Optimization in Machine Learning

Lecture 16: Accelerated, Stochastic and Generalized Gradient Descent

Ganesh Ramakrishnan

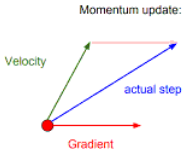
Department of Computer Science
Dept of CSE, IIT Bombay
https://www.cse.iitb.ac.in/~ganesh

March, 2025

Outline of next few topics

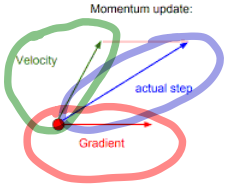
- Algorithms for Optimization: First Order and thereafter
- Accelerated Gradient Descent
- Stochastic Gradient Descent
- Accelerated Stochastic Variants: Hybrid, Adam, Adagrad, RMSProp, *etc.*
- Generalized/Proximal Gradient Descent
- Constrained Optimization and Projected Gradient Descent

[Recap] Efficient alternatives to GD: Momentum



- **Momentum Accelerated GD:** $x_{act} = x_{old} + \hat{\mathbf{v}}$ where $\hat{\mathbf{v}} = \mu \mathbf{v} - \eta_k \frac{\partial f}{\partial x}$

[Recap] Efficient alternatives to GD: Momentum



- **Momentum Accelerated GD:** $x_{act} = x_{old} + \hat{v}$ where $\hat{v} = \mu v - \eta_k \frac{\partial f}{\partial x}$

Accounting for momentum

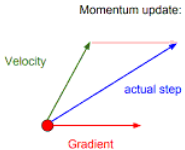
Multipicative factor associated with the history (captured through the velocity)

$\mu > 1$?

$\mu < 0$?

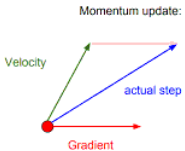
$\mu \in (0, 1)$ ✓

[Recap] Efficient alternatives to GD: Momentum



- **Momentum Accelerated GD:** $x_{act} = x_{old} + \hat{\mathbf{v}}$ where $\hat{\mathbf{v}} = \mu \mathbf{v} - \eta_k \frac{\partial f}{\partial x}$
- $\mathbf{v}$ plays the role of velocity - it is the direction and speed at which the parameter x_i moves through parameter space.
- Set to an exponentially decaying average of the negative gradient: Decay factor will be $\mu \in [0, 1)$

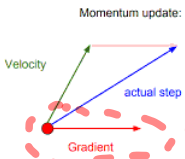
[Recap] Efficient alternatives to GD: Momentum



- **Momentum Accelerated GD:** $x_{act} = x_{old} + \hat{\mathbf{v}}$ where $\hat{\mathbf{v}} = \mu \mathbf{v} - \eta_k \frac{\partial f}{\partial x}$
- $\mathbf{v}$ plays the role of velocity - it is the direction and speed at which the parameter x_i moves through parameter space.
- Set to an exponentially decaying average of the negative gradient: Decay factor will be $\mu \in [0, 1)$. Common values of μ used in practice are .5, .9, and .99
- Physical analogy $\Rightarrow$ negative gradient is a force moving a particle through parameter space, according to Newton's laws of motion
- Step size largest when

Suggestion: When the gradient and velocity are in the same direction
More generally: When successive gradients are in the same direction

[Recap] Efficient alternatives to GD: Momentum



option 2: $\frac{\partial f}{\partial x_{new}}$

- **Momentum Accelerated GD:** $x_{act} = x_{old} + \hat{\mathbf{v}}$ where $\hat{\mathbf{v}} = \mu \mathbf{v} - \eta_k \frac{\partial f}{\partial x}$

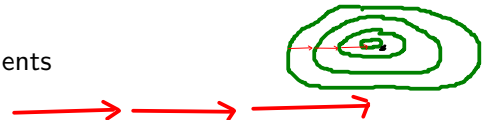
option 1: $\frac{\partial f}{\partial x_{old}}$

Q: x is x_{old} OR x_{new}

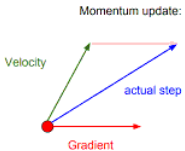
- $\mathbf{v}$ plays the role of velocity - it is the direction and speed at which the parameter x_i moves through parameter space.
- Set to an exponentially decaying average of the negative gradient: Decay factor will be $\mu \in [0, 1)$. Common values of μ used in practice are .5, .9, and .99.

fading history

- Physical analogy $\Rightarrow$ negative gradient is a force moving a particle through parameter space, according to Newton's laws of motion
- Step size largest when many successive gradients point in exactly the same direction



[Recap] Efficient alternatives to GD: Momentum



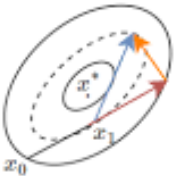
- **Momentum Accelerated GD:** $x_{act} = x_{old} + \hat{\mathbf{v}}$ where $\hat{\mathbf{v}} = \mu \mathbf{v} - \eta_k \frac{\partial f}{\partial x}$
- $\mathbf{v}$ plays the role of velocity - it is the direction and speed at which the parameter x_i moves through parameter space.
- Set to an exponentially decaying average of the negative gradient: Decay factor will be $\mu \in [0, 1)$. Common values of μ used in practice are .5, .9, and .99.

