink $|w_j^{k+\frac{1}{2}}|$ by $\alpha_k\lambda$.

# Proximal-Gradient for L1-Regularization

- An example sof-threshold operator with $\alpha_k \lambda = 1$:

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

- Symbolically, the soft-threshold operation computes

$$w_j^{k+1} = \underbrace{\text{sign}(w^{k+\frac{1}{2}})}_{-1 \text{ or } +1} \max\left\{0, |w_j^{k+\frac{1}{2}}| - \alpha_k \lambda\right\}.$$

- Has the nice property that iterations $w^k$ are sparse.
  - Compared to subgradient method which wouldn't give exact zeroes.

# Proximal-Gradient Method

- So proximal-gradient step takes the form:

$$w^{k+\frac{1}{2}} = w^k - \alpha_k \nabla f(w^k)$$
$$w^{k+1} = \operatorname*{argmin}_{v \in \mathbb{R}^d} \left\{ \frac{1}{2}\|v - w^{k+\frac{1}{2}}\|^2 + \alpha_k r(v) \right\}.$$

- Second part is called the proximal operator with respect to a convex $\alpha_k r$.
  - We say that $r$ is simple if you can efficiently compute proximal operator.

- Very similar properties to projected-gradient when $\nabla f$ is Lipschitz-continuous:
  - Guaranteed improvement for $\alpha < 2/L$, practical backtracking methods work better.
  - Solution is a fixed point, $w^* = \text{prox}_r[w^* - \nabla f(w^*)]$.
  - If $f$ is strongly-convex then

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

where $F(w) = f(w) + r(w)$.

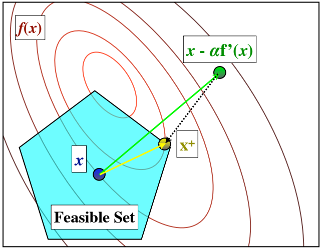
# Projected-Gradient is Special case of Proximal-Gradient

- Projected-gradient methods are a special case:

$$r(w) = \begin{cases} 0 & \text{if } w \in \mathcal{C} \\ \infty & \text{if } w \notin \mathcal{C} \end{cases}, \quad (\text{indicator function for convex set } \mathcal{C})$$

gives

$$w^{k+1} \in \underbrace{\mathop{\mathrm{argmin}}_{v \in \mathbb{R}^d} \frac{1}{2}\|v - w^{k+\frac{1}{2}}\|^2 + r(v)}_{\text{proximal operator}} \equiv \mathop{\mathrm{argmin}}_{v \in \mathcal{C}} \frac{1}{2}\|v - w^{k+\frac{1}{2}}\|^2 \equiv \underbrace{\mathop{\mathrm{argmin}}_{v \in \mathcal{C}} \|v - w^{k+\frac{1}{2}}\|}_{\text{projection}}.$$



# Proximal-Gradient Linear Convergence Rate

- Simplest linear convergence proofs are based on the proximal-PL inequality,

$$\frac{1}{2}\mathcal{D}_r(w, L) \geq \mu(F(w) - F^*),$$

where compared to PL inequality we've replaced $\|\nabla f(w)\|^2$ with

$$\mathcal{D}_r(w, \alpha) = -2\alpha \min_v \left[\nabla g(w)^\top (v - w) + \frac{\alpha}{2}\|v - w\|^2 + r(v) - r(w)\right],$$

and recall that $F(w) = f(w) + r(w)$ (bonus).

- This non-intuitive property holds for many important problems:
  - L1-regularized least squares.
  - Any time $f$ is strong-convex (i.e., add an L2-regularizer as part of $f$).
  - Any $f = g(Ax)$ for strongly-convex $g$ and $r$ being indicator for polyhedral set.
- But it can be painful to show that functions satisfy this property.

