First-Order Optimization Algorithms for Machine Learning

Projected-Gradient Methods

Mark Schmidt

University of British Columbia

Summer 2020

Last Time: Subgradient Methods

- The basic subgradient method for solving non-smooth problems:

$$w^{k+1} = w^k - \alpha_k g_k,$$

for any $g_k \in \partial f(w^k)$ ("sub-differential").

- Subgradient-based methods are slower than gradient descent:

Assumption	Gradient	Subgradient	Quantity
Convex	$O(1/\epsilon)$	$O(1/\epsilon^2)$	$f(w^t) - f^* \leq \epsilon$
Strongly-Convex	$O(\log(1/\epsilon))$	$O(1/\epsilon)$	$f(w^t) - f^* \leq \epsilon$

- You cannot improve subgradient rates by acceleration.
  - There are matching lower bounds for dimension-independent algorithms.
  - Later we'll show stochastic subgradient methods have these rates at lower cost.

The Key to Faster Methods

- How can we achieve the speed of gradient descent on non-smooth problems?
  - Make extra assumptions about the function f or algorithm.

- For L1-regularized least squares, we'll use that the objective has the form

$$F(w) = \underbrace{f(w)}_{\text{smooth}} + \underbrace{r(w)}_{\text{"simple"}},$$

  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.

- Proximal-gradient methods have rates of gradient descent for such problems.
  - A generalization of projected gradient methods, which we'll cover first.

Projected-Gradient for Non-Negative Constraints

- We used projected gradient in 340 for NMF to find non-negative solutions,

$$\underset{w \geq 0}{\text{argmin}}\, f(w).$$

- In this case the algorithm has a simple form,

$$w^{k+1} = \max\{0, \underbrace{w^k - \alpha_k \nabla f(w^k)}_{\text{gradient descent}}\},$$

  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.

Transforming L1-Regularization into a Problem with Bound Constraints

- What does this have to do with L1-regularization?
- Can transform many non-smooth problems into smooth + simple constraints.
- For smooth objectives with L1-regularization,

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} f(w) + \lambda \|w\|_1,$$

  we can re-write as a smooth problem with only non-negative constraints,

$$\operatorname*{argmin}_{w_+ \geq 0,\, w_- \geq 0} f(w_+ - w_-) + \lambda \sum_{j=1}^{d} (w_+ + w_-).$$

- Can then apply projected-gradient to this problem.

A Broken "Projected-Gradient" Algorithms

- Projected-gradient addresses problem of minimizing smooth f over a convex set $\mathcal{C}$,

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

- As another example, we often want w to be a probability,

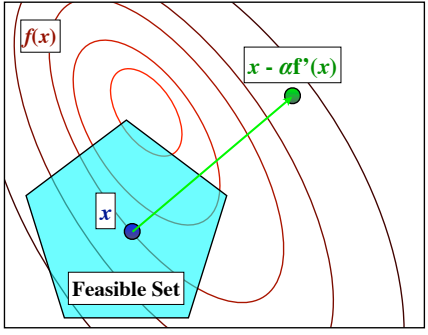
$$\operatorname*{argmin}_{w \geq 0,\ 1^\top w = 1} f(w),$$

- Based on our "set negative values to 0" intuition, we might consider this:
  1. Perform an unconstrained gradient descent step.
  2. Set negative values to 0 and divide by the sum.

- This algorithms does NOT work.
  - But it can be fixed if we replace Step 2 by "project onto the constraint set".

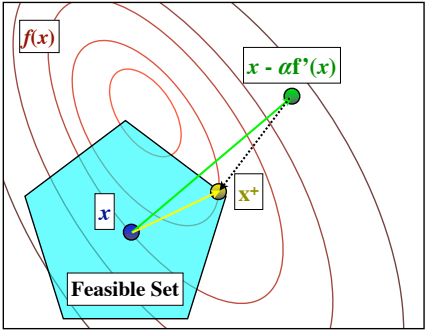
Projected-Gradient

$$w^{k+\frac{1}{2}} = \underbrace{w^k - \alpha_k \nabla f(w^k)}_{\text{gradient step}}, \quad w^{k+1} \in \underbrace{\underset{v \in \mathcal{C}}{\text{argmin}} \, \|v - w^{k+\frac{1}{2}}\|}_{\text{projection step}}.$$



Projected-Gradient

$$w^{k+\frac{1}{2}} = \underbrace{w^k - \alpha_k \nabla f(w^k)}_{\text{gradient step}}, \quad w^{k+1} \in \underbrace{\underset{v \in \mathcal{C}}{\text{argmin}} \, \|v - w^{k+\frac{1}{2}}\|}_{\text{projection step}}.$$



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.
  - 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 (later).
  - 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 α_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: $w^{k+1} = \text{proj}_{\mathcal{C}}[w^k - \alpha_k \nabla f(w^k)]$.

