Numerical Optimization for Machine Learning

Projected-Gradient, Projecte-Newton, and Frank-Wolfe

Mark Schmidt

University of British Columbia

Summer 2022 - Summer 2023

Review: Gradient Descent

- The "training" phase in machine learning usually involves numerical optimization.
  - Minimizing a function f depending on d parameters w,

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

- For differentiable f, a prototypical method is gradient descent,

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

- Cost of update is $O(d)$ in terms.
- Guaranteed to decrease f for small enough step size α_k.

Review: Lipschitz Continuity of Gradient

- We considered functions f where the gradient is Lipschitz continuous,

$$\|\nabla f(w) - \nabla f(v)\| \leq L\|w - v\|,$$

  meaning that gradient cannot change faster than a constant L.
- Under this assumption we showed that gradient descent with $\alpha_k = 1/L$ satisfies

$$f(w^{k+1}) \leq f(w^k) - \frac{1}{2L}\|\nabla f(w^k)\|^2.$$

- But works better with clever step size choices and line searches.
  - We can show that similar progress bounds hold for these choices.

Review: Convergence Rates

- We discussed convergence rates and iteration complexities:
  1. Sublinear rates like $O(1/t)$, which requires $O(1/\epsilon)$ to get error ϵ.
  2. Linear rates like $O(\rho^t)$ for $\rho < 1$, which requires $O(\log(1/\epsilon))$.
  3. Superlinear rates like $O(\rho^{2^t})$, which requires $O(\log\log(1/\epsilon))$.

- When gradient is Lipschitz continuous, convergence rate of gradient descent is:
  - Linear $O(\rho^t)$ on $f(w^t) - f^*$ for functions that are strongly-convex (strongest).
  - Sublinear $O(1/t)$ on $\|\nabla f(w^t)\|^2$ for functons that are bounded below (weakest).
  - Sublinear $O(1/t)$ on $f(w^t) - f^*$ for functions that are convex.
  - Linear $O(\rho^t)$ on $f(w^t) - f^*$ for functions that satisfy Polyak-Łojasiewicz inequality.

Review: Faster Algorithms

- Get faster rates for strongly-convex quadratics with heavy-ball method,

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

for appropriate β^k (special case with optimal α_k and β_k is conjugate gradient).

- Get faster rates for convex functions with Nesterov's accelerated gradient method,

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

for appropriate β^k.

- Get faster local rates with Newton's method,

$$w^{k+1} = w^k - \alpha_k [\nabla^2 f(w^k)]^{-1} \nabla f(w^k),$$

and exist variations that handle case where Hessian $\nabla^2 f(w^k)$ is not invertible.

Review: Cheaper Algorithms

- Get cheaper iterations for many problems using coordinate optimization.
  - Optimizes 1 variable at a time, chosen cyclically/randomly/greedily.
  - Faster than gradient descent if iterations are d-times cheaper.
- Common problem structure is optimizing averages,

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w).$$

- In this setting get cheaper iterations with stochastic gradient descent (SGD),

$$w^{k+1} = w^k - \alpha_k \nabla f_{i_k}(w^k),$$

where i_k is a random sample from sum.

  - Converges slower than gradient descent but iterations are n-times cheaper.

Review: SGD Issues

- Progress of SGD depends on $\|\nabla f(w^k)\|$, α_k, batch size, and noise variance σ_k^2.
  - Decreasing step sizes or increasing batch sizes required for convergence.
  - Constant step size and batch size sufficient for any fixed accuracy.
- In various settings, can reduce/avoid effect of noise variance σ_k^2:
  - Variance reduction methods (SAG/SVRG) do this with a fixed step and batch size.
  - Over-parameterized models assume $\sigma_*^2 = 0$, which allows fixed step/batch size.
- Growing batches and SAG/SVRG and over-parameterzation line-search easier.

Review: Practical Newton Methods

- Get cheap approximations to Newton's method using:
  - Diagonal approximations of Hessian.
    - Cheap/easy but often does not work well.
  - Hessian-free Newton.
    - Uses cheap Hessian-vector products within conjugate gradient to approximate Newton.
  - Quasi-Newton
    - Updates approximation of Hessian based on observed differences in gradients.
  - Barzilai-Borwein step size
    - Degenerate quasi-Newton method that just sets the step size for gradient descent.

Outline

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.

A Broken "Projected-Gradient" Algorithms

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

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

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

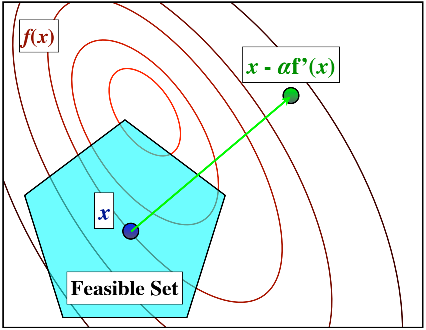
$$\underset{w \geq 0,\ 1^\top w = 1}{\operatorname{argmin}}\ f(w),$$

- 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

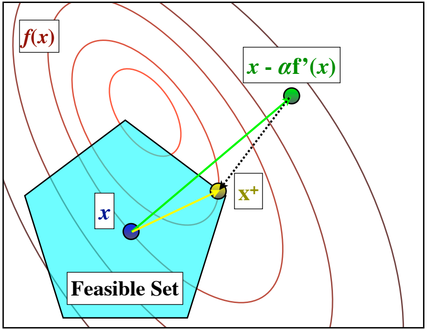
$$w^{k+\frac{1}{2}} = \underbrace{w^k - \alpha_k \nabla f(w^k)}_{\text{gradient step}}, \quad w^{k+1} \in \underbrace{\underset{v \in \mathcal{C}}{\text{argmin}} \|v - w^{k+\frac{1}{2}}\|}_{\text{projection step}}.$$



First proposed by Goldstein [1964] and Leviting & Polyak [1965].