- Physical analogy $\Rightarrow$ negative gradient is a force moving a particle through parameter space, according to Newton's laws of motion
- Step size largest when many successive gradients point in exactly the same direction

[Recap] Momentum and its variants

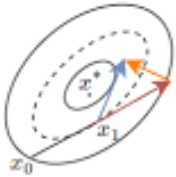
- Modified Nesterov momentum $\Rightarrow$ gradient is evaluated after the current velocity is applied. Speeds up rate of convergence from $O(1/T)$ to $O(1/T^2)$

Polyak's Momentum



$$x_{t+1} = x_t - \alpha \nabla f(x_t) + \mu(x_t - x_{t-1})$$

Nesterov Momentum

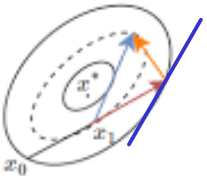


$$x_{t+1} = x_t + \mu(x_t - x_{t-1}) - \gamma \nabla f(x_t + \mu(x_t - x_{t-1}))$$

[Recap] Momentum and its variants

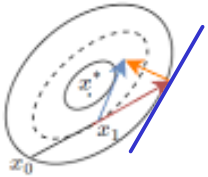
- Modified Nesterov momentum ⇒ gradient is evaluated after the current velocity is applied. Speeds up rate of convergence from $O(1/T)$ to $O(1/T^2)$ for L-continuity and strong convexity case

Polyak's Momentum



$$x_{t+1} = x_t - \alpha \nabla f(x_t) + \mu(x_t - x_{t-1})$$

Nesterov Momentum



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

$-\alpha \nabla f(x_t)$ ← Difference → $-r\nabla f(x_t + u(x_t - x_{t-1})) = -r\nabla f(x_{new})$

Attempt 1: Polyak's Heavy Ball Momentum

- Recall standard gradient descent: $x_{t+1} = x_t - \alpha_t \nabla f(x_t)$
- Polyak's Heavy Ball Momentum adds inertia: $x_{t+1} = x_t - \alpha_t \nabla f(x_t) + \mu_t (x_t - x_{t-1})$
- Heavy Ball result: For smooth + strongly convex functions, the heavy ball algorithm converges in $\frac{R^2}{2}\left(1 - \sqrt{\frac{1}{\kappa}}\right)^T = \frac{L}{2}\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}}\right)^T$ in comparison with GD's $\frac{R^2}{2}\left(1 - \frac{1}{\kappa}\right)^T = \frac{L}{2}\left(\frac{\kappa-1}{\kappa}\right)^T$ iterations.
- Heavy Ball momentum is not optimal for the Smooth case (though it is optimal for the strongly convex + smooth class)

Attempt 1: Polyak's Heavy Ball Momentum

Conservatively exploits curvature

In this term which is standard GD the curvature is not being exploited

Velocity exploits curvature empirically

Exponential Decay μ_t

- Recall standard gradient descent: $x_{t+1} = x_t - \alpha_t \nabla f(x_t)$
- Polyak's Heavy Ball Momentum adds inertia: $x_{t+1} = x_t - \alpha_t \nabla f(x_t) + \mu_t (x_t - x_{t-1})$
- Heavy Ball result: For smooth + strongly convex functions, the heavy ball algorithm converges in $\frac{R^2}{2}\left(1 - \sqrt{\frac{1}{\kappa}}\right)^T = \frac{L}{2}\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}}\right)^T$ in comparison with GD's $\frac{R^2}{2}\left(1 - \frac{1}{\kappa}\right)^T = \frac{L}{2}\left(\frac{\kappa-1}{\kappa}\right)^T$ iterations.

Comparable and Commensurate with Lower Bound

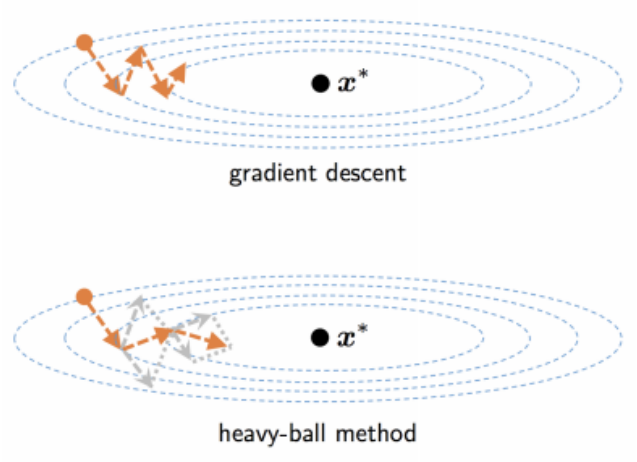
- Heavy Ball momentum is not optimal for the Smooth case (though it is optimal for the strongly convex + smooth class)