Gradient Mapping

- The projected gradient iteration is

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

- We can re-write this iteration as

$$w^{k+1} = w^k - \alpha g(w^k),$$

where g is called the gradient mapping

$$g(w^k) = \frac{1}{\alpha}(w^k - w^{k+1}).$$

- If we have no constraints then $g(w^k) = \nabla f(w^k)$ (so we get gradient descent).
  - Gradient mappings like this are often used in analyzing first-order methods.

- Since $g(w^*) = 0$, we often use $\|g(w^k)\|$ to monitor convergence.

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 $O(d)$ algorithms 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 τ).
  - Having a norm-cone constraint, $\|x\|_p \leq \tau$, for $p = 1, 2, \infty$ (variable τ).
- 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:

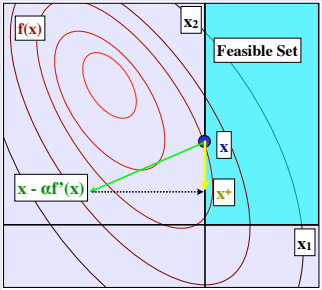
$$\mathcal{C} \equiv \{w \mid a_i^T w \leq b_i, \forall_i\}.$$

- 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 ϵ.

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

Line-Search for Projected Gradient

- There are two ways to do Armijo 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 α 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 (later).

Outline

Faster Projected-Gradient Methods

- Accelerated projected-gradient method has the form

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

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

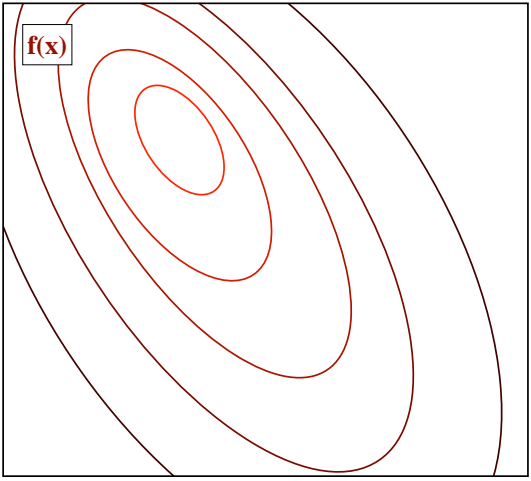
- The naive Newton-like methods with Hessian approximation H_t,

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

  does NOT work.

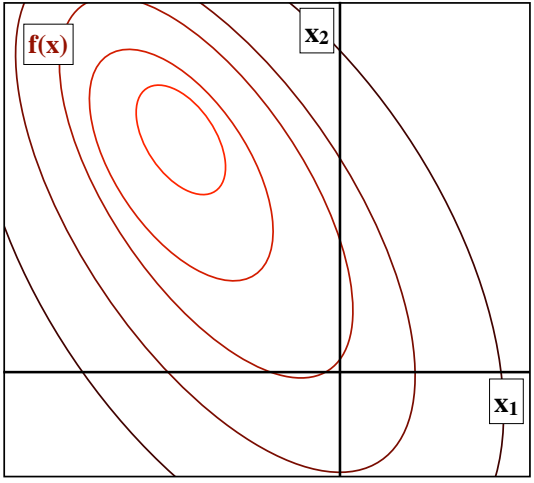
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



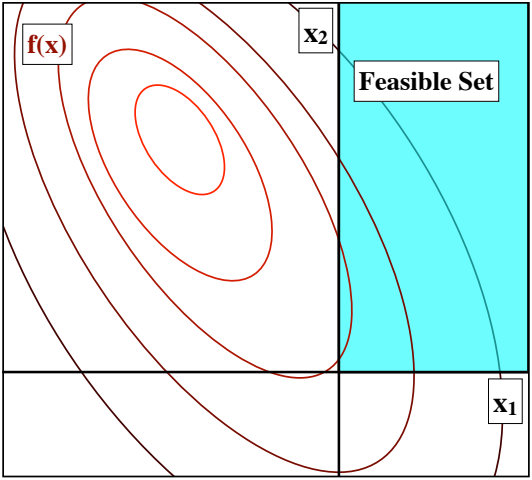
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