Projected-Gradient

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



First proposed by Goldstein [1964] and Leviting & Polyak [1965].

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 $\mathcal{C}$,

$$w^{k+1} \in \underset{v \in \mathcal{C}}{\text{argmin}} \|v - w^{k+\frac{1}{2}}\|.$$

- Projection is the closest point that satisfies the constraints.
  - Generalizes "projection onto subspace" from linear algebra.
  - We will also write projection of w onto $\mathcal{C}$ as

$$\text{proj}_{\mathcal{C}}[w] = \underset{v \in \mathcal{C}}{\text{argmin}} \|v - w\|,$$

and for convex $\mathcal{C}$ it's unique.

Why the Projected Gradient?

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

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

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

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

  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,

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

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

Properties of Projected Gradient

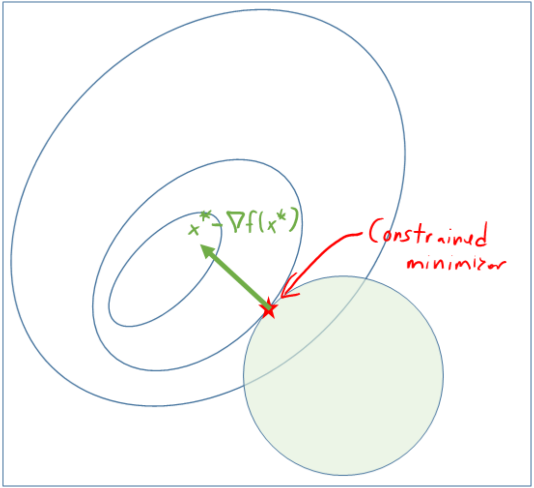
- Projected gradient for convex $\mathcal{C}$ has similar properties to unconstrained GD.
  - With $\alpha_k < 2/L$, guaranteed to decrease objective.
  - With $\alpha_k = 1/L$, get $O(1/k)$ rate for convex f with Lipschitz ∇f.
  - With $\alpha_k = 1/L$, get $O((1-\mu/L)^k)$ rate for strongly-convex f with Lipschitz ∇f.
    - And again get faster rate with $\alpha_k = 2/(\mu + L)$.

  - Minimizers w^* are "fixed points" of the update,

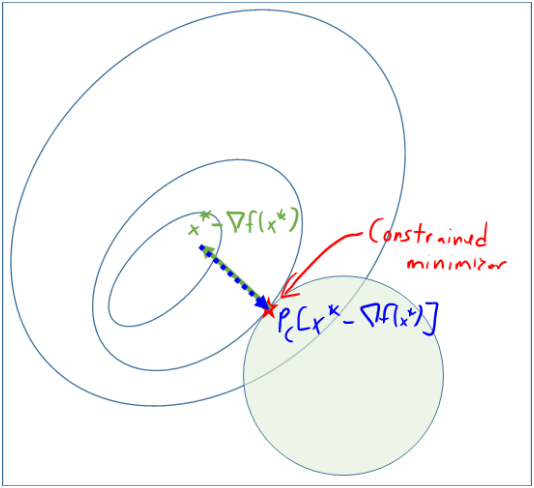
$$w^* = \text{proj}_{\mathcal{C}}[w^* - \alpha \nabla f(w^*)],$$

  for any step-size $\alpha > 0$ (generalizes $\nabla f(w^*) = 0$ for unconstrained case).
    - If f is convex then w^* is optimal iff the above holds.

Solution is Fixed Point of Projected Gradient Update

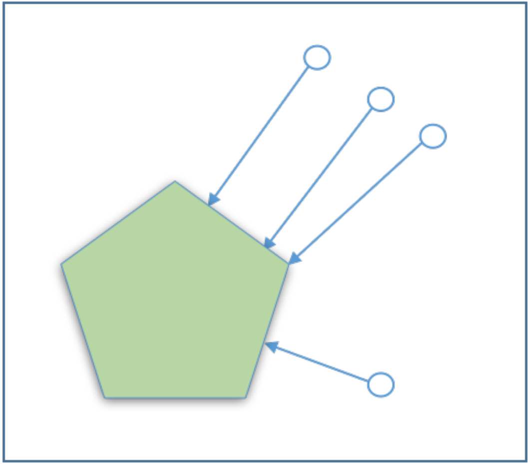


Solution is Fixed Point of Projected Gradient Update



Non-Expansiveness of Projection Operator

- Some analyses use that projection onto convex sets is non-expansive.
  - After projection, points will be the same distance or closer.

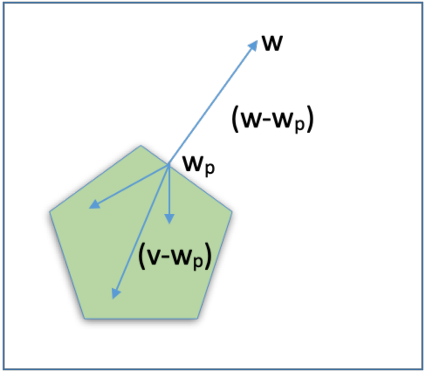


Gradient Mapping and Checking Convergence

- 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
$$w^{k+1} = w^k - \alpha_k g(w^k, \alpha_k),$$
where g is called the gradient mapping
$$g(w^k, \alpha_k) = \frac{1}{\alpha_k}(w^k - \underbrace{\text{proj}_{\mathcal{C}}[w^k - \alpha_k \nabla f(w^k)]}_{w^{k+1}}).$$
- If we have no constraints then $g(w^k, \alpha_k) = \nabla f(w^k)$ (so we get gradient descent).
- The gradient mapping has similar properties to the gradient:
  - We have that $-g(w^k, \alpha_k)$ points in a direction that decreases f (for any α_k).
  - We will next show that $\|g(w^k, \alpha_k)\|^2$ gives a measure of guaranteed progress.
  - Since $g(w^*, \alpha_k) = 0$ for any α_k, we could use $\|g(w^k, 1)\|$ to monitor convergence.