+ vanilla convexity

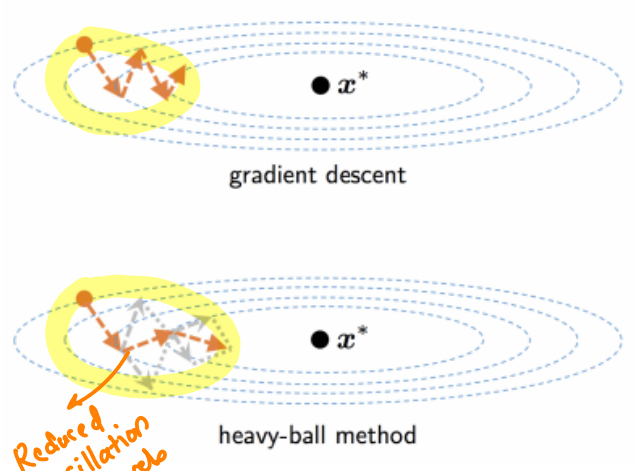
We have both an upper bound and a lower bound on the curvature

We might have to be a bit careful in going aggressively in exploiting curvature

GD vs Momentum



GD vs Momentum



Attempt 2: Nesterov's Accelerated Gradient Descent

- There is a gap of a factor of T for the (Lipschitz) Smooth case! $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ vs $\frac{LR^2}{2T}$!

Attempt 2: Nesterov's Accelerated Gradient Descent

(Post velocity update gradient computation)

- There is a gap of a factor of T for the (Lipschitz) Smooth case! $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ vs $\frac{LR^2}{2T}$!

Target

In theory this gap is fixed

Attempt 2: Nesterov's Accelerated Gradient Descent

- There is a gap of a factor of T for the (Lipschitz) Smooth case! $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ vs $\frac{LR^2}{2T}$!
- Nesterov's accelerated algorithm fixes this (a very unintuitive algorithm and proof procedure – we will not prove it in the class (For proof, see additional optional slides & https://www.cse.iitb.ac.in/~ganesh/cs769/notes/enotes/24-23-10-2018-generalized-proximal-projected-gradientdescent-examples-geometry-convergence-accelerated-annotated.pdf)

Attempt 2: Nesterov's Accelerated Gradient Descent

- There is a gap of a factor of T for the (Lipschitz) Smooth case! $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ vs $\frac{LR^2}{2T}$!
- Nesterov's accelerated algorithm fixes this (a very unintuitive algorithm and proof procedure – we will not prove it in the class (For proof, see additional optional slides & https://www.cse.iitb.ac.in/~ganesh/cs769/notes/enotes/24-23-10-2018-generalized-proximal-projected-gradientdescent-examples-geometry-convergence-accelerated-annotated.pdf))

Proof is generic enough for so-called "generalized gradient descent" (proximal gradient descent is an instance of generalized gradient descent)

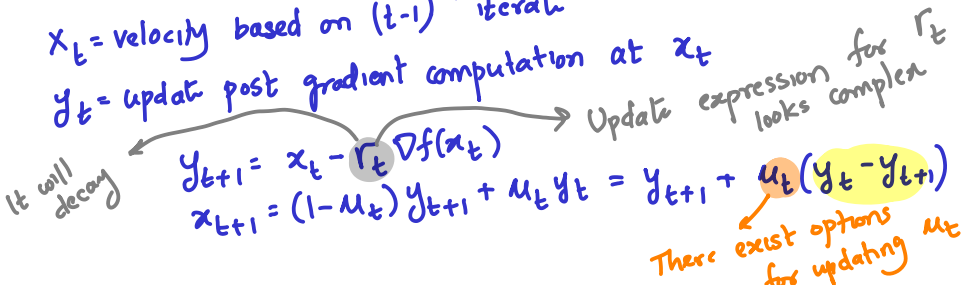
Attempt 2: Nesterov's Accelerated Gradient Descent

- There is a gap of a factor of T for the (Lipschitz) Smooth case! $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ vs $\frac{LR^2}{2T}$!
- Nesterov's accelerated algorithm fixes this (a very unintuitive algorithm and proof procedure – we will not prove it in the class (For proof, see additional optional slides & `https://www.cse.iitb.ac.in/~ganesh/cs769/notes/enotes/24-23-10-2018-generalized-proximal-projected-gradientdescent-examples-geometry-convergence-accelerated-annotated.pdf`)
- The algorithm is as follows.

RECAP: HOMEWORK

Attempt 2: Nesterov's Accelerated Gradient Descent

- There is a gap of a factor of T for the (Lipschitz) Smooth case! $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ vs $\frac{LR^2}{2T}$!
- Nesterov's accelerated algorithm fixes this (a very unintuitive algorithm and proof procedure – we will not prove it in the class (For proof, see additional optional slides & https://www.cse.iitb.ac.in/~ganesh/cs769/notes/enotes/24-23-10-2018-generalized-proximal-projected-gradientdescent-examples-geometry-convergence-accelerated-annotated.pdf)
- The algorithm is as follows.