# Outline

1 Proximal-Gradient

2 Group Sparsity

# Motivation for Group Sparsity

- Recall that multi-class logistic regression uses
$$\hat{y}^i = \underset{c}{\text{argmax}}\{w_c^\top x^i\},$$
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,
$$\underset{W \in \mathbb{R}^{d \times k}}{\text{argmin}} \underbrace{f(W)}_{\text{softmax loss}} + \underbrace{\lambda \sum_{c=1}^{k} \|w_c\|_1}_{\text{L1-regularization}}.$$
- Unfortunately, setting elements of $W$ to zero may not select features.

# Motivation for Group Sparsity

- Suppose L1-regularizationgives 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.
  - Remember that classifier multiplies feature $j$ by each value in row $j$.
  - 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 $w_2$.
- 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 Sparsiy

- Categorical features are another setting where group sparsity is needed.
- Consider categorical features encoded as binary indicator features ("1 of $k$"):

| City | Age |
|---|---|
| Vancouver | 22 |
| Burnaby | 35 |
| Vancouver | 28 |

| Vancouver | Burnaby | Surrey | Age ≤ 20 | 20 < Age ≤ 30 | Age > 30 |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 |

- 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".

# Group L1-Regularization

- 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:

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} f(w) + \lambda \sum_{g \in \mathcal{G}} \|w_g\|_p,$$

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,

$$\sum_{g \in G} \|w_g\|_2 = \sqrt{w_1^2 + w_2^2} + \sqrt{w_3^2 + w_4^2}.$$

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

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

- So groups 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,

$$\|w\|_{1,p} \triangleq \sum_{g \in \mathcal{G}} \|w_g\|_p.$$

- The most common choice for the norm is the L2-norm:
  - If $\mathcal{G} = \{\{1,2\},\{3,4\}\}$ 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,

$$\|w\|_{1,\infty} = \max\{|w_1|,|w_2|\} + \max\{|w_3|,|w_4|\}.$$

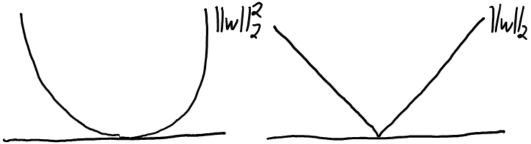
- But note that the L1-norm does not give group sparsity,

$$\|w\|_{1,1} = |w_1| + |w_2| + |w_3| + |w_4| = \|w\|_1,$$

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:




- Non-squared L2-norm is absolute value.
    - Non-squared L2-regularizer will set $w = 0$ for some finite $\lambda$.
- Squaring the L2-norm gives a smooth function but destroys sparsity.

# Sparsity from the L2-Norm?

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


- The squared L2-norm is smooth and has no sparsity.
- Non-squared L2-norm is non-smooth at the zero vector.
  - It doesn't encourage us to set any $w_j = 0$ as long as one $w_{j'} \neq 0$.
  - But if $\lambda$ is large enough it encourages all $w_j$ to be set to 0.

# Sub-differential of Group L1-Regularization

- For our group L1-regularization objective with the 2-norm,

$$F(w) = f(w) + \lambda \sum_{g \in \mathcal{G}} \|w_g\|_2,$$

  the indices $g$ in the sub-differential are given by

$$\partial_g F(w) \equiv \nabla_g f(w) + \lambda \partial \|w_g\|_2.$$

- In order to have $0 \in \partial F(w)$, we thus need for each group that

$$0 \in \nabla_g f(w) + \lambda \partial \|w_g\|_2,$$

  and subtracting $\nabla_g f(w)$ from both sides gives

$$-\nabla_g f(w) \in \lambda \partial \|w_g\|_2.$$

# Sub-differential of Group L1-Regularization

- So at minimizer $w^*$ we must have for all groups that

$$-\nabla_g f(w^*) \in \lambda \partial \|w_g^*\|_2.$$

- The sub-differential of the scaled L2-norm is given by

$$\partial \|w\|_2 = \begin{cases} \left\{ \frac{w}{\|w\|_2} \right\} & w \neq 0 \\ \{v \mid \|v\|_2 \leq 1\} & w = 0. \end{cases}$$

- So at a solution $w^*$ we have for each group that

$$\begin{cases} -\nabla_g f(w^*) = \lambda \frac{w_g^*}{\|w_g^*\|_2} & w_g \neq 0, \\ \|\nabla_g f(w^*)\| \leq \lambda & w_g^* = 0. \end{cases}$$

- For sufficiently-large $\lambda$ we'll set the group to zero.
  - With squared group norms we would need $\nabla_g f(w^*) = 0$ with $w_g^* = 0$ (unlikely).

