CS7301: Advanced Topics in Optimization for Machine Learning

Lecture 4.1: Projected Gradient Descent and Conditional Gradient Descent

Rishabh Iyer

Department of Computer Science
University of Texas, Dallas
https://github.com/rishabhk108/AdvancedOptML

February 12, 2021

Summary of what has been studied so far...

- Basics of Convex Functions and Convex Sets
- Properties of Convex Funcions, Examples
- Analysis of Gradient Descent: Lipschitz Continuous, Smooth, Strongly Convex
- Lower bounds of gradient algorithms (first order algorithms)
- Accelerated Gradient Descent
- Proximal Gradient Descent

Recap: Proximal Gradient Descent

- From the previous slide:

$$\text{argmin}_x \frac{1}{2\gamma}(x - [x_t - \gamma \nabla f(x_t)])^2 + h(x)$$

- Define $\text{prox}_t(x) = \text{argmin}_z \frac{1}{2t} ||x - z||^2 + h(z)$
- Notice that the update rule then is

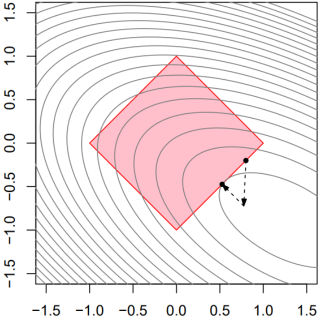
$$x_{t+1} = \text{prox}_\gamma(x_t - \gamma \nabla f(x_t))$$

- Convergence bound: Assume f, h are convex and the proximal minimization is easy (ideally closed form), then using a step size of $\gamma = 1/L$, we can bound: $f(x_T) - f(x^*) \leq \frac{LR^2}{2T}$
- This is exactly the same convergence rate for smooth functions!

Recap on Lagrangian and Duality

Projected Gradient Descent

- Consider the Problem of Constrained Convex Minimization: $\min_{x \in \mathcal{C}} f(x)$
- A simple modification of the gradient descent procedure is:
  1. At every iteration t: (Gradient Step): Compute $y_{t+1} = x_t - \alpha \nabla f(x_t)$
  2. (Projection step) $x_{t+1} = P_{\mathcal{C}}(y_{t+1}$
- Key here is the Projection step. Define $P_{\mathcal{C}}(x) = \mathrm{argmin}_{y \in \mathcal{C}} \frac{1}{2}||x - y||^2$



Projected Gradient Descent and Proximal Gradient Descent

- There is a close connection between Proximal and Projected Gradient Descent.
- Define $h(x) = I(x \in \mathcal{C})$ where $I(.)$ is the Indicator function.
- Its easy to see that the $\text{prox}_h(x) = P_{\mathcal{C}}(x)$, i.e. the Prox operator is exactly the same as a projection operator.
- As a result, projected gradient descent becomes a special case of proximal gradient descent.
- Theoretical results of Proj. GD: All results for standard Gradient descent carry over to the projected case as long as the projection operator is easy to compute!

Algorithm: Projected Gradient Descent (We use x_u^k instead of z^k)

Find a starting point $x_p^0 \in \mathcal{C}$.
Set $k = 1$
repeat

1. Choose a step size $t^k \propto 1/\sqrt{k}$.
2. Set $x_u^k = x_p^{k-1} - t^k \nabla f(x_p^{k-1})$.
3. Set $x_p^k = \operatorname{argmin}_{z \in \mathcal{C}} ||x_u^k - z||_2^2$.
4. Set $k = k + 1$.

until stopping criterion (such as $||x_p^k - x_p^{k-1}|| \leq \epsilon$ or $f(x_p^k) > f(x_p^{k-1})$) is satisfied[1]

Figure 1: The projected gradient descent algorithm.

[1] Better criteria can be found using Lagrange duality theory, etc.