Attempt 2: Nesterov's Accelerated Gradient Descent

- There is a gap of a factor of T for the (Lipschitz) Smooth case! $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ vs $\frac{LR^2}{2T}$!
- Nesterov's accelerated algorithm fixes this (a very unintuitive algorithm and proof procedure – we will not prove it in the class (For proof, see additional optional slides & https://www.cse.iitb.ac.in/~ganesh/cs769/notes/enotes/24-23-10-2018-generalized-proximal-projected-gradientdescent-examples-geometry-convergence-accelerated-annotated.pdf)
- The algorithm is as follows.
  - Define $\lambda_0 = 0, \lambda_t = \frac{1+\sqrt{1+4\lambda_{t-1}^2}}{2}$ and $\gamma_t = \frac{1-\lambda_t}{\lambda_{t+1}}$. Note $\gamma_t \geq 0$

Attempt 2: Nesterov's Accelerated Gradient Descent

- There is a gap of a factor of T for the (Lipschitz) Smooth case! $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ vs $\frac{LR^2}{2T}$!
- Nesterov's accelerated algorithm fixes this (a very unintuitive algorithm and proof procedure – we will not prove it in the class (For proof, see additional optional slides & https://www.cse.iitb.ac.in/~ganesh/cs769/notes/enotes/24-23-10-2018-generalized-proximal-projected-gradientdescent-examples-geometry-convergence-accelerated-annotated.pdf)
- The algorithm is as follows.
  - Define $\lambda_0 = 0$, $\lambda_t = \frac{1+\sqrt{1+4\lambda_{t-1}^2}}{2}$ and $\gamma_t = \frac{1-\lambda_t}{\lambda_{t+1}}$. Note $\gamma_t \geq 0$ (check.)

Inspired by
1) sufficient descrease condition (Armijo conditions, Goldstein conditions in optional slides)
2) interpolation on step size across iterations to help satisfy these conditions quickly

Attempt 2: Nesterov's Accelerated Gradient Descent

- There is a gap of a factor of T for the (Lipschitz) Smooth case! $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ vs $\frac{LR^2}{2T}$!
- Nesterov's accelerated algorithm fixes this (a very unintuitive algorithm and proof procedure – we will not prove it in the class (For proof, see additional optional slides & https://www.cse.iitb.ac.in/~ganesh/cs769/notes/enotes/24-23-10-2018-generalized-proximal-projected-gradientdescent-examples-geometry-convergence-accelerated-annotated.pdf)
- The algorithm is as follows.
  - Define $\lambda_0 = 0, \lambda_t = \frac{1+\sqrt{1+4\lambda_{t-1}^2}}{2}$ and $\gamma_t = \frac{1-\lambda_t}{\lambda_{t+1}}$. Note $\gamma_t \geq 0$
  - Initialize $x_1 = y_1$ as an arbitrary point

Attempt 2: Nesterov's Accelerated Gradient Descent

- There is a gap of a factor of T for the (Lipschitz) Smooth case! $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ vs $\frac{LR^2}{2T}$!
- Nesterov's accelerated algorithm fixes this (a very unintuitive algorithm and proof procedure – we will not prove it in the class (For proof, see additional optional slides & https://www.cse.iitb.ac.in/~ganesh/cs769/notes/enotes/24-23-10-2018-generalized-proximal-projected-gradientdescent-examples-geometry-convergence-accelerated-annotated.pdf)
- The algorithm is as follows.
  - Define $\lambda_0 = 0, \lambda_t = \frac{1+\sqrt{1+4\lambda_{t-1}^2}}{2}$ and $\gamma_t = \frac{1-\lambda_t}{\lambda_{t+1}}$. Note $\gamma_t \geq 0$
  - Initialize $x_1 = y_1$ as an arbitrary point

Attempt 2: Nesterov's Accelerated Gradient Descent

- There is a gap of a factor of T for the (Lipschitz) Smooth case! $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ vs $\frac{LR^2}{2T}$!
- Nesterov's accelerated algorithm fixes this (a very unintuitive algorithm and proof procedure – we will not prove it in the class (For proof, see additional optional slides & https://www.cse.iitb.ac.in/~ganesh/cs769/notes/enotes/24-23-10-2018-generalized-proximal-projected-gradientdescent-examples-geometry-convergence-accelerated-annotated.pdf)
- The algorithm is as follows.
  - Define $\lambda_0 = 0, \lambda_t = \frac{1+\sqrt{1+4\lambda_{t-1}^2}}{2}$ and $\gamma_t = \frac{1-\lambda_t}{\lambda_{t+1}}$. Note $\gamma_t \geq 0$
  - Initialize $x_1 = y_1$ as an arbitrary point
  - Step 1: $y_{t+1} = x_t - \gamma_t \nabla f(x_t)$ (like normal GD; γ_t is learning rate, *e.g.*, $\frac{1}{L}$)