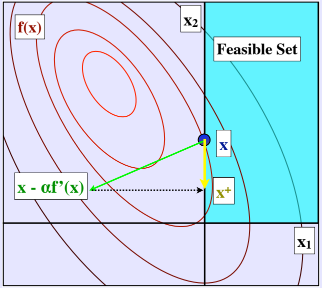
# Summary

- Simple convex sets are those that allow efficient projection.
- Simple regularizers are those that allow efficient proximal operator.
- Proximal-gradient: linear rates for sum of smooth and simple non-smooth.
- Group L1-regularization encourages sparsity in variable groups.

- Next time: going beyond L1-regularization to "structured sparsity".

# Line-Search for Projected Gradient

- There are two ways to do line-search for the projected gradient:
  - Backtrack along the line between $x^+$ and $x$ (search interior).
    - "Backtracking along the feasible direction", costs 1 projection per iteration.




  - Backtrack by decreasing $\alpha$ and re-projecting (search boundary).
    - "Backtracking along the projection arc", costs 1 projection per backtrack.
    - More expensive but (under weak conditions) we reach boundary in finite time.

# Faster Projected-Gradient Methods

- Accelerated projected-gradient method has the form

$$x^{k+1} = \text{proj}_{\mathcal{C}}[y^k - \alpha_k \nabla f(x^k)]$$
$$y^{k+1} = x^k + \beta_k(x^{k+1} - x^k).$$

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

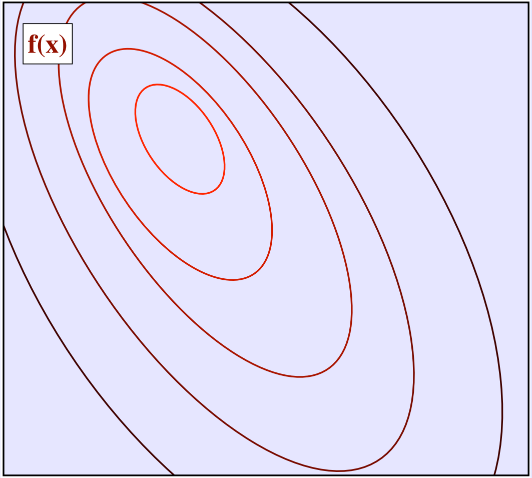
- The naive Newton-like methods with Hessian approximation $H_t$,

$$x^{k+1} = \text{proj}_{\mathcal{C}}[x^k - \alpha_k [H_k]^{-1} \nabla f(x^k)],$$

does NOT work.

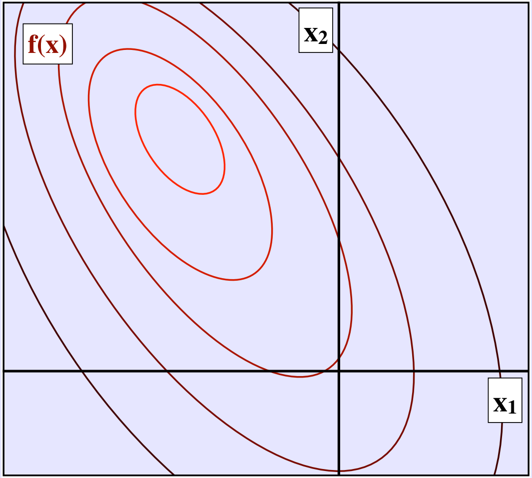
# Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