Convergence Results for Projected Gradient Descent

- Lipschitz continuous functions PGD: R^2B^2/ϵ^2 iterations*
- Lipschitz continuous functions + Strongly Convex PGD: $2B^2/\epsilon - 1$ iterations*
- Smooth Functions PGD: $\frac{R^2L}{\epsilon}$ iterations.
- Smooth Functions Nesterov's PGD: $\sqrt{\frac{2LR^2}{\epsilon}}$ iterations*
- Smooth + Strongly Convex PGD: With $\gamma = 1/L$, achieve an ϵ-approximate solution in $\frac{L}{\mu}\log(\frac{R^2L}{2\epsilon})$ iterations.
- Smooth + Strongly Convex Nesterov's PGD: With $\gamma = 1/L$, achieve an ϵ-approximate solution in $\sqrt{\frac{L}{\mu}}\log(\frac{R^2L}{2\epsilon})$ iterations*.
- **Key Requirement for Projected Gradient to Work:** Projection must be easy (closed form obtainable) for the constraint.

* Ti...

Computing the Projection Operator

- Lets assume for simplicity that $\mathcal{C} = \{x | f(x) \leq c\}$
- Computing the projection step involves solving: $\min_z\{\frac{1}{2}||z - x||^2,$ such that $f(z) \leq c\}$.
- Use the idea of Lagrange multipliers!
- Define $g(z, \lambda) = \frac{1}{2}||z - x||^2 + \lambda(f(z) - c)$.
- Optimality conditions are: $\nabla_z g = 0$ and $\nabla_\lambda g = 0$!
- There are two options. Either $x \in \mathcal{C}$, in which case the constraints are not active, or x is outside $\mathcal{C}$ in which case we need $\nabla_z g = 0$ and $\nabla_\lambda g = 0$.
- The second case implies: $f(z) = c$ and $z - x + \lambda \nabla f(z) = 0$. If both these can be solved in closed form, we are done!

Computing the Projection Operator

- Optimality conditions imply: $f(z) = c$ and $z - x + \lambda \nabla f(z) = 0$. If both these can be solved in closed form, we are done!
- Compute the Projection operators for the constraints $\mathcal{C}_f = \{x \mid f(x) \leq c\}$
  - $f(x) = a^T x$
  - $f(x) = ||x||_2^2$
  - $f(x) = ||x||_1$
  - $f(x) = ||x - x_0||^2$
  - $f(x) = x^T Ax + bx + c$
  - $f(x) = ||x||_\infty$

Easy to Project Sets $\mathcal{C}$ (with closed form solutions)

- Solution set of a linear system $\mathcal{C} = \{\mathrm{x} \in \Re^n : A^T\mathrm{x} = \mathrm{b}\}$
- Affine images $\mathcal{C} = \{A\mathrm{x} + \mathrm{b} : \mathrm{x} \in \Re^n\}$
- Nonnegative orthant $\mathcal{C} = \{\mathrm{x} \in \Re^n : \mathrm{x} \succeq 0\}$. It may be hard to project on arbitrary polyhedron.
- Norm balls $\mathcal{C} = \{\mathrm{x} \in \Re^n : \|\mathrm{x}\|_p \leq 1\}$, for $p = 1, 2, \infty$

Table of Orthogonal Projections: See https://archive.siam.org/books/mo25/mo25_ch6.pdf

$$P_C(z) = prox_{I_C}(z) = \operatorname*{argmin}_{x} \frac{1}{2t}||x - z||^2 + I_C(x) = \operatorname*{argmin}_{x \in C} \frac{1}{2t}||x - z||^2$$