Attempt 2: Nesterov's Accelerated Gradient Descent

- There is a gap of a factor of T for the (Lipschitz) Smooth case! $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ vs $\frac{LR^2}{2T}$!
- Nesterov's accelerated algorithm fixes this (a very unintuitive algorithm and proof procedure – we will not prove it in the class (For proof, see additional optional slides & https://www.cse.iitb.ac.in/~ganesh/cs769/notes/enotes/24-23-10-2018-generalized-proximal-projected-gradientdescent-examples-geometry-convergence-accelerated-annotated.pdf)
- The algorithm is as follows.
  - Define $\lambda_0 = 0, \lambda_t = \frac{1+\sqrt{1+4\lambda_{t-1}^2}}{2}$ and $\gamma_t = \frac{1-\lambda_t}{\lambda_{t+1}}$. Note $\gamma_t \geq 0$
  - Initialize $x_1 = y_1$ as an arbitrary point
  - Step 1: $y_{t+1} = x_t - \gamma_t \nabla f(x_t)$ (like normal GD; γ_t is learning rate, *e.g.*, $\frac{1}{L}$)

Attempt 2: Nesterov's Accelerated Gradient Descent

- There is a gap of a factor of T for the (Lipschitz) Smooth case! $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ vs $\frac{LR^2}{2T}$!
- Nesterov's accelerated algorithm fixes this (a very unintuitive algorithm and proof procedure – we will not prove it in the class (For proof, see additional optional slides & https://www.cse.iitb.ac.in/~ganesh/cs769/notes/enotes/24-23-10-2018-generalized-proximal-projected-gradientdescent-examples-geometry-convergence-accelerated-annotated.pdf)
- The algorithm is as follows.
  - Define $\lambda_0 = 0, \lambda_t = \frac{1+\sqrt{1+4\lambda_{t-1}^2}}{2}$ and $\gamma_t = \frac{1-\lambda_t}{\lambda_{t+1}}$. Note $\gamma_t \geq 0$
  - Initialize $x_1 = y_1$ as an arbitrary point
  - Step 1: $y_{t+1} = x_t - \gamma_t \nabla f(x_t)$ (like normal GD; γ_t is learning rate, *e.g.*, $\frac{1}{L}$)
  - Step 2: $x_{t+1} = (1-\mu_t) y_{t+1} + \mu_t y_t$ (slide a little bit further than y_{t+1} towards the previous point y_t!) - *E.g.*, $\mu_t = 2/(t+1)$. Notice similarity with heavy-ball method!

Attempt 2: Nesterov's Accelerated Gradient Descent

- There is a gap of a factor of T for the (Lipschitz) Smooth case! $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ vs $\frac{LR^2}{2T}$!
- Nesterov's accelerated algorithm fixes this (a very unintuitive algorithm and proof procedure – we will not prove it in the class (For proof, see additional optional slides & https://www.cse.iitb.ac.in/~ganesh/cs769/notes/enotes/24-23-10-2018-generalized-proximal-projected-gradientdescent-examples-geometry-convergence-accelerated-annotated.pdf)
- The algorithm is as follows.
  - Define $\lambda_0 = 0, \lambda_t = \frac{1+\sqrt{1+4\lambda_{t-1}^2}}{2}$ and $\gamma_t = \frac{1-\lambda_t}{\lambda_{t+1}}$. Note $\gamma_t \geq 0$
  - Initialize $x_1 = y_1$ as an arbitrary point
  - Step 1: $y_{t+1} = x_t - \gamma_t \nabla f(x_t)$ (like normal GD; γ_t is learning rate, *e.g.*, $\frac{1}{L}$)
  - Step 2: $x_{t+1} = (1-\mu_t)\, y_{t+1} + \mu_t\, y_t$ (slide a little bit further than y_{t+1} towards the previous point y_t!) - *E.g.*, $\mu_t = 2/(t+1)$. Notice similarity with heavy-ball method!

Inspired by
1) sufficient descrease condition (Armijo conditions, Goldstein conditions in optional slides)
2) interpolation on step size across iterations to help satisfy these conditions quickly

Velocity update for next iterate

also diminishes

$x_{t+1} = y_{t+1} + \mu_t (y_t - y_{t+1})$ (velocity)

Attempt 2: Nesterov's Accelerated Gradient Descent

- There is a gap of a factor of T for the (Lipschitz) Smooth case! $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ vs $\frac{LR^2}{2T}$!
- Nesterov's accelerated algorithm fixes this (a very unintuitive algorithm and proof procedure – we will not prove it in the class (For proof, see additional optional slides & https://www.cse.iitb.ac.in/~ganesh/cs769/notes/enotes/24-23-10-2018-generalized-proximal-projected-gradientdescent-examples-geometry-convergence-accelerated-annotated.pdf)
- The algorithm is as follows.
  - Define $\lambda_0 = 0, \lambda_t = \frac{1+\sqrt{1+4\lambda_{t-1}^2}}{2}$ and $\gamma_t = \frac{1-\lambda_t}{\lambda_{t+1}}$. Note $\gamma_t \geq 0$
  - Initialize $x_1 = y_1$ as an arbitrary point
  - Step 1: $y_{t+1} = x_t - \gamma_t \nabla f(x_t)$ (like normal GD; γ_t is learning rate, *e.g.*, $\frac{1}{L}$)
  - Step 2: $x_{t+1} = (1-\mu_t) y_{t+1} + \mu_t y_t$ (slide a little bit further than y_{t+1} towards the previous point y_t!) - *E.g.*, $\mu_t = 2/(t+1)$. Notice similarity with heavy-ball method!
- Theorem (Nesterov 1983): If f is convex and L-smooth, $f(y_T) - f(x^*) \leq \frac{2LR^2}{T^2}$