Digression: Projection Theorem

- Projection theorem: for convex sets, w_p is the projection of w iff
$$(w - w_p)^T(v - w_p) \leq 0,$$
for all v in $\mathcal{C}$.
    - "Angle between $(w - w_p)$ and $(v - w_p)$ is ≥ 90 degrees".



Projection Theorem and Gradient Mapping

- Projection theorem: for convex sets w_p is the projection of w iff

$$(w - w_p)^T(v - w_p) \leq 0,$$

  for all v in $\mathcal{C}$.

- If we set $w = w^k - \alpha_k \nabla f(w^k)$ then $w_p = w^{k+1}$, and choosing $v = w^k$ gives

$$\begin{aligned}
(w^k - \alpha_k \nabla f(w^k) - w^{k+1})^T(w^k - w^{k+1}) &\leq 0 \\
-\alpha_k \nabla f(w^k)^T(w^k - w^{k+1}) &\leq -(w^k - w^{k+1})^T(w^k - w^{k+1}) \\
\nabla f(w^k)^T(w^{k+1} - w^k) &\leq -\frac{1}{\alpha_k}\|\underbrace{w^k - w^{k+1}}_{\alpha_k g(w^k,\alpha_k)}\|^2 \\
\nabla f(w^k)^T(w^{k+1} - w^k) &\leq -\alpha_k \|g(w^k, \alpha_k)\|^2,
\end{aligned}$$

- Can be used in descent lemma to get a progress bound and convergence rate.

Progress Bound and Convergence Rate for Projected Gradient

- Recall the descent lemma when ∇f is Lipschitz then for all w and v,

$$f(v) \leq f(w) + \nabla f(w)^T(v-w) + \frac{L}{2}\|v-w\|^2,$$

- Setting $w = w^k$ and $v = w^{k+1}$ from projected gradient and using $\alpha_k = 1/L$ gives

$$f(w^{k+1}) \leq f(w^k) + \underbrace{\nabla f(w^k)^T(w^{k+1} - w^k)}_{\leq -\alpha_k\|g(w^k,\alpha_k\|)^2} + \frac{L}{2}\|\underbrace{w^{k+1} - w^k}_{\alpha_k g(w^k,\alpha_k)}\|^2$$

$$\leq f(w^k) - \alpha_k\|g(w,\alpha_k)\|^2 + \frac{L\alpha^2}{2}\|g(w^k,\alpha_k)\|^2$$

$$= f(w^k) - \frac{1}{2L}\|g(w,1/L)\|^2, \qquad \text{(with } \alpha_k = 1/L\text{)}$$

we get our usual progress bound but in terms of gradient mapping.

- For f bounded below, old arguments gives $\|g(w^k,1/L)\|^2$ converges at rate $O(1/k)$.

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 $O(d)$ 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 is 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".)

Outline

L1-Regularization

- A popular approach to feature selection we saw in 340 is L1-regularization:

$$F(w) = f(w) + \lambda\|w\|_1.$$

- Advantages:
  - Fast: can apply to large datasets, just minimizing one function.
    - Convex if f is convex.
  - Reduces overfitting because it simultaneously regularizes.
- Disadvantages:
  - Prone to false positives, particularly if you pick λ by cross-validation.
  - Not unique: there may be infinite solutions.
- There exist many extensions:
  - "Elastic net" adds L2-regularization to make solution unique.
  - "Bolasso" applies this on bootstrap samples to reduce false positives.
  - Non-convex regularizers reduce false positives but are NP-hard to optimize.

L1-Regularization

- Key property of L1-regularization: if λ is large, solution w^* is sparse:
    - w^* has many values that are exactly zero.

- How setting variables to exactly 0 performs feature selection in linear models:

$$\hat{y}^i = w_1 x_1^i + w_2 x_2^i + w_3 x_3^i + w_4 x_4^i + w_5 x_5^i.$$

- If $w = \begin{bmatrix} 0 & 0 & 3 & 0 & -2 \end{bmatrix}^\top$ then:

$$\begin{aligned} \hat{y}^i &= 0x_1^i + 0x_2^i + 3x_3^i + 0x_4^i + (-2)x_5^i \\ &= 3x_3^i - 2x_5^i. \end{aligned}$$

- Features $\{1, 2, 4\}$ are not used in making predictions: we "selected" $\{3, 5\}$.

Transforming L1-Regularization into a Problem with Bound Constraints

- What does L1-regularization have to do with constrained optimization?

- Can transform many non-smooth problems into smooth + simple constraints.
  - See the convex optimization notes on the webpage for a generic way to do this.

- For smooth objectives with L1-regularization,

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

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

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

- Can then apply projected-gradient to this problem.