Set $C =$	For $t = 1$, $P_C(z) =$	Assumptions
$\Re^n_+$	$[z]_+$	
$\mathrm{Box}[l, u]$	$P_C(z)_i = min\{max\{z_i, l_i\}, u_i\}$	$l_i \leq u_i$
$\mathrm{Ball}[c, r]$	$c + \frac{r}{max\{\|z - c\|_2, r\}}(z - c)$	$\|.\|_2$ ball, centre $c \in \Re^n$ & radius $r > 0$
$\{x \| Ax = b\}$	$z - A^T(AA^T)^{-1}(Az - b)$	$A \in \Re^{m \times n}$, $b \in \Re^m$, A is full row rank
$\{x \| a^T x \leq b\}$	$z - \frac{[a^T z - b]_+}{\|a\|^2} a$	$0 \neq a \in \Re^n$ $b \in \Re$
Δ_n	$[z - \mu^* e]_+$ where $\mu^* \in \Re$ satisfies $e^T[z - \mu^* e]_+ = 1$	
$H_{a,b} \cap \mathrm{Box}[l, u]$	$P_{\mathrm{Box}[l,u]}(z - \mu^* a)$ where $\mu^* \in \Re$ satisfies $a^T P_{\mathrm{Box}[l,u]}(z - \mu^* a) = b$	$0 \neq a \in \Re^n$ $b \in \Re$
$H^-_{a,b} \cap \mathrm{Box}[l, u]$	$P_{\mathrm{Box}[l,u]}(z)$ $a^T P_{\mathrm{Box}[l,u]}(z) \leq b$ $P_{\mathrm{Box}[l,u]}(z - \lambda^* a)$ $a^T P_{\mathrm{Box}[l,u]}(z) > b$ where $\lambda^* \in \Re$ satisfies $a^T P_{\mathrm{Box}[l,u]}(z - \lambda^* a) = b$ & $\lambda^* > 0$	$0 \neq a \in \Re^n$ $b \in \Re$
$B_{\|\cdot\|_1}[0, \alpha]$	z $\|z\|_1 \leq \alpha$ $[z - \lambda^* e]_+ \odot sign(z)$ $\|z\|_1 > \alpha$ where $\lambda^* > 0$, & $[z - \lambda^* e]_+ \odot sign(z) = \alpha$	$\alpha > 0$

Convergence Results for Projected Gradient Descent

- Lipschitz continuous functions PGD: R^2B^2/ϵ^2 iterations*
- Lipschitz continuous functions + Strongly Convex PGD: $2B^2/\epsilon - 1$ iterations*
- Smooth Functions PGD: $\frac{R^2L}{\epsilon}$ iterations.
- Smooth Functions Nesterov's PGD: $\sqrt{\frac{2LR^2}{\epsilon}}$ iterations*
- Smooth + Strongly Convex PGD: With $\gamma = 1/L$, achieve an ϵ-approximate solution in $\frac{L}{\mu}\log(\frac{R^2L}{2\epsilon})$ iterations.
- Smooth + Strongly Convex Nesterov's PGD: With $\gamma = 1/L$, achieve an ϵ-approximate solution in $\sqrt{\frac{L}{\mu}}\log(\frac{R^2L}{2\epsilon})$ iterations*.
- **Key Requirement for Projected Gradient to Work:** Projection must be easy (closed form obtainable) for the constraint.

Computing the Projection Operator

- Note that the Projected gradient descent can be seen as optimizing the local quadratic expansion of f:

$$
\begin{aligned}
x_{k+1} &= P_{\mathcal{C}}(\text{argmin}_y[f(x_k) + \nabla f(x_k)^T(y - x_k) + \frac{1}{2\alpha_k}||y - x_k||^2]) \\
&= P_{\mathcal{C}}(\text{argmin}_y||y - (x_k - \alpha_k \nabla f(x_k))||^2)
\end{aligned}
$$

Computing the Projection Operator

- Note that the Projected gradient descent can be seen as optimizing the local quadratic expansion of f:

$$\begin{aligned} x_{k+1} &= P_{\mathcal{C}}(\text{argmin}_y[f(x_k) + \nabla f(x_k)^T(y - x_k) + \frac{1}{2\alpha_k}||y - x_k||^2]) \\ &= P_{\mathcal{C}}(\text{argmin}_y||y - (x_k - \alpha_k \nabla f(x_k))||^2) \end{aligned}$$

- This involves optimizing a quadratic function over the constraints, which is easy for only a smaller class of constraints.

Computing the Projection Operator

- Note that the Projected gradient descent can be seen as optimizing the local quadratic expansion of f:

$$\begin{aligned} x_{k+1} &= P_{\mathcal{C}}(\mathrm{argmin}_y[f(x_k) + \nabla f(x_k)^T(y - x_k) + \frac{1}{2\alpha_k}||y - x_k||^2]) \\ &= P_{\mathcal{C}}(\mathrm{argmin}_y||y - (x_k - \alpha_k \nabla f(x_k))||^2) \end{aligned}$$

- This involves optimizing a quadratic function over the constraints, which is easy for only a smaller class of constraints.
- For example, the Projection operator can only be computed for L_p norms with $p = 1, 2, \infty$ and not for other values of p.