Attempt 2: Nesterov's Accelerated Gradient Descent

- There is a gap of a factor of T for the (Lipschitz) Smooth case! $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ vs $\frac{LR^2}{2T}$!
- Nesterov's accelerated algorithm fixes this (a very unintuitive algorithm and proof procedure – we will not prove it in the class (For proof, see additional optional slides & https://www.cse.iitb.ac.in/~ganesh/cs769/notes/enotes/24-23-10-2018-generalized-proximal-projected-gradientdescent-examples-geometry-convergence-accelerated-annotated.pdf)
- The algorithm is as follows.
  - Define $\lambda_0 = 0, \lambda_t = \frac{1+\sqrt{1+4\lambda_{t-1}^2}}{2}$ and $\gamma_t = \frac{1-\lambda_t}{\lambda_{t+1}}$. Note $\gamma_t \geq 0$
  - Initialize $x_1 = y_1$ as an arbitrary point
  - Step 1: $y_{t+1} = x_t - \gamma_t \nabla f(x_t)$ (like normal GD; γ_t is learning rate, *e.g.*, $\frac{1}{L}$)
  - Step 2: $x_{t+1} = (1-\mu_t)\, y_{t+1} + \mu_t\, y_t$ (slide a little bit further than y_{t+1} towards the previous point y_t!) - *E.g.*, $\mu_t = 2/(t+1)$. Notice similarity with heavy-ball method!
- Theorem (Nesterov 1983): If f is convex and L-smooth, $f(y_T) - f(x^*) \leq \frac{2LR^2}{T^2}$

not necessarily strongly convex

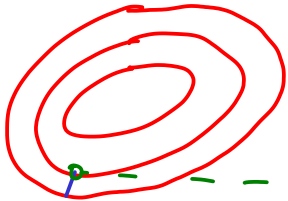
Recap: Lower bnd had $O\left(\frac{1}{T+1}\right)$

Attempt 2: Nesterov's Accelerated Gradient Descent

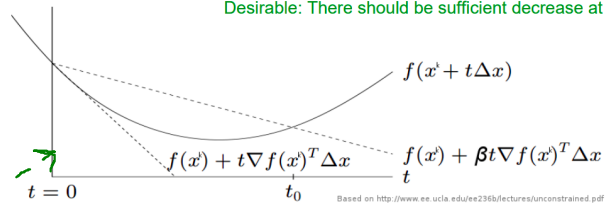
- There is a gap of a factor of T for the (Lipschitz) Smooth case! $\frac{3L}{32}\frac{R^2}{(T+1)^2}$ vs $\frac{LR^2}{2T}$!
- Nesterov's accelerated algorithm fixes this (a very unintuitive algorithm and proof procedure – we will not prove it in the class (For proof, see additional optional slides & https://www.cse.iitb.ac.in/~ganesh/cs769/notes/enotes/24-23-10-2018-generalized-proximal-projected-gradientdescent-examples-geometry-convergence-accelerated-annotated.pdf)
- The algorithm is as follows.
  - Define $\lambda_0 = 0, \lambda_t = \frac{1+\sqrt{1+4\lambda_{t-1}^2}}{2}$ and $\gamma_t = \frac{1-\lambda_t}{\lambda_{t+1}}$. Note $\gamma_t \geq 0$
  - Initialize $x_1 = y_1$ as an arbitrary point
  - Step 1: $y_{t+1} = x_t - \gamma_t \nabla f(x_t)$ (like normal GD; γ_t is learning rate, *e.g.*, $\frac{1}{L}$)
  - Step 2: $x_{t+1} = (1-\mu_t)\, y_{t+1} + \mu_t\, y_t$ (slide a little bit further than y_{t+1} towards the previous point y_t!) - *E.g.*, $\mu_t = 2/(t+1)$. Notice similarity with heavy-ball method!
- Theorem (Nesterov 1983): If f is convex and L-smooth, $f(y_T) - f(x^*) \leq \frac{2LR^2}{T^2}$
- Matches the lower bound upto contant factors!

https://colab.research.google.com/drive/1f1y8_ZkL_WS7Kfd7q-KB9Yt2VME1wV35#scrollTo=r50pPWrwLO-1

https://colab.research.google.com/drive/1f1y8_ZkL_WS7Kfd7q-KB9Yt2VME1wV35#scrollTo=r50pPWrwLO-1



Desirable: There should be sufficient decrease at each iterate



From https://moodle.iitb.ac.in/pluginfile.php/268768/mod_resource/content/4/CS769_2022___ExtraSlidesOnConvergence-annotated.pdf

① Line/Ray search will consider some step size α ($\alpha \geq 0$)

② Will next check if some sufficient decrease condition is met

③ If NOT, shrink the α !

Summary of Results so Far...