Active-Set Identification

- For L1-regularization, projected-gradient "identifies" active set in finite time:

  (under mild assumptions)

  - For all sufficiently large k, sparsity pattern of w^k matches sparsity pattern of w^*.

$$w^0 = \begin{pmatrix} w_1^0 \\ w_2^0 \\ w_3^0 \\ w_4^0 \\ w_5^0 \end{pmatrix} \xrightarrow{\text{after finite } k \text{ iterations}} w^k = \begin{pmatrix} w_1^k \\ 0 \\ 0 \\ w_4^k \\ 0 \end{pmatrix}, \quad \text{where} \quad w^* = \begin{pmatrix} w_1^* \\ 0 \\ 0 \\ w_4^* \\ 0 \end{pmatrix}$$

- Useful if we are only interested in finding the sparsity pattern.
- Convergence rate will be faster once this happens (optimizing over subspace).
  - You could also apply unconstrained Newton-like methods on the non-zero variables.

Related Work and More-General Results

- Idea of finitely identifying non-zeroes dates back (at least) to Bertskeas [1976].
  - For projected-gradient applied to smooth functions with non-negative constraints.
- Has been shown for a variety of convex/non-convex problems.
  - Burke & Moré [1988], Wright [1993], Hare & Lewis [2004], Hare [2011].
- We will show the active-set identification property for non-negative constraints.
  - In this setting you identify the variables that are exactly zero.
  - For general problems, you identify constraints that hold with equality at solution.

Special Case: Optimizing with Non-Negative Constraints

- Consider optimization with non-negative constraints,

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

using the projected-gradient method with a step-size of $1/L$,

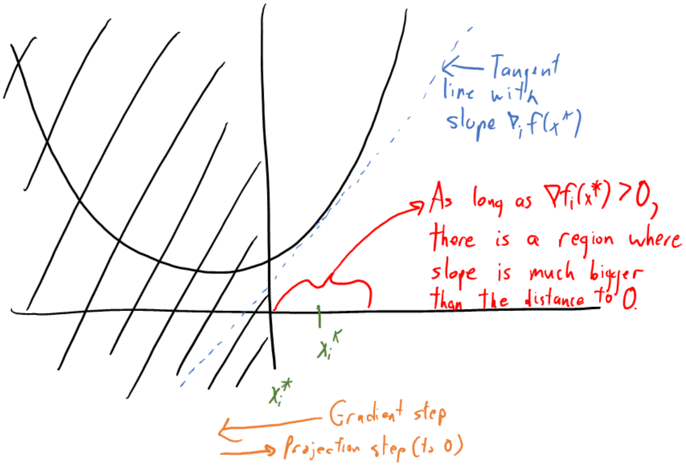
$$w^{k+1} = \left[w^k - \frac{1}{L}\nabla f(w^k)\right]^+.$$

- This leads to sparsity, and we use $\mathcal{Z}$ as the indices i where $w_i^* = 0$.

- We will assume:
  1. Gradient ∇f is L-Lipschitz continuous.
  2. We converge to an isolated minimizer w^*.
  3. Non-degeneracy condition: for all $i \in \mathcal{Z}$ we have $\nabla f(w_i^*) \geq \delta$ for some $\delta > 0$.
     - "You cannot have $\nabla_i f(w^*) = 0$ for variables i that are supposed to be zero."
     - This type of condition is standard: prevents reaching solution through interior.

Active-Set Identification for Non-Negative Constraints

- Let's show that we set $i \in \mathcal{Z}$ to zero when we're "close" to the solution.
  - Implies "for large 'k', if w_i^* is zero then the algorithm sets w_i^k to 0".



Active-Set Identification for Non-Negative Constraints

- Let's show that we set $i \in \mathcal{Z}$ to zero when we're "close" to the solution.
  - Implies "for large 'k', if w_i^* is zero then the algorithm sets w_i^k to 0".

- Since we assume projected-gradient converges to an isolated optimum, for all sufficiently large k we have $\|w^k - w^*\| \leq \frac{\delta}{2L}$.

- In this region we have two useful properties for all $i \in \mathcal{Z}$:
  1. The value of the variable must be small: $w_i^k \leq \frac{\delta}{2L}$.
     - Since $w_i^* = 0$ and w_i^k is within $\delta/2L$ of w_i.
  2. The value of the gradient must be large: $\nabla_i f(w^k) \geq \delta/2$.
     - Since we assumed $\nabla_i f(w^*) \geq \delta$ and ∇f is Lipschitz so $|\nabla_i f(w^k) - \nabla_i f(w^*)| \leq \|\nabla f(w^k) - \nabla f(w^*)\| \leq L\|w^k - w^*\| \leq \delta/2$.

- Plugging these into the projected-gradient update gives for $i \in \mathcal{Z}$ that

$$w_i^{k+1} = \left[w_i^k - \frac{1}{L}\nabla_i f(w^k)\right]^+ \leq \left[\frac{\delta}{2L} - \frac{\delta}{2L}\right]^+ = 0.$$

Superlinear Convergence after Identifying Active Set

- In a typical setting, we might hope that $|\mathcal{Z}^c| << d$.
  - Here we have the potential for faster algorithms by doing Newton steps on $\mathcal{Z}$.

- Some possibilities:
  - At some point, switch from projected-gradient to Newton on the manifold.
    - Unfortunately, hard to decide when to switch.
  - Each iteration checks progress of projected-gradient and Newton [Wright, 2012].
    - Choose whichever one makes the most progres.
  - Two-metric projection [Gafni & Bertsekas, 1984], discussed in next section.
    - May require expensive Newton steps before we're on the manifold.

  - There remains some theoretical and experimental work to do here.

Gradient Projection with 2 Step Sizes

- Active set identification can be affect by how we backtrack.