Computing the Projection Operator

- Note that the Projected gradient descent can be seen as optimizing the local quadratic expansion of f:

$$\begin{aligned} x_{k+1} &= P_{\mathcal{C}}(\mathrm{argmin}_y[f(x_k) + \nabla f(x_k)^T(y - x_k) + \frac{1}{2\alpha_k}||y - x_k||^2]) \\ &= P_{\mathcal{C}}(\mathrm{argmin}_y||y - (x_k - \alpha_k \nabla f(x_k))||^2) \end{aligned}$$

- This involves optimizing a quadratic function over the constraints, which is easy for only a smaller class of constraints.
- For example, the Projection operator can only be computed for L_p norms with $p = 1, 2, \infty$ and not for other values of p.
- Similarly, projection is not easy on Polyhedral constraints (like the submodular polyhedron, combinatorial constraints like paths, cuts, ...)

Pros and Cons of Projected Gradient Descent

- Projected Gradient Descent:

$$x_{k+1} = P_{\mathcal{C}}(\text{argmin}_y ||y - (x_k - \alpha_k \nabla f(x_k))||^2)$$

Pros and Cons of Projected Gradient Descent

- Projected Gradient Descent:

$$x_{k+1} = P_{\mathcal{C}}(\text{argmin}_y ||y - (x_k - \alpha_k \nabla f(x_k))||^2)$$

- Pros:

Pros and Cons of Projected Gradient Descent

- Projected Gradient Descent:

$$x_{k+1} = P_{\mathcal{C}}(\text{argmin}_y ||y - (x_k - \alpha_k \nabla f(x_k))||^2)$$

- Pros:
  - Computing Projection operator easy in some cases: L2 Ball, convex cone, halfspaces, box, simplex

Pros and Cons of Projected Gradient Descent

- Projected Gradient Descent:

$$x_{k+1} = P_{\mathcal{C}}(\text{argmin}_y ||y - (x_k - \alpha_k \nabla f(x_k))||^2)$$

- Pros:
  - Computing Projection operator easy in some cases: L2 Ball, convex cone, halfspaces, box, simplex
  - Enjoys the same convergence rates as the unconstrained case

Pros and Cons of Projected Gradient Descent

- Projected Gradient Descent:

$$x_{k+1} = P_{\mathcal{C}}(\text{argmin}_y ||y - (x_k - \alpha_k \nabla f(x_k))||^2)$$

- Pros:
  - Computing Projection operator easy in some cases: L2 Ball, convex cone, halfspaces, box, simplex
  - Enjoys the same convergence rates as the unconstrained case
- Cons:

Pros and Cons of Projected Gradient Descent

- Projected Gradient Descent:

$$x_{k+1} = P_{\mathcal{C}}(\text{argmin}_y ||y - (x_k - \alpha_k \nabla f(x_k))||^2)$$

- Pros:
  - Computing Projection operator easy in some cases: L2 Ball, convex cone, halfspaces, box, simplex
  - Enjoys the same convergence rates as the unconstrained case
- Cons:
  - For high dimensional problems, projection can be expensive: $O(n)$ for Simplex, $O(nm^2)$ for nuclear norms, and involves a QP for general polyhedrons

Pros and Cons of Projected Gradient Descent

- Projected Gradient Descent:

$$x_{k+1} = P_{\mathcal{C}}(\text{argmin}_y ||y - (x_k - \alpha_k \nabla f(x_k))||^2)$$

- Pros:
  - Computing Projection operator easy in some cases: L2 Ball, convex cone, halfspaces, box, simplex
  - Enjoys the same convergence rates as the unconstrained case
- Cons:
  - For high dimensional problems, projection can be expensive: $O(n)$ for Simplex, $O(nm^2)$ for nuclear norms, and involves a QP for general polyhedrons
  - Full Gradient Descent may destroy certain desirable structure like sparsity