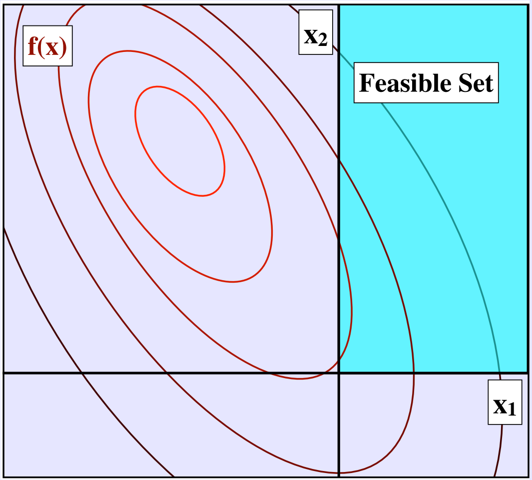
# Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



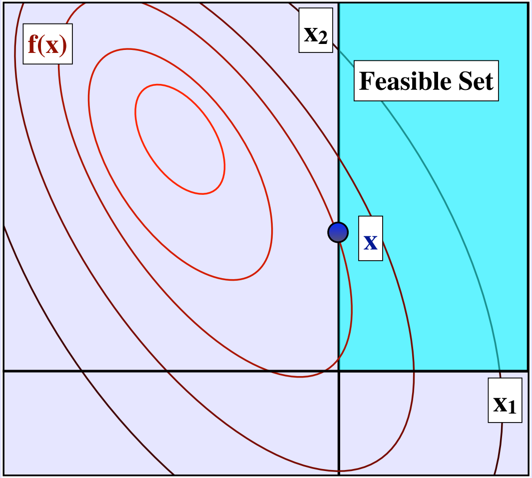
# Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



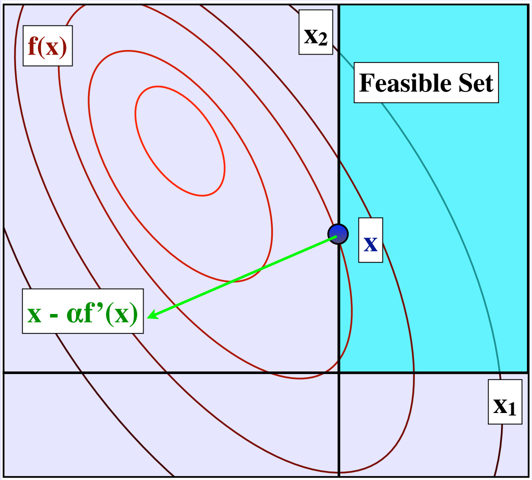
# Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



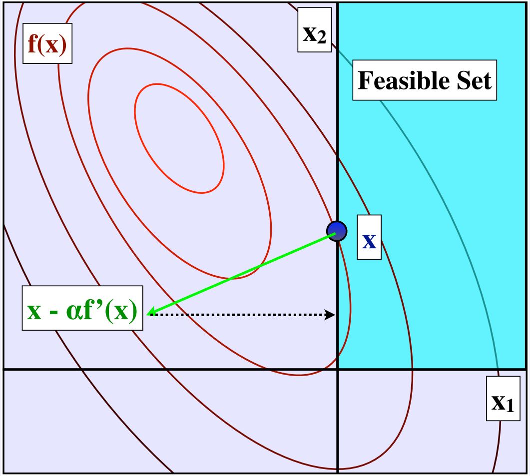
# Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



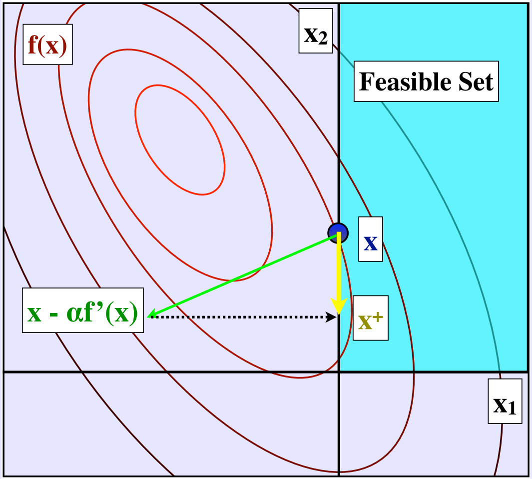
# Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