- Written in terms of the gradient mapping, projected gradient iteration is

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

  where notice that α_k appears twice.
  - In definition of gradient mapping, and how far we move.

- Consider introducing a second step size $\eta_k \leq 1$,

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

  which only affects how far we move in gradient mapping direction.
  - We previously considered $\alpha_k = 1/L$ and $\eta_k = 1$, but this works poorly in practice.
  - In practice, we typically fix one step size and backtrack along the other.

2 Backtracking Strategies for Projected Gradient

- Projected gradient written in terms of 2 step sizes is

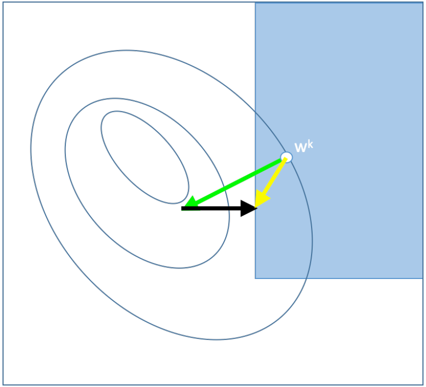
$$w^{k+1} = w^k - \eta_k \alpha_k g(w^k, \alpha_k),$$

- Backtracking along the feasible direction:
  - Fix α_k (typically at 1) and backtrack by reducing η_k.
  - Only 1 projection per iteration (good if projection is expensive).
  - But may is not guaranteed to identify active set.

- Backtracking along the projection arc:
  - Fix η_k at 1 and backtrack by reducing α_k.
  - 1 projection per backtracking step (bad if projection is expensive).
  - But identifies active set after finite number of iterations.

Backtracking Along the Feasible Direction

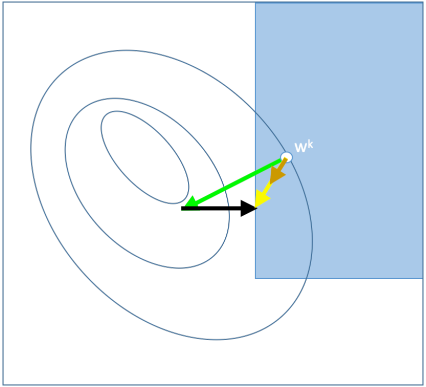
- Backtracking along the feasible direction:
  - Project once, then backtrack through the interior.



- Better if projection is expensive (and do not care about active set).

Backtracking Along the Feasible Direction

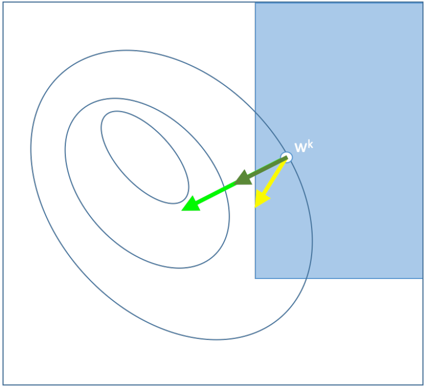
- Backtracking along the feasible direction:
  - Project once, then backtrack through the interior.



- Better if projection is expensive (and do not care about active set).

Backtracking Along the Projection Arc

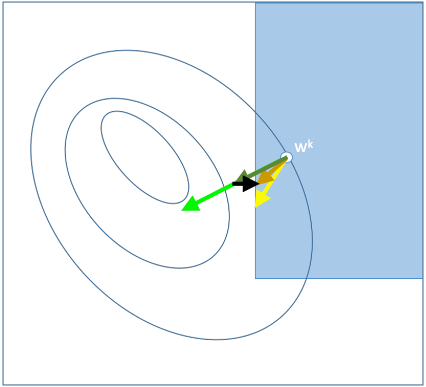
- Backtracking along the projection arc:
  - Backtrack then re-project, which may move along boundary.



- Better if projection is cheap (or want to identify active set).

Backtracking Along the Projection Arc

- Backtracking along the projection arc:
  - Backtrack then re-project, which may move along boundary.



- Better if projection is cheap (or want to identify active set).

Outline

Faster Projected-Gradient Methods

- An accelerated projected-gradient method has the form

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

  and this achieves accelerated rate with same α_k and β_k as unconstrained case.
  - Note that v^k may not satisfy constraints, but variants exist that keep v^k feasible.

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

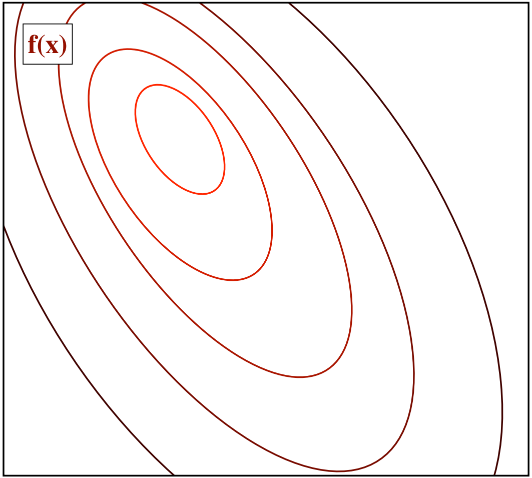
- The naive Newton-like methods with Hessian approximation H_k,

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

  does not work.