Pros and Cons of Projected Gradient Descent

- Projected Gradient Descent:

$$x_{k+1} = P_{\mathcal{C}}(\text{argmin}_y ||y - (x_k - \alpha_k \nabla f(x_k))||^2)$$

- Pros:
  - Computing Projection operator easy in some cases: L2 Ball, convex cone, halfspaces, box, simplex
  - Enjoys the same convergence rates as the unconstrained case
- Cons:
  - For high dimensional problems, projection can be expensive: $O(n)$ for Simplex, $O(nm^2)$ for nuclear norms, and involves a QP for general polyhedrons
  - Full Gradient Descent may destroy certain desirable structure like sparsity
- On the other hand, optimizing a linear function over constraints are much easier.

Pros and Cons of Projected Gradient Descent

- Projected Gradient Descent:

$$x_{k+1} = P_{\mathcal{C}}(\text{argmin}_y ||y - (x_k - \alpha_k \nabla f(x_k))||^2)$$

- Pros:
  - Computing Projection operator easy in some cases: L2 Ball, convex cone, halfspaces, box, simplex
  - Enjoys the same convergence rates as the unconstrained case
- Cons:
  - For high dimensional problems, projection can be expensive: $O(n)$ for Simplex, $O(nm^2)$ for nuclear norms, and involves a QP for general polyhedrons
  - Full Gradient Descent may destroy certain desirable structure like sparsity
- On the other hand, optimizing a linear function over constraints are much easier.
- Can we come up with an algorithm that only needs to optimize a linear function over constraints?

Conditional Gradient Descent

- Conditional Gradient Descent, also known as Frank Wolfe Method uses a local linear expansion of f:

$$s_k = \text{argmin}_{s \in \mathcal{C}} \nabla f(x_k)^T s \tag{1}$$

$$x_{k+1} = (1 - \gamma_k) x_k + \gamma_k s_k \tag{2}$$

Conditional Gradient Descent

- Conditional Gradient Descent, also known as Frank Wolfe Method uses a local linear expansion of f:

$$s_k = \text{argmin}_{s \in \mathcal{C}} \nabla f(x_k)^T s \quad (1)$$

$$x_{k+1} = (1 - \gamma_k) x_k + \gamma_k s_k \quad (2)$$

- Most critically, there is **no projection**! Just involves solving a linear program over $\mathcal{C}$

Conditional Gradient Descent

- Conditional Gradient Descent, also known as Frank Wolfe Method uses a local linear expansion of f:

$$s_k = \text{argmin}_{s \in \mathcal{C}} \nabla f(x_k)^T s \quad (1)$$

$$x_{k+1} = (1 - \gamma_k) x_k + \gamma_k s_k \quad (2)$$

- Most critically, there is **no projection**! Just involves solving a linear program over $\mathcal{C}$
- Default choice of step sizes is $\gamma_k = 2/(k+1), k = 1, 2, \cdots,$

Conditional Gradient Descent

- Conditional Gradient Descent, also known as Frank Wolfe Method uses a local linear expansion of f:

$$s_k = \text{argmin}_{s \in \mathcal{C}} \nabla f(x_k)^T s \tag{1}$$

$$x_{k+1} = (1 - \gamma_k)x_k + \gamma_k s_k \tag{2}$$

- Most critically, there is **no projection**! Just involves solving a linear program over $\mathcal{C}$
- Default choice of step sizes is $\gamma_k = 2/(k+1), k = 1, 2, \cdots,$
- Since $x_k, s_k \in \mathcal{C}$, it implies that $x_{k+1} \in \mathcal{C}$

Conditional Gradient Descent

- Conditional Gradient Descent, also known as Frank Wolfe Method uses a local linear expansion of f:

$$s_k = \text{argmin}_{s \in \mathcal{C}} \nabla f(x_k)^T s \tag{1}$$
$$x_{k+1} = (1 - \gamma_k)x_k + \gamma_k s_k \tag{2}$$

- Most critically, there is **no projection**! Just involves solving a linear program over $\mathcal{C}$
- Default choice of step sizes is $\gamma_k = 2/(k+1), k = 1, 2, \cdots,$
- Since $x_k, s_k \in \mathcal{C}$, it implies that $x_{k+1} \in \mathcal{C}$
- We are moving less and less in the direction of the linearization as the algorithm proceeds!

