CS6301: Optimization in Machine Learning

Lecture 9: Proximal and Projected Gradient Descent

Rishabh Iyer

Department of Computer Science
University of Texas, Dallas
`https://sites.google.com/view/cs-6301-optml/home`

February 12th, 2020

Project and Assignment

- Project Deadline 1: Finalize on your Project Topics and partners: **February 15th 2020**.
- Projects can be done in Groups with 1-3 students per group
- You need to upload the following:
  - A Project Proposal File with a) Team members, b) Introduction and Motivation of the Project, and c) Expected Outcomes
  - A 5-7 slide summary of this for each group. You will have around 5 mins to present this on Monday (and possibly Wednesday) next week

Outline

- Summary of Results for Gradient Descent: Continuous, Smooth and Strong Convex & Accelerated Gradient Descent
- Proximal Gradient Descent
- Constrained Optimization: Lagrange Multipliers
- Projected Gradient Descent

Summary of Upper Bounds

- Lipschitz continuous functions (C): R^2B^2/ϵ^2 iterations
- Lipschitz continuous functions + Smooth (CS): $2B^2/\epsilon - 1$ iterations
- Smooth Functions GD (SGD): $\frac{R^2L}{\epsilon}$ iterations.
- Smooth Functions AGD (SAGD): $\sqrt{\frac{2LR^2}{\epsilon}}$ iterations
- Smooth + Strongly Convex (SS): With $\gamma = 1/L$, achieve an ϵ-approximate solution in $\frac{L}{\mu}\log(\frac{R^2L}{2\epsilon})$ iterations.
- Concrete examples. Let $L = B = 10, R = 1, \mu = 1$. Then we have the:
  - $\epsilon = 0.1$, C: 10000, CS: 2000, SGD = 50, SAGD = 14.4, SS = 8.49 iterations
  - $\epsilon = 0.01$, C: 1000000, CS: 20000, SGD = 500, SAGD: 44.72, SS = 13.49 iterations
  - $\epsilon = 0.001$, C: 100000000, CS: 200000, SGD = 5000, SAGD = 141.42, SS = 18.49 iterations

Lower Bounds (No Proof)

- Case I: Lipschitz Continuous: Any black-box procedure will have an error of at least $\frac{RB}{2(1+\sqrt{T})}$ (GD: $\frac{RB}{\sqrt{T}}$)
- Case II: Lipschitz Continuous + Strongly Convex: Any black-box procedure have an error of at least $\frac{B^2}{2\mu T}$, (GD: $\frac{2B^2}{\mu(T+1)}$)
- Case III: Smooth: Any black box procedure have an error of at least $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ (GD: $\frac{LR^2}{2T}$, AGD: $\frac{LR^2}{T^2}$
- Case IV: Smooth + Strongly Convex: Define $\kappa = \frac{L}{\mu}$. Then Any black box procedure will have an error of at least $\frac{\mu}{2}(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^{2(T-1)}$ (GD: $\frac{L}{2}(1-\frac{\mu}{L})^T = \frac{L}{2}(\frac{\kappa-1}{\kappa})^T$)

In Practice

- Tuning the Learning rate

In Practice

- Tuning the Learning rate
- Different Line Search based Procedures

In Practice

- Tuning the Learning rate
- Different Line Search based Procedures
- Accelerated Gradient Descent

Lipschitz Continuous Functions

- Plain Lipschitz Continuous (without strong convexity) can be very slow for converergence!

Lipschitz Continuous Functions

- Plain Lipschitz Continuous (without strong convexity) can be very slow for converergence!
- Many practical ML problems such as L1 Regularized Logistic Regression are Lipschitz Continuous

Lipschitz Continuous Functions

- Plain Lipschitz Continuous (without strong convexity) can be very slow for converergence!
- Many practical ML problems such as L1 Regularized Logistic Regression are Lipschitz Continuous
- Can we do better than the $O(1/\epsilon^2)$ convergence?

Lipschitz Continuous Functions

- Plain Lipschitz Continuous (without strong convexity) can be very slow for converergence!
- Many practical ML problems such as L1 Regularized Logistic Regression are Lipschitz Continuous
- Can we do better than the $O(1/\epsilon^2)$ convergence?
- Yes, we can. However we need to make some assumptions! (Recall the lower bound result?)