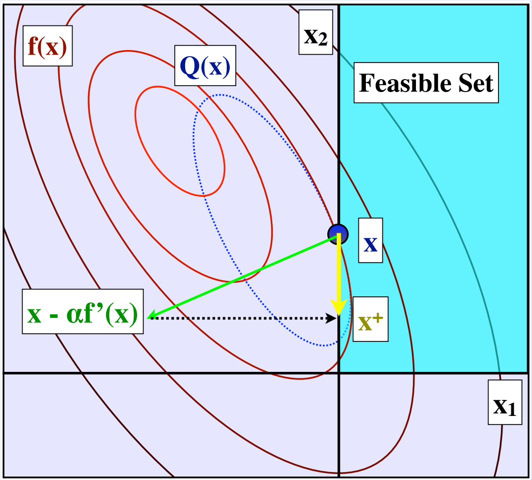
# Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



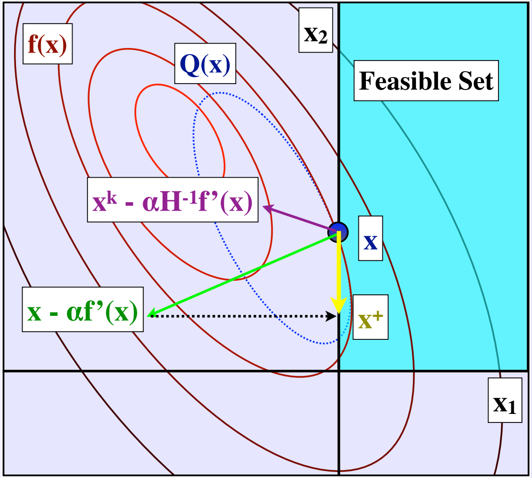
# Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



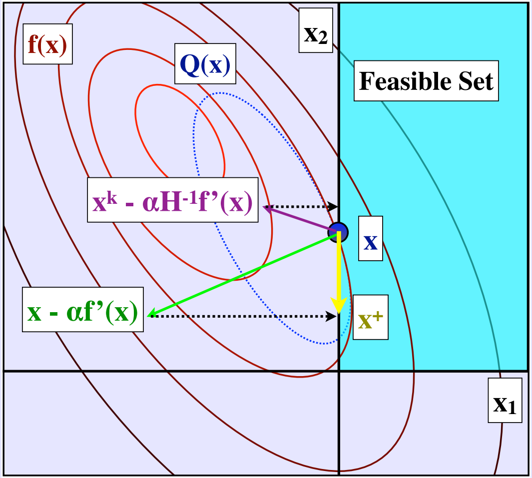
# Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



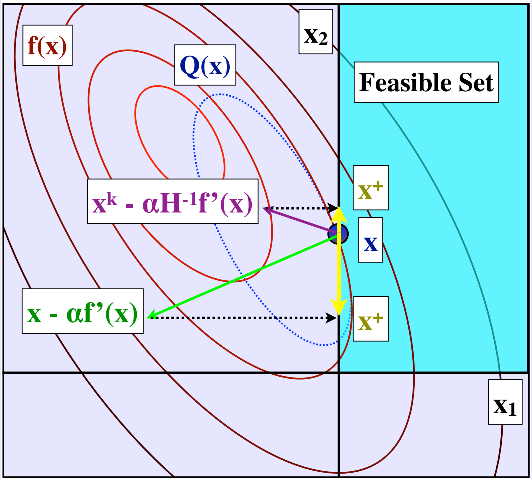
# Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



# Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



# Should we use projected-gradient for non-smooth problems?

- Some non-smooth problems can be turned into smooth problems with simple constraints.
- But transforming might make problem harder:
  - For L1-regularization least squares,

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2}\|Xw - y\|^2 + \lambda\|w\|_1,$$

  we can re-write as a smooth problem with bound constraints,

$$\underset{w_+ \geq 0,\ w_- \geq 0}{\operatorname{argmin}} \|X(w_+ - w_-) - y\|^2 + \lambda \sum_{j=1}^{d}(w_+ + w_-).$$

  - Doubles the number of variables.
  - Transformed problem is not strongly convex even if the original was.

# Projected-Newton Method

- We discussed how the naive projected-Newton method,

$$x^{k+\frac{1}{2}} = x^k - \alpha_k [H_k]^{-1} \nabla f(x^k) \qquad \text{(Newton-like step)}$$

$$x^{k+1} = \underset{y \in \mathcal{C}}{\text{argmin}} \|y - x^{k+\frac{1}{2}}\| \qquad \text{(projection)}$$

will not work.

- The correct projected-Newton method uses

$$x^{k+\frac{1}{2}} = x^k - \alpha_k [H_k]^{-1} \nabla f(x^k) \qquad \text{(Newton-like step)}$$

$$x^{k+1} = \underset{y \in \mathcal{C}}{\text{argmin}} \|y - x^{k+\frac{1}{2}}\|_{H_k} \qquad \text{(projection under Hessian metric)}$$

# Projected-Newton Method

- Projected-gradient minimizes quadratic approximation,

$$x^{k+1} = \operatorname*{argmin}_{y \in C} \left\{ f(x^k) + \nabla f(x^k)(y - x^k) + \frac{1}{2\alpha_k}\|y - x^k\|^2 \right\}.$$

- Newton's method can be viewed as quadratic approximation ($H_k \approx \nabla^2 f(x^k)$):

$$x^{k+1} = \operatorname*{argmin}_{y \in \mathbb{R}^d} \left\{ f(x^k) + \nabla f(x^k)(y - x^k) + \frac{1}{2\alpha_k}(y - x^k)H_k(y - x^k) \right\}.$$

- Projected Newton minimizes constrained quadratic approximation:

$$x^{k+1} = \operatorname*{argmin}_{y \in C} \left\{ f(x^k) + \nabla f(x^k)(y - x^k) + \frac{1}{2\alpha_k}(y - x^k)H_k(y - x^k) \right\}.$$

- Equivalently, we project Newton step under different Hessian-defined norm,

$$x^{k+1} = \operatorname*{argmin}_{y \in C} \|y - (x^k - \alpha_t H_k^{-1}\nabla f(x^k))\|_{H_k},$$

where general "quadratic norm" is $\|z\|_A = \sqrt{z^\top A z}$ for $A \succ 0$.

# Discussion of Projected-Newton

- Projected-Newton iteration is given by

$$x^{k+1} = \operatorname*{argmin}_{y \in C} \left\{ f(x^k) + \nabla f(x^k)(y - x^k) + \frac{1}{2\alpha_k}(y - x^k) H_k (y - x^k) \right\}.$$

- But this is expensive even when $\mathcal{C}$ is simple.

- There are a variety of practical alternatives:
  - If $H_k$ 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 $H_k$ partially diagonal.
  - Inexact projected-Newton: solve the above approximately.
    - Useful when $f$ is very expensive but $H_k$ and $\mathcal{C}$ are simple.
    - "Costly functions with simple constraints".

# Indicator Function for Convex Sets

- The indicator function for a convex set is
$$r(w) = \begin{cases} 0 & \text{if } w \in \mathcal{C} \\ \infty & \text{if } w \notin \mathcal{C} \end{cases}.$$
- This is a function with "extended-real-valued" output: $r : \mathbb{R}^d \to \{\mathbb{R}, \infty\}$.
- The convention for convexity of such functions:
  - The "domain" is defined as the $w$ values where $r(w) \neq \infty$ (in this case $\mathcal{C}$).
  - We need this domain to be convex.
  - And the function should to be convex on this domain.