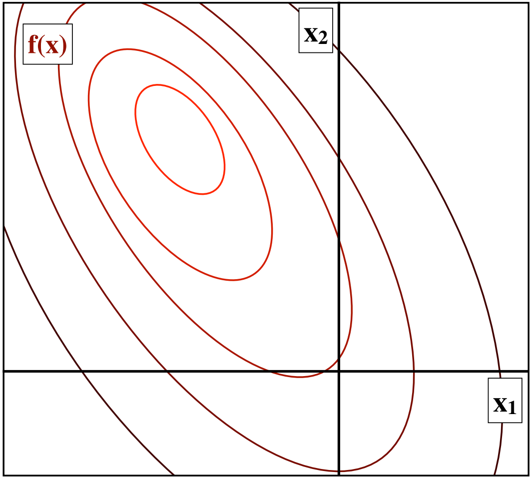
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



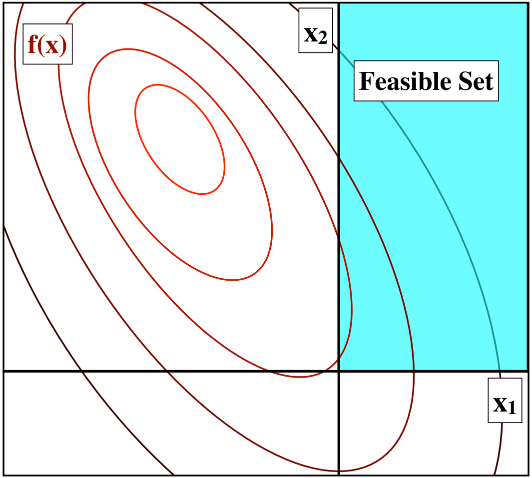
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



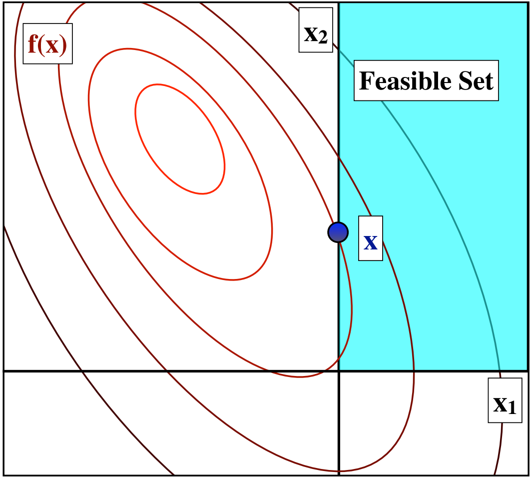
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



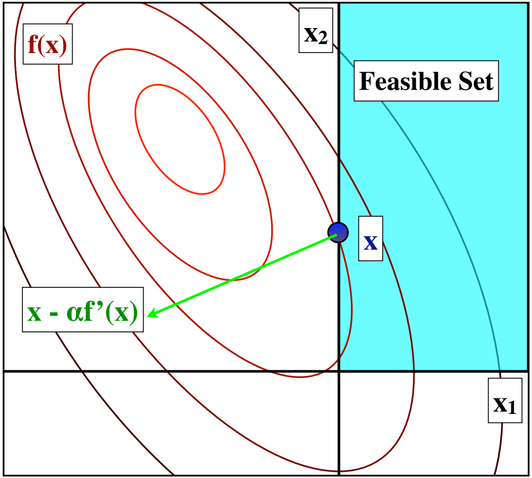
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



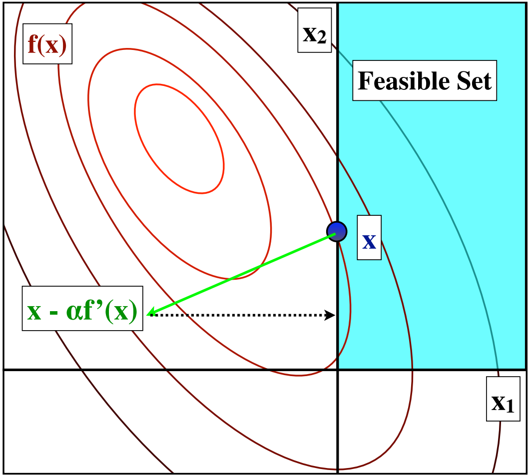
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



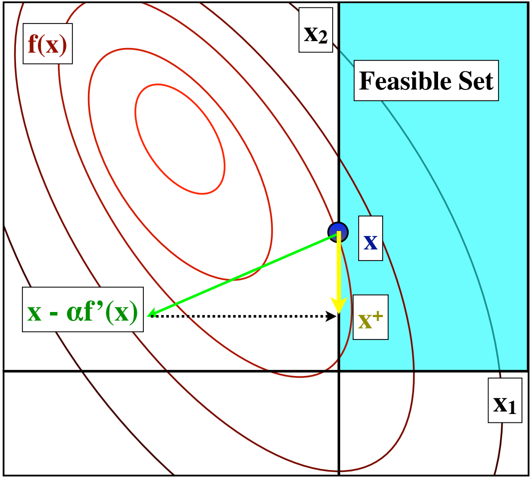
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



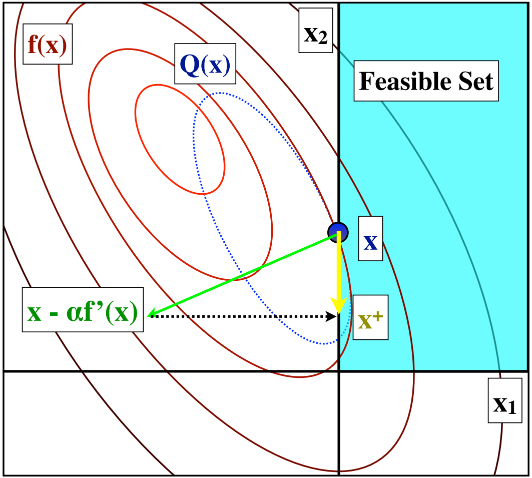
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