Lipschitz Continuous Functions

- Plain Lipschitz Continuous (without strong convexity) can be very slow for converergence!
- Many practical ML problems such as L1 Regularized Logistic Regression are Lipschitz Continuous
- Can we do better than the $O(1/\epsilon^2)$ convergence?
- Yes, we can. However we need to make some assumptions! (Recall the lower bound result?)
- The assumption is the function that causes the non-differentiability (for example, L1 norm) should be *simple*.

Proximal Gradient Descent

- Recall Gradient Descent as: $x_{t+1} = x_t - \gamma \nabla f(x_t)$.

Proximal Gradient Descent

- Recall Gradient Descent as: $x_{t+1} = x_t - \gamma \nabla f(x_t)$.
- This is equivalent to:

$$x_{t+1} = \text{argmin}_x f(x_t) + \nabla f(x_t)^T (x - x_t) + \frac{1}{2\gamma} ||x - x_t||^2$$

Proximal Gradient Descent

- Recall Gradient Descent as: $x_{t+1} = x_t - \gamma \nabla f(x_t)$.
- This is equivalent to:

$$x_{t+1} = \text{argmin}_x f(x_t) + \nabla f(x_t)^T (x - x_t) + \frac{1}{2\gamma} ||x - x_t||^2$$

- Now consider the optimization problem $\min_x [f(x) + h(x)]$ where h is a non-differentiable function.

Proximal Gradient Descent

- Recall Gradient Descent as: $x_{t+1} = x_t - \gamma \nabla f(x_t)$.
- This is equivalent to:

$$x_{t+1} = \text{argmin}_x f(x_t) + \nabla f(x_t)^T (x - x_t) + \frac{1}{2\gamma} ||x - x_t||^2$$

- Now consider the optimization problem $\min_x [f(x) + h(x)]$ where h is a non-differentiable function.
- The update then becomes:

$$x_{t+1} = \text{argmin}_x f(x_t) + \nabla f(x_t)^T (x - x_t) + \frac{1}{2\gamma} ||x - x_t||^2 + h(x)$$

Proximal Gradient Descent

- Recall Gradient Descent as: $x_{t+1} = x_t - \gamma \nabla f(x_t)$.
- This is equivalent to:

$$x_{t+1} = \text{argmin}_x f(x_t) + \nabla f(x_t)^T (x - x_t) + \frac{1}{2\gamma} ||x - x_t||^2$$

- Now consider the optimization problem $\min_x [f(x) + h(x)]$ where h is a non-differentiable function.
- The update then becomes:

$$x_{t+1} = \text{argmin}_x f(x_t) + \nabla f(x_t)^T (x - x_t) + \frac{1}{2\gamma} ||x - x_t||^2 + h(x)$$

- After some manipulation, the update becomes:

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

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!

How easy is it to compute the Prox operator?

- This depends on the specific function at hand.
- Lets consider some examples:
  - $h(x) = a^T x$
  - $h(x) = c$ (a constant)
  - $h(x) = \lambda ||x||^2$
  - $h(x) = -\lambda \log(x)$ (defined only when $x \geq 0$)
  - $h(x) = \frac{1}{2} x^T A x + b^T x + c$, A is positive semi-definite
  - $h(x) = \mu x$ if $x \geq 0$ or ∞ else.
- How to compute the Prox? Solve the optimization problem:

$$\text{prox}_t(x) = \text{argmin}_z \frac{1}{2t} ||z - x||^2 + h(z) = \text{argmin}_z \frac{1}{2} ||z - x||^2 + th(z)$$

- Using the optimality conditions (since h is convex):

$$z - x + th'(z) = 0$$

- What about if the function is non-differentiable?

Equivalent way of looking at Prox

- Some textbooks define prox as:

$$\text{prox-new}_h(x) = \text{argmin}_z \frac{1}{2}||z - x||^2 + h(z)$$

- Note that with our definition, $\text{prox}_t(x) = \text{prox-new}_{th}(x)$
- We shall use this definition of prox for the rest of this class.