# Properties of Proximal-Gradient

- Two convenient properties of proximal-gradient:
  - Proximal operators are non-expansive,

    $$\|\text{prox}_r(x) - \text{prox}_r(y)\| \leq \|x - y\|,$$

    it only moves points closer together.

    (including $x^k$ and $x^*$)
  - For convex $f$, only fixed points are global optima,

    $$x^* = \text{prox}_r(x^* - \alpha \nabla f(x^*)),$$

    for any $\alpha > 0$.

    (can test $\|x^k - \text{prox}_r(x^k - \nabla f(x^k))\|$ for convergence )
- Proximal gradient/Newton has two line-searches (generalized projected variants):
  - Fix $\alpha_k$ and search along direction to $x^{k+1}$ (1 proximal operator, non-sparse iterates).
  - Vary $\alpha_k$ values (multiple proximal operators per iteration, gives sparse iterations).

# Implicit subgradient viewpoint of proximal-gradient

- The proximal-gradient iteration is

$$w^{k+1} \in \underset{v \in \mathbb{R}^d}{\text{argmin}} \frac{1}{2}\|v - (w^k - \alpha_k \nabla f(w^k))\|^2 + \alpha_k r(v).$$

- By non-smooth optimality conditions that 0 is in subdifferential, we have that

$$0 \in (w^{k+1} - (w^k - \alpha_k \nabla f(w^k)) + \alpha_k \partial r(w^{k+1}),$$

  which we can re-write as

$$w^{k+1} = w^k - \alpha_k(\nabla f(w^k) + \partial r(w^{k+1})).$$

- So proximal-gradient is like doing a subgradient step, with
  1. Gradient of the smooth term at $w^k$.
  2. A particular subgradient of the non-smooth term at $w^{k+1}$.
     - "Implicit" subgradient.

# Proximal-Gradient Convergence under Proximal-PL

- By Lipschitz continuity of $g$ we have

$$\begin{aligned} F(x_{k+1}) &= g(x_{k+1}) + r(x_k) + r(x_{k+1}) - r(x_k) \\ &\leq F(x_k) + \langle \nabla g(x_k), x_{k+1} - x_k \rangle + \frac{L}{2}||x_{k+1} - x_k||^2 + r(x_{k+1}) - r(x_k) \\ &\leq F(x_k) - \frac{1}{2L}\mathcal{D}_r(x_k, L) \\ &\leq F(x_k) - \frac{\mu}{L}[F(x_k) - F^*], \end{aligned}$$

and then we can take our usual steps.

# Faster Rate for Proximal-Gradient

- It's possible to show a slightly faster rate for proximal-gradient using $\alpha_t = 2/(\mu + L)$.
- See `http://www.cs.ubc.ca/~schmidtm/Documents/2014_Notes_ProximalGradient.pdf`

# Debugging a Proximal-Gradient Code

- In general, debugging optimization codes can be difficult.
  - The code can appear to work even if it's wrong.
- A reasonable strategy is to test things you expect to be true.
  - And keep a set of tests that should remain true if you update the code.
- For example, for proximal-gradient methods you could check:
  - Does it decrease the objective function for a small enough step-size?
  - Are the step-sizes sensible (are they much smaller than $1/L$)?
  - Is the optimality condition going to zero as you run the algorithm?
- For group L1-regularization, some other checks that you can do:
  - Set $\lambda = 0$ and see if you get the unconstrained solution.
  - Assign each variable to its own group and see if you get the L1-regularized solution.
  - Assign all variables to the same group and see if you get an L2-regularization solution (and 0 for large-enough $\lambda$).