Example: Norm Constraints

What happens when $C = \{x : \|x\| \leq t\}$ for a norm $\|\cdot\|$? Then

$$\begin{aligned} s &\in \underset{\|s\|\leq t}{\operatorname{argmin}} \ \nabla f(x^{(k-1)})^T s \\ &= -t \cdot \left(\underset{\|s\|\leq 1}{\operatorname{argmax}} \ \nabla f(x^{(k-1)})^T s \right) \\ &= -t \cdot \partial \|\nabla f(x^{(k-1)})\|_* \end{aligned}$$

where $\|\cdot\|_*$ is the corresponding dual norm. In other words, if we know how to compute subgradients of the dual norm, then we can easily perform Frank-Wolfe steps

A key to Frank-Wolfe: this can often be simpler or cheaper than projection onto $C = \{x : \|x\| \leq t\}$. Also often simpler or cheaper than the prox operator for $\|\cdot\|$

Aside Dual Norms

- Define $f(x) = ||x||$ as a norm
- The dual norm $||x||_*$ is defined as:

$$||x||_* = \max_{||z|| \leq 1} z^T x$$

- Examples: Consider p-norm
- The dual norm of a p-norm $f(x) = ||x||_p$ is a q-norm such that $1/p + 1/q = 1$
- Dual of the 1-norm is the ∞-norm, Dual of the 2-norm is itself!
- Also,

$$\partial ||x||_* = \text{argmax}_{||Z|| \leq 1} z^T x$$

Example: L_1 Norm Constraints

For the ℓ_1-regularized problem

$$\min_x \; f(x) \;\; \text{subject to} \;\; \|x\|_1 \le t$$

we have $s^{(k-1)} \in -t\partial\|\nabla f(x^{(k-1)})\|_\infty$. Frank-Wolfe update is thus

$$\begin{aligned} i_{k-1} &\in \underset{i=1,\dots p}{\operatorname{argmax}} \; \left|\nabla_i f(x^{(k-1)})\right| \\ x^{(k)} &= (1-\gamma_k)x^{(k-1)} - \gamma_k t \cdot \operatorname{sign}\left(\nabla_{i_{k-1}} f(x^{(k-1)})\right) \cdot e_{i_{k-1}} \end{aligned}$$

Like greedy coordinate descent!

Note: this is a lot simpler than projection onto the ℓ_1 ball, though both require $O(n)$ operations

Example: L_p Norm Constraints

For the ℓ_p-regularized problem

$$\min_x \ f(x) \ \text{ subject to } \ \|x\|_p \leq t$$

for $1 \leq p \leq \infty$, we have $s^{(k-1)} \in -t\partial\|\nabla f(x^{(k-1)})\|_q$, where p, q are dual, i.e., $1/p + 1/q = 1$. Claim: can choose

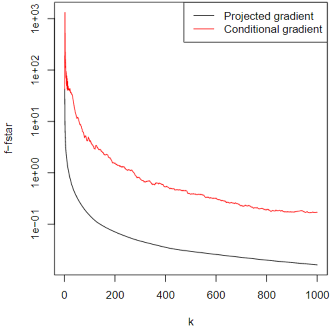
$$s_i^{(k-1)} = -\alpha \cdot \text{sign}\big(\nabla f_i(x^{(k-1)})\big) \cdot \big|\nabla f_i(x^{(k-1)})\big|^{p/q}, \quad i = 1, \dots n$$

where α is a constant such that $\|s^{(k-1)}\|_q = t$ (check this!), and then Frank-Wolfe updates are as usual

Note: this is a lot simpler projection onto the ℓ_p ball, for general p! Aside from special cases ($p = 1, 2, \infty$), these projections cannot be directly computed (must be treated as an optimization)

Empirically: Projected vs Conditional Gradient Descent

Comparing projected and conditional gradient for constrained lasso problem, with $n = 100$, $p = 500$:



We will see that Frank-Wolfe methods match convergence rates of known first-order methods; but in practice they can be slower to converge to high accuracy (note: fixed step sizes here, line search would probably improve convergence)

Duality Gap

Frank-Wolfe iterations admit a very natural duality gap (truly, a suboptimality gap):

$$\max_{s \in C} \nabla f(x^{(k-1)})^T (x^{(k-1)} - s)$$

This is an upper bound on $f(x^{(k-1)}) - f^\star$

Proof: by the first-order condition for convexity

$$f(s) \geq f(x^{(k-1)}) + \nabla f(x^{(k-1)})^T (y - x^{(k-1)})$$

Minimizing both sides over all $s \in C$ yields

$$f^\star \geq f(x^{(k-1)}) + \min_{s \in C} \nabla f(x^{(k-1)})^T (y - x^{(k-1)})$$

Rearranged, this gives the duality gap above

LAS

Convergence Results

Following Jaggi (2011), define the curvature constant of f over C:

$$M = \max_{\substack{x,s,y \in C \\ y=(1-\gamma)x+\gamma s}} \frac{2}{\gamma^2}\Big(f(y) - f(x) - \nabla f(x)^T(y-x)\Big)$$

(Above we restrict $\gamma \in [0,1]$.) Note that $\kappa = 0$ when f is linear. The quantity $f(y) - f(x) - \nabla f(x)^T(y-x)$ is called the Bregman divergence defined by f

Theorem: Conditional gradient method using fixed step sizes $\gamma_k = 2/(k+1)$, $k = 1, 2, 3, \ldots$ satisfies

$$f(x^{(k)}) - f^\star \leq \frac{2M}{k+2}$$

Hence the number of iterations needed to achieve $f(x^{(k)}) - f^\star \leq \epsilon$ is $O(1/\epsilon)$

LAS

Convergence Results

This matches the known rate for projected gradient descent when ∇f is Lipschitz, but how do the assumptions compare?. In fact, if ∇f is Lipschitz with constant L then $M \leq \operatorname{diam}^2(C) \cdot L$, where

$$\operatorname{diam}(C) = \max_{x,s \in C} \|x - s\|_2$$

To see this, recall that ∇f Lipschitz with constant L means

$$f(y) - f(x) - \nabla f(x)^T (y - x) \leq \frac{L}{2} \|y - x\|_2^2$$

Maximizing over all $y = (1 - \gamma)x + \gamma s$, and multiplying by $2/\gamma^2$,

$$M \leq \max_{\substack{x,s,y \in C \\ y=(1-\gamma)x+\gamma s}} \frac{2}{\gamma^2} \cdot \frac{L}{2} \|y - x\|_2^2 = \max_{x,s \in C} L \|x - s\|_2^2$$

and the bound follows. Essentially, assuming a bounded curvature is no stronger than what we assumed for proximal gradient

ALLAS

Step Size Variants

- Convergence analysis:

$$\gamma_k = 2/(k+1)$$

- Line Search:

$$\gamma_k = \text{argmin}_{\gamma \in [0,1]} f((1-\gamma)x_k + \gamma s_k)$$

(Can also be done using backtracking line search)

- Gap based:

$$\gamma_k = \min\left(\frac{f(x_k)}{L||s_k - x_k||^2}, 1\right)$$

(L is the Lipschitz constant of f)

Frank Wolfe Algorithm

Input : initial guess $\boldsymbol{x}_0$, tolerance $\delta > 0$

$$\textbf{For } t = 0, 1, \ldots \textbf{ do} \tag{2}$$

$$\boldsymbol{s}_t \in \arg\max_{\boldsymbol{s} \in \mathcal{D}} \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{s} \rangle \tag{3}$$

$$\boldsymbol{d}_t = \boldsymbol{s}_t - \boldsymbol{x}_t \tag{4}$$

$$g_t = -\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle \tag{5}$$

$$\textbf{If } g_t < \delta : \tag{6}$$

// exit if gap is below tolerance

$$\textbf{return } \boldsymbol{x}_t \tag{7}$$

Variant 1 : set step size as

$$\gamma_t = \min \left\{ \frac{g_t}{L \|\boldsymbol{d}_t\|^2}, 1 \right\} \tag{8}$$