- Lipschitz continuous functions (C): R^2B^2/ϵ^2 iterations
- Lipschitz continuous functions + Strongly Convex (CS): $2B^2/\epsilon - 1$ iterations
- Smooth Functions GD (SGD): $\frac{R^2L}{\epsilon}$ iterations.
- Smooth Functions AGD (SAGD): $\sqrt{\frac{2R^2L}{\epsilon}}$ 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 following:
  - $\epsilon = 0.1$, 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

Learnings from https://colab.research.google.com/drive/1f1y8_ZkL_WS7Kfd7q-KB9Yt2VME1wV35#scrollTo=r50pPWrwLO-1
1) Initialization of \alpha matters: Armijo condition V4 helped... Even Accelerated gradient descent is somewhat reasonable in addressing this

2) Reinitializing \alpha (that is step size) matters. Else it will become too small and the steps will become meaningless

3) What value one sets the \alpha (step size) matters especially when one is doing polynomial interpolation

4) Accelerated gradient descent seems to be a principled approach to determine the direction (using velocity) as well as the step size. However, we could improve our implementation of Nestorov accelerated Gradient descent (beyond the current default Amijo condition V2) to say use V4.

For M_t the updates are exclusively for accelerated gradient descent

For f_t

Hw1: Understand the convergence proof of subgradient descent (applicable when the objective is not differentiable)

HW2: Do all the points matter equally for update. Can we select the points on which updates must be computed Leads to stochastic GD. Can we analyze stochastic GD?

(Sub)Gradient Descent: Generalization of Gradient Descent

Given a convex function $f : \mathbb{R}^n \to \mathbb{R}$, not necessarily differentiable. Subgradient method is just like gradient descent, but replacing gradients with subgradients. *I.e.*, initialize $x_{(0)}$, then repeat

$$\mathbf{x}_t = \mathbf{x}_{(t-1)} - \gamma_t \cdot \mathbf{h}_{(t-1)}, k = 1, 2, 3, \cdots$$

where $\mathbf{h}_{(t-1)}$ is **any** subgradient of f at $\mathbf{x}_{(t-1)}$. We keep track of best iterate $\mathbf{x}_t^{best}$ among $\mathbf{x}_{(1)}, \cdots, \mathbf{x}_t$:

$$f(\mathbf{x}_T^{best}) = \min_{t=1,\cdots,T} f(\mathbf{x}^t)$$

To update each $\mathbf{x}_t$, there are basically two ways to select the step size:

- Fixed step size: $\gamma_t = \gamma$ for all $t = 1, 2, 3 \cdots T$
- Diminishing step size: choose γ_t to satisfy

$$\lim_{t \to \infty} \gamma_t = 0 \ , \ \sum_{t=1}^{\infty} \gamma_t = \infty$$

(Sub)Gradient Descent: Generalization of Gradient Descent

Given a convex function $f : \mathbb{R}^n \to \mathbb{R}$, not necessarily differentiable. Subgradient method is just like gradient descent, but replacing gradients with subgradients. *I.e.*, initialize $x_{(0)}$, then repeat

$$\mathbf{x}_t = \mathbf{x}_{(t-1)} - \gamma_t \cdot \mathbf{h}_{(t-1)}, k = 1, 2, 3, \cdots$$

where $\mathbf{h}_{(t-1)}$ is **any** subgradient of f at $\mathbf{x}_{(t-1)}$. We keep track of best iterate $\mathbf{x}_t^{best}$ among $\mathbf{x}_{(1)}, \cdots, \mathbf{x}_t$:

"best" because we are no longer guaranteed descent

$$f(\mathbf{x}_T^{best}) = \min_{t=1,\cdots,T} f(\mathbf{x}^t)$$

In Gradient Descent were we guaranteed descent?

To update each $\mathbf{x}_t$, there are basically two ways to select the step size:

- Fixed step size: $\gamma_t = \gamma$ for all $t = 1, 2, 3 \cdots T$
- Diminishing step size: choose γ_t to satisfy

$$\lim_{t \to \infty} \gamma_t = 0 \ , \ \sum_{t=1}^{\infty} \gamma_t = \infty$$

Subgradient Algorithm: Convergence analysis

Given the convex function $f : \mathbb{R}^n \to \mathbb{R}$ that satisfies:

- f is Lipschitz continuous with constant $l > 0$,

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq l||\mathbf{x} - \mathbf{y}|| \text{ for all } \mathbf{x}, \mathbf{y}$$

- $||\mathbf{x}_{(1)} - \mathbf{x}_*|| \leq R$ which means it is bounded

Theorem

For a fixed step size γ, subgradient method satisfies

$$\lim_{T \to \infty} f(x_T^{best}) \leq f(\mathbf{x}_*) + \frac{l^2 T}{2}$$

For diminishing step size such as $\gamma_t = O\left(\frac{1}{\sqrt{t}}\right)$,

$$f(x_T^{best}) \leq f(\mathbf{x}_*) + O\left(\frac{1}{\sqrt{T}}\right)$$

Subgradient Descent: Convergence Analysis (contd.)