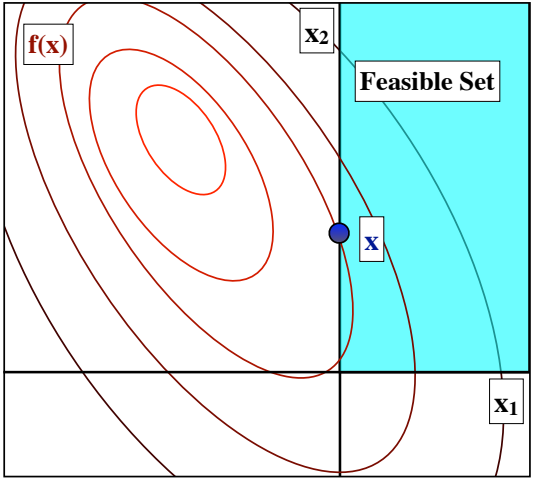
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



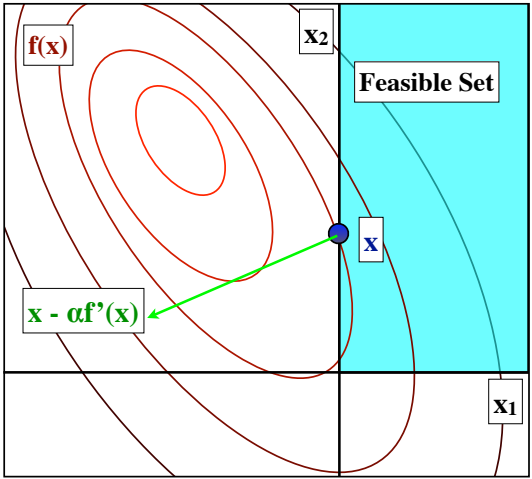
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



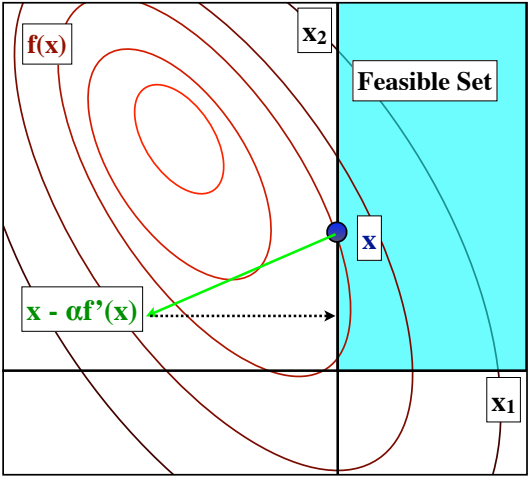
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



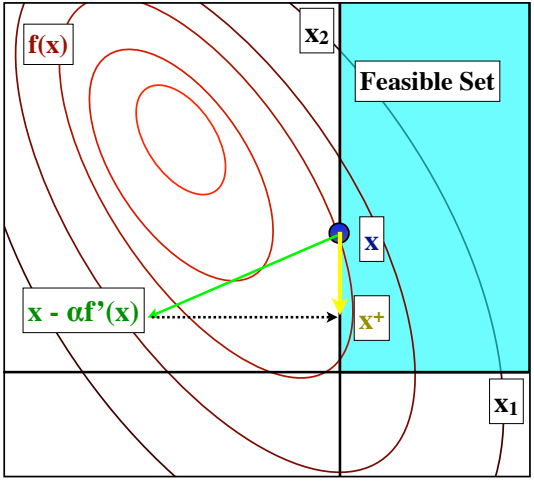
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



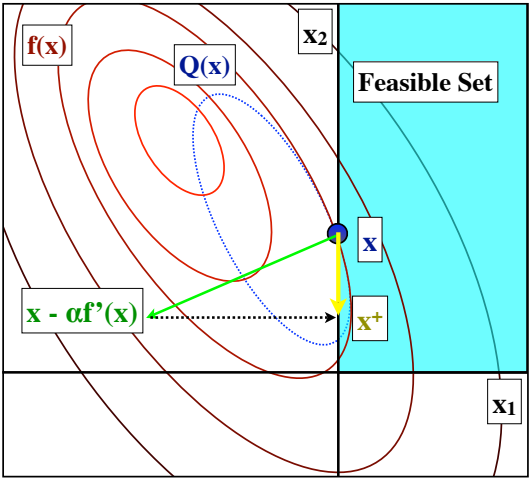
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