Main ideas for computing Prox

- The main ideas of computing Prox operator are:
  1. If $h'(x) = 0$ for a convex function h, the x must be one of its minimizers.
  2. If a minimizer of a convex function exists and is not attained at any point of differentiability, it must be attained at a point of non-differentiability.
- Try computing the Prox operator for some more functions (homework)
  - $h(x) = \lambda x^3, x \geq 0$ and $-\infty$ otherwise.
  - $h(x) = 0$, for $x \neq 0$ and $-\lambda$ if $x = 0$
  - $h(x) = 0$, for $x \neq 0$ and λ if $x = 0$

Computing Prox when h is non-differentiable

- We want to solve: $\text{prox}_f(x) = \text{argmin}_z \frac{1}{2}||z - x||^2 + h(z)$
- If h is differentiable, we have the optimality condition as: $z - x + \nabla h(z) = 0$
- If h is non-differentiable, denote $d_h \in \partial h(z)$ as the sub-gradient. The optimality condition is $0 \in z - x + d_h(z)$
- Consider $h(z) = \lambda||z||_1$
- Its easy to see that $x - z = \lambda d$ where $d_i = \{1\}$ if $z_i > 0$, $d_i = \{-1\}$ if $z_i < 0$ and $d_i \in [-1, 1]$ if $z_i = 0$.
- In other words, the sub-gradient optimality conditions are $x_i - z_i = \lambda \text{sign}(z_i)$ if $z_i \neq 0$ and $|x_i - z_i| \leq \lambda$ if $z_i = 0$
- Its easy to see that:

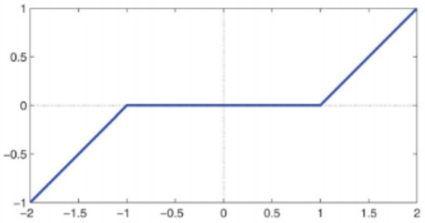
$$z_i = \begin{cases} x_i - \lambda & \text{if } x_i > \lambda \\ 0 & \text{if } -\lambda < x_i < \lambda \\ x_i + \lambda & \text{if } x_i < -\lambda \end{cases}$$

Computing Prox for L1 Norm

- From the previous slide, the Prox opterator is:

$$z_i = \begin{cases} x_i - \lambda & \text{if } x_i > \lambda \\ 0 & \text{if } -\lambda < x_i < \lambda \\ x_i + \lambda & \text{if } x_i < -\lambda \end{cases}$$

- In other words $\text{prox}^i_{\lambda||x||_1} = [|x_i| - \lambda]_+ \text{sign}(x_i)$ is the soft-thresholding operator!

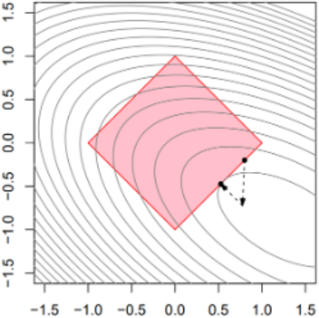


Summary of Proximal Gradient Descent

- Proximal Gradient Descent becomes gradient descent if $h = 0$
- If f is 0 (i.e. no smooth function), one can minimize a non-differentiable function h as long as the prox operator is easy to compute!
- Key to Proximal GD: Being able to compute Proximal operator!
- What is Prox can be efficiently computed approximately?
- There are papers (Schmidt et al 2011, Inexact Proximal Gradient Methods) where Prox is computed approximately but one can still derive the convergence rates if the errors due to approximation can be controlled!
- Accelerated Proximal GD: Similar to GD, one can accelerate GD to get optimal convergence rates! (Beck and Teboulle 2008)

Detour: 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 $P_{\mathcal{C}}(x) = \text{argmin}_{y \in \mathcal{C}} \frac{1}{2}||x - y||^2$



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!

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 $\mathcal{C}_f = \{x \mid f(x) \leq c\}$
  - $f(x) = a^T x$
  - $f(x) = ||x||^2$
  - $f(x) = ||x||_1$
  - $f(x) = ||x - x_0||^2$
  - $f(x) = x^T Ax + bx + c$
  - $f(x) = ||x||_\infty$ $f($