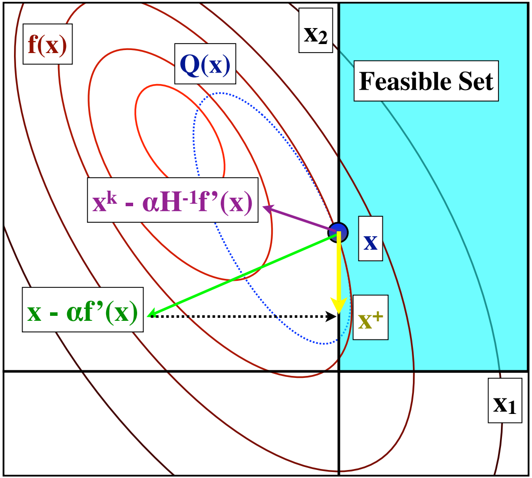
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



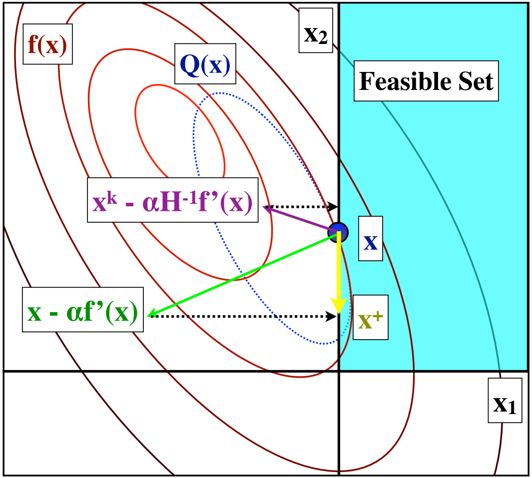
Naive Projected-Newton

Naive projected Newton method can point in the wrong direction.



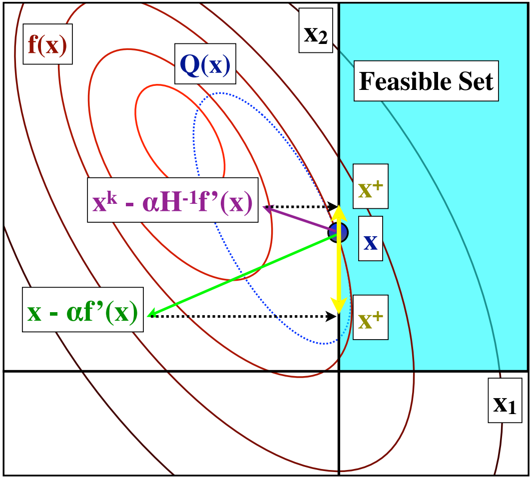
Naive Projected-Newton

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,

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

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

  which will not work.
  - Projection theorem does not imply Newton gives a descent direction.

- The correct projected-Newton method uses

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

$$w^{k+1} = \operatorname*{argmin}_{v \in \mathcal{C}} \|v - w^{k+\frac{1}{2}}\|_{H_k} \qquad \text{(projection under Hessian metric)}$$

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 $\mathcal{C}$.
  - Inexact projected-Newton: solve the above approximately.
    - Use a projected-gradient variant to minimize the above strongly-convex quadratic.
    - Useful when f is very expensive but H_k and $\mathcal{C}$ are simple.
    - "Optimizing costly functions with simple constraints" uses L-BFGS for H_k.
  - 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".

Two-Metric Projection for Bound Constraints

- Consider again optimizing with non-negative constraints, $\min_{w \in \mathcal{C}} f(w)$.
- The two-metric projection method splits the variables into two sets:

$$\mathcal{A}^k \equiv \{i \mid w_i^k = 0, \nabla_i f(w^k) > 0\},$$
$$\mathcal{I}^k \equiv \{i \mid w_i^k \neq 0 \text{ or } \nabla_i f(w^k) \leq 0\},$$

  the "active" variables (constrained at boundary) and "inactive variables".
- Uses a projected-gradient step on $\mathcal{A}^k$ and "naive" projected-Newton on $\mathcal{I}^k$.

$$w_{A^k}^{k+1} = \text{proj}_{\mathcal{C}}[w_{A^k}^k - \alpha_k \nabla_{A^k} f(w^k)]$$
$$w_{I^k}^{k+1} = \text{proj}_{\mathcal{C}}[w_{I^k}^k - \alpha_k [\nabla_{I^k}^2 f(w^k)]^{-1} \nabla_{I^k} f(w^k)]$$

- Eventually switches to unconstrained Newton on unconstrained variables.
- Can be generalized to general lower and upper bounds on individual variables.
  - Also exists a two-metric projection method for optimizing over probability simplex.

Outline

Cheaper Iterations with Projected Coordinate Optimization

- We can consider various ways to make projected-gradient iterations cheaper.
- In the special case of bounds constraints,

$$\min f(w), \quad l_i \leq w_i \leq u_i,$$

we can coordinate optimization or projected coordinate descent,

$$w_i^{k+1} = \text{proj}_{l_i \leq w_i \leq u_i}[w_i^k - \alpha_k \nabla_i f(w^k)],$$

where the projection step clips gradient descent to stay within the bounds.

  - Random coordinate optimization has same convergence rates as unconstrained case.