# Proximal-Newton

- We can define proximal-Newton methods using

$$x^{k+\frac{1}{2}} = x^k - \alpha_k [H_k]^{-1} \nabla f(x^k) \qquad \text{(gradient step)}$$

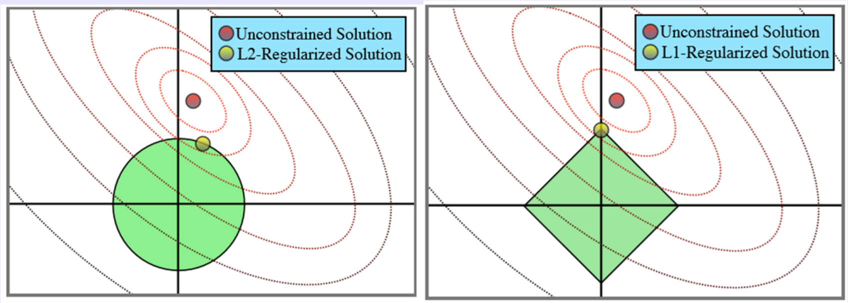
$$x^{k+1} = \operatorname*{argmin}_{y \in \mathbb{R}^d} \left\{ \frac{1}{2} \|y - x^{k+\frac{1}{2}}\|_{H_k}^2 + \alpha_k r(y) \right\} \qquad \text{(proximal step)}$$

- This is expensive even for simple $r$ like L1-regularization.
- But there are analogous tricks to projected-Newton methods:
  - Diagonal or Barzilai-Borwein Hessian approximation.
  - "Orthant-wise" methods are analogues of two-metric projection.
  - Inexact methods use approximate proximal operator.

# L1-Regularization vs. L2-Regularization

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

$$\underset{w\in\mathbb{R}^d}{\text{argmin}}\, f(w) + \lambda\|w\|_p \quad \Leftrightarrow \quad \underset{w\in\mathbb{R}^d,\tau\in\mathbb{R}}{\text{argmin}}\, f(w) + \lambda\tau \text{ with } \tau \geq \|w\|_p.$$

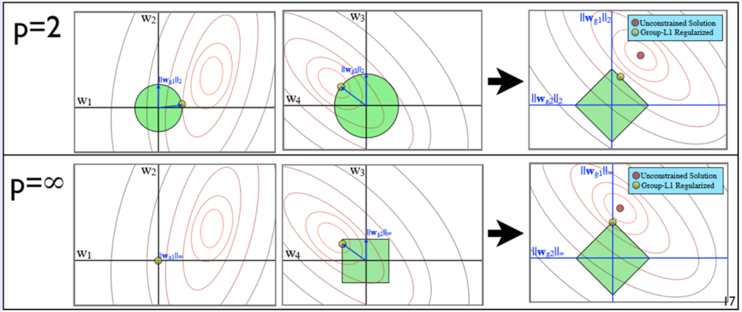



- Note that we're also minimizing the radius $\tau$.
  - If $\tau$ shrinks to zero, all $w$ are set to zero.
  - But if $\tau$ is squared there is virtually no penalty for having $\tau$ non-zero.

# Group L1-Regularization

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

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




- 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

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

$$\underset{x \in \mathbb{R}^d}{\text{argmin}}\, g(x) + \lambda \sum_{g \in G} \|x_g\|_2,$$

into a smooth problem with simple constraints:

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

- Here the constraitnts are separable:
  - We can project onto each norm-cone separately.
- Since norm-cones are simple we can solve this with projected-gradient.
  - But we have more variables in the transformed problem and lose strong-convexity.

# Proximal-Gradient for L0-Regularization

- There are some results on proximal-gradient for non-convex $r$.
- Most common case is L0-regularization,

$$f(w) + \lambda\|w\|_0,$$

  where $\|w\|_0$ is the number of non-zeroes.
  - Includes AIC and BIC from 340.
- The proximal operator for $\alpha_k\lambda\|w\|_0$ is simple:
  - Set $w_j = 0$ wihenever $|w_j| \leq \alpha_k\lambda$ ("hard" thresholding).
- Analysis is complicated a bit because discontinuity of prox as function of $\alpha_k$.