Variant 2 : set step size by line search

$$\gamma_t = \arg\min_{\gamma \in [0,1]} f(\boldsymbol{x}_t + \gamma \boldsymbol{d}_t) \tag{9}$$

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma_t \boldsymbol{d}_t \, . \tag{10}$$

$$\textbf{end For loop} \tag{11}$$

$$\textbf{return } \boldsymbol{x}_t \tag{12}$$

Convergence for Non Convex Objectives

- Define $g_k = \max_{s \in \mathcal{C}} \langle s - x_k, -\nabla f(x_k) \rangle$
- Then a point x_k is a stationary point for constrained optimization if and only if $g_k = 0$.
- Then (Simon Lacoste-Julien, 2017: Convergence rates of Frank Wolfe for Non-Convex Objectives) shoed that:

$$\min_{0 \le i \le t} g_i \le \frac{\max(2h_0, L\text{diam}(\mathcal{C})^2)}{\sqrt{t+1}}$$

  where $h_0 = f(x_0) - \min_{x \in \mathcal{C}} f(x)$ is the initial global suboptimality.
- Similar result also holds for projected gradient descent.

More Results and Additional Reading

- Lower Bound: Conditional Gradient Analysis is tight. For any $x_0 \in \mathbb{R}^d$ and for $1 \leq k \leq d/2 - 1$, there exists a L smooth convex function and convex set $\mathcal{C}$ with diameter D s.t. for any algorithm of f that computes local gradients of f and does a linear minimization over $\mathcal{C}$, we have $f(x_k) - f_* \geq \frac{LD^2}{8(k+1)}, \forall k \geq 1$
- Additionally, Linear Convergence, i.e. $O(\log 1/\epsilon)$ can be shown under certain cases (domain is given by a polyhedron or optimum lies in the interior)

BS16 A. Beck and S. Shtern, "Linearly convergent away-step conditional gradient for non-strongly convex functions," Mathematical Programming, 1–27, 2016.

LJ15 S. Lacoste-Julien and M. Jaggi, "On the Global Linear Convergence of Frank-Wolfe Optimization Variants," NIPS, 2015.

Projected Gradient vs Conditional Gradient: Takeaways

- Projected Gradient Descent requires optimizing a quadratic function over the constraints

Projected Gradient vs Conditional Gradient: Takeaways

- Projected Gradient Descent requires optimizing a quadratic function over the constraints
- While it is easy for a few constraints, several common constraints do not admit easy projection operators

Projected Gradient vs Conditional Gradient: Takeaways

- Projected Gradient Descent requires optimizing a quadratic function over the constraints
- While it is easy for a few constraints, several common constraints do not admit easy projection operators
- Conditional Gradient Descent requires solving a linear program over constraints

Projected Gradient vs Conditional Gradient: Takeaways

- Projected Gradient Descent requires optimizing a quadratic function over the constraints
- While it is easy for a few constraints, several common constraints do not admit easy projection operators
- Conditional Gradient Descent requires solving a linear program over constraints
- This is easy for a large class of constraints (e.g. combinatorial constraints, polyhedral constraints etc.)

Projected Gradient vs Conditional Gradient: Takeaways

- Projected Gradient Descent requires optimizing a quadratic function over the constraints
- While it is easy for a few constraints, several common constraints do not admit easy projection operators
- Conditional Gradient Descent requires solving a linear program over constraints
- This is easy for a large class of constraints (e.g. combinatorial constraints, polyhedral constraints etc.)
- In contrast, conditional gradient can be slower in terms of convergence

Projected Gradient vs Conditional Gradient: Takeaways

- Projected Gradient Descent requires optimizing a quadratic function over the constraints
- While it is easy for a few constraints, several common constraints do not admit easy projection operators
- Conditional Gradient Descent requires solving a linear program over constraints
- This is easy for a large class of constraints (e.g. combinatorial constraints, polyhedral constraints etc.)
- In contrast, conditional gradient can be slower in terms of convergence
- Takeaways: If projection is easy, use projected gradient descent. Else, conditional gradient is the go to algorithm!