Proof:

$$\begin{aligned} ||\mathbf{x}_{(t+1)} - \mathbf{x}_*||^2 &= ||\mathbf{x}_t - \gamma_t \mathbf{h}_t - \mathbf{x}_*||^2 \\ &= ||\mathbf{x}_t - \mathbf{x}_*||^2 -2\gamma_t(\mathbf{h}_t)^T(\mathbf{x}_t - \mathbf{x}_*) + (\gamma_t)^2||\mathbf{h}_t||^2 \end{aligned}$$

By definition of the subgradient method, we have

$$f(\mathbf{x}_*) \geq f(\mathbf{x}_t) + (\mathbf{h}_t)^T(\mathbf{x}_* - \mathbf{x}_t)$$

$$-(\mathbf{h}_t)^T(\mathbf{x}_* - \mathbf{x}_t) \leq -\left(f(\mathbf{x}_t) - f(\mathbf{x}_*)\right)$$

Using this inequality for $t = 0, \ldots T$ we have

Subgradient Descent: Convergence Analysis (contd.)

Proof:

$$\begin{aligned} ||\mathbf{x}_{(t+1)} - \mathbf{x}_*||^2 &= ||\mathbf{x}_t - \gamma_t \mathbf{h}_t - \mathbf{x}_*||^2 \\ &= ||\mathbf{x}_t - \mathbf{x}_*||^2 \boxed{-2\gamma_t (\mathbf{h}_t)^T (\mathbf{x}_t - \mathbf{x}_*)} + (\gamma_t)^2 ||\mathbf{h}_t||^2 \end{aligned}$$

By definition of the subgradient method, we have

$$f(\mathbf{x}_*) \geq f(\mathbf{x}_t) + (\mathbf{h}_t)^T (\mathbf{x}_* - \mathbf{x}_t)$$

$$\boxed{-(\mathbf{h}_t)^T (\mathbf{x}_* - \mathbf{x}_t)} \leq \boxed{-\left(f(\mathbf{x}_t) - f(\mathbf{x}_*)\right)}$$

Using this inequality for $t = 0, \ldots T$ we have

Supporting hyperplane to the epigraph of the function

Subgradient Descent: Convergence Analysis (contd.)

Proof:

$$\begin{aligned} ||\mathbf{x}_{(t+1)} - \mathbf{x}_*||^2 &= ||\mathbf{x}_t - \gamma_t \mathbf{h}_t - \mathbf{x}_*||^2 \\ &= ||\mathbf{x}_t - \mathbf{x}_*||^2 \, -2\gamma_t (\mathbf{h}_t)^T (\mathbf{x}_t - \mathbf{x}_*) + (\gamma_t)^2 ||\mathbf{h}_t||^2 \end{aligned}$$

By definition of the subgradient method, we have

$$f(\mathbf{x}_*) \geq f(\mathbf{x}_t) + (\mathbf{h}_t)^T (\mathbf{x}_* - \mathbf{x}_t)$$

$$-(\mathbf{h}_t)^T (\mathbf{x}_* - \mathbf{x}_t) \leq -\left(f(\mathbf{x}_t) - f(\mathbf{x}_*)\right)$$

Using this inequality for $t = 0, \ldots T$ we have

$$\begin{aligned} ||\mathbf{x}_{(T+1)} - \mathbf{x}_*||^2 &\leq ||\mathbf{x}_T - \mathbf{x}_*||^2 \, -2\gamma_T \left(f(\mathbf{x}_T) - f(\mathbf{x}^*)\right) + \gamma_T^2 ||\mathbf{h}_T||^2 \\ &\leq ||\mathbf{x}_{(1)} - \mathbf{x}_*||^2 - 2\sum_{t=1}^{T} \gamma_t (f(\mathbf{x}_t) - f(\mathbf{x}_*)) + \sum_{t=1}^{T} \gamma_t^2 ||\mathbf{h}_t||^2 \end{aligned}$$

Subgradient Descent: Convergence Analysis (contd.)

Proof:

$$\begin{aligned} ||\mathbf{x}_{(t+1)} - \mathbf{x}_*||^2 &= ||\mathbf{x}_t - \gamma_t \mathbf{h}_t - \mathbf{x}_*||^2 \\ &= ||\mathbf{x}_t - \mathbf{x}_*||^2 \boxed{-2\gamma_t (\mathbf{h}_t)^T (\mathbf{x}_t - \mathbf{x}_*)} + (\gamma_t)^2 ||\mathbf{h}_t||^2 \end{aligned}$$

By definition of the subgradient method, we have

$$f(\mathbf{x}_*) \geq f(\mathbf{x}_t) + (\mathbf{h}_t)^T (\mathbf{x}_* - \mathbf{x}_t)$$

$$\boxed{-(\mathbf{h}_t)^T (\mathbf{x}_* - \mathbf{x}_t)} \leq \boxed{-\left(f(\mathbf{x}_t) - f(\mathbf{x}_*)\right)}$$

Using this inequality for $t = 0, \ldots T$ we have By completing squares

$$\begin{aligned} 0 \leq ||\mathbf{x}_{(T+1)} - \mathbf{x}_*||^2 &\leq ||\mathbf{x}_T - \mathbf{x}_*||^2 \boxed{-2\gamma_T \left(f(\mathbf{x}_T) - f(\mathbf{x}^*)\right)} + \gamma_T^2 ||\mathbf{h}_