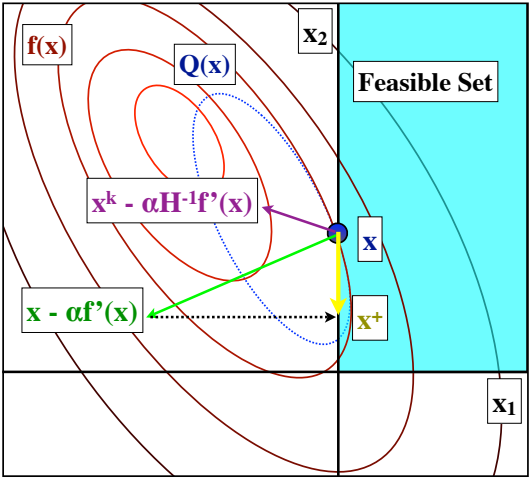
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



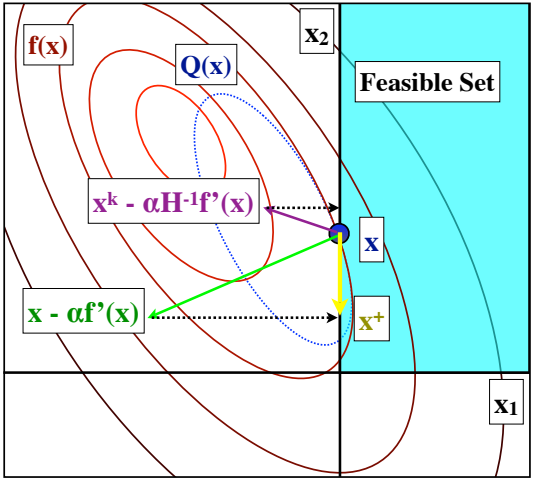
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



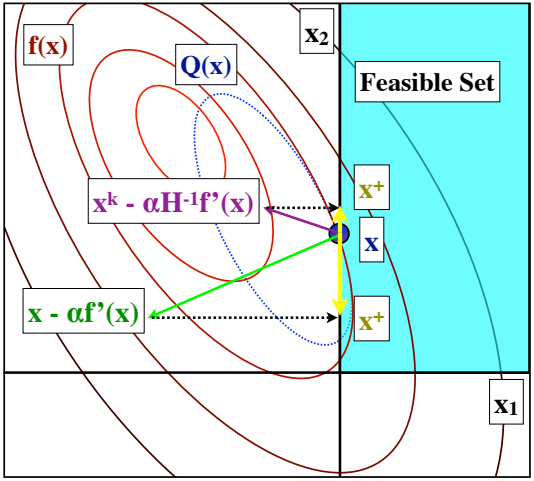
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



Projected-Newton Method

- The naive projected-Newton method,

$$w^{k+\frac{1}{2}} = w^k - \alpha_k [H_k]^{-1} \nabla f(w^k) \qquad \text{(Newton-like step)}$$
$$w^{k+1} = \underset{v \in \mathcal{C}}{\text{argmin}} \, \|v - w^{k+\frac{1}{2}}\| \qquad \text{(projection)}$$

  which will not work.

- The correct projected-Newton method uses

$$w^{k+\frac{1}{2}} = w^k - \alpha_k [H_k]^{-1} \nabla f(w^k) \qquad \text{(Newton-like step)}$$
$$w^{k+1} = \underset{v \in \mathcal{C}}{\text{argmin}} \, \|v - w^{k+\frac{1}{2}}\|_{H_k} \qquad \text{(projection under Hessian metric)}$$

Projected-Newton Method

- Projected-gradient minimizes quadratic approximation,

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

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

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

- Projected Newton minimizes constrained quadratic approximation:

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

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

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

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

$$w^{k+1} = \underset{y \in C}{\text{argmin}} \left\{ f(w^k) + \nabla f(w^k)(v - w^k) + \frac{1}{2\alpha_k}(v - w^k)H_k(v - w^k) \right\}.$$

- 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 for simple $\mathcal{C}$.
  - 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".

Summary

- Projected-gradient allows optimization with simple constraints.
- Simple convex sets are those that allow efficient projection.
- Projected Newton adds second-order information.
  - Faster convergence but expensive even for simple sets, need approximation
- Next time: how long does it take to find the sparsity pattern?