Coordinate Optimization with Non-Separable Constraints

- Coordinate optimization will not work for non-separable constraints.
- For example, consider optimizing with an equality constraint,

$$\min_w f(w), \quad \sum_{i=1}^n w_i = 1.$$

- If w satisfies the constraint, you cannot change any w_i without violating it.
- But you can change 2 variables i and j to maintain the constraint:

$$w_i^{k+1} = w_i^k - \alpha_k(\nabla_i f(w^k) - \nabla_j f(w^k))$$
$$w_j^{k+1} = w_j^k - \alpha_k(\nabla_j f(w^k) - \nabla_i f(w^k)).$$

- How to handle more complicated constraints gets ugly.
    - Special case: block-separable constraints (can use block coordinate optimization).

Projected Stochastic Gradient Descent

- We can consider projected stochastic gradient,

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

  where we do projected gradient on a random training example i_k.
  - Convergence properties are similar to unconstrained SGD.
  - Constraint does not need to be separable, but projection should be cheap.
    - Need to project n times per epoch.

- Some properties of SGD and projected-gradient that do not hold:
  - Lose fast convergence for over-parameterized models.
    - Because we no longer even have $\nabla f(w^*) = 0$.
  - Lose active set identification property of projected gradient.
    - Can leave boundary of constraints infinitely often.
    - Variant that restores this property is dual averaging,

$$w^{k+1} = \text{proj}_{\mathcal{C}}[w^0 - \frac{\alpha_k}{k} \sum_{t=1}^{k} \nabla f(w^k)],$$

      since it uses the average of the previous gradients (variance of direction goes to 0).

Frank-Wolfe Method ("Conditional Gradient")

- The projected-gradient method uses a quadratic approximation

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

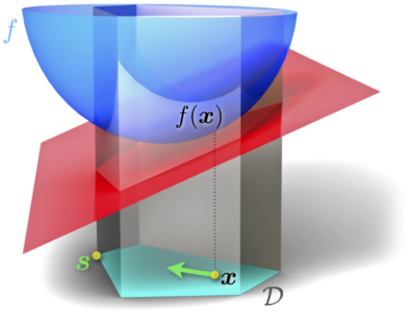
  and in some cases may be hard to compute (or even approximate).
- For these problems we can sometimes solve the simplified problem,

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

  which optimizes a linear approximation to the function over the constraint set.
  - This requires the set $\mathcal{C}$ to bounded, otherwise may be no solution.
- This is the basis of the conditional gradient method, also known as Frank-Wolfe
  - Marguerite Frank at NeurIPS in 2013: `https://www.youtube.com/watch?v=24e08AX9Eww`.

Frank-Wolfe Method ("Conditional Gradient")

- Visualization of the Frank-Wolfe approximation:



https://en.wikipedia.org/wiki/Frank\OT1\textendashWolfe_algorithm

- For convex f, minimizer of linear approximation gives lower bound on $f(w^*)$.
  - Like Newton, iterations are affine-invariant (don't change with affine transformation).

Frank-Wolfe Method ("Conditional Gradient")

- The Frank-Wolfe algorithm takes steps of the form

$$w^{k+1} = w^k + \alpha_k(v^k - w^k),$$

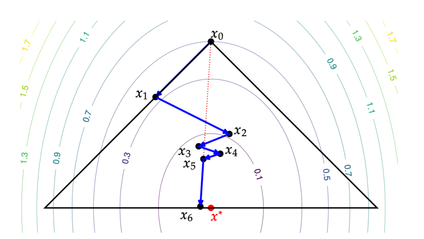
where $v^k \in \text{argmin}_{v\in\mathcal{C}} \nabla f(w^k)^\top v$.
  - So the gradient mapping would be $\frac{1}{\alpha_k}(w^k - v^k)$.

- Common ways to set the step size:
  - Decreasing: $\alpha_k = 2/(k+2)$.
  - Descent lemma: $\min\left\{1, \frac{\langle \nabla f(w^k), w^k - v^k\rangle}{L\|w^k - v^k\|^2}\right\}$ (works better if you approximate L).
  - Line search: $\text{argmin}_{0\leq\alpha\leq 1} f(w^k + \alpha_k(v^k - w^k))$ (works best).

- Convergence rate is $O(1/k)$ for convex and non-convex f.
  - Tends to be slower than projected-gradient in cases where they have similar costs.

Linear Convergence of Frank-Wolfe

- Basic Frank-Wolfe method has linear convergence in certain settings:
  - Function f is PL and solution is in interior of $\mathcal{C}$.
  - Function f is strongly convex and constraint $\mathcal{C}$ is uniformly convex.
- Several variations exist that obtain linear rates including away-step Frank-Wolfe:



https://arxiv.org/pdf/2211.14103.pdf

  - Frank-Wolfe moves towards vertex minimizing approximation resulting in zigzagging.
  - Away-steps move away from maximizing vertex (if larger directional derivative).
    - Above, iteration 6 moves away from initial vertex, moving onto boundary.
    - Recent variant is pairwise Frank-Wolfe, combining the above two steps.
    - Another variant is conditional gradient sliding, acceleration in terms of gradients.

Summary

- Projected-gradient allows optimization with simple constraints.
  - Same convergence speed as gradient descent.
- Simple convex sets are those that allow efficient projection.
- Active set identification of projected gradient.
  - Finds active constraints at solution in a finite number of iterations.
- 2 backtracking strategies for projected gradient.
  - Line search along feasible direction or backtrack along projection arc.
- Projected Newton adds second-order information.
  - Faster convergence but expensive even for simple sets, needs approximation.
- Projected coordinate descent works for bound constraints.
- Projected SGD works for large datasets.
  - But lose active set identification and fast convergence under over-parameterization
- Frank-Wolfe uses a linear rather than quadratic approximation.
  - Much cheaper than projection for some problems.

- Next time: non-smooth functions and finding the non-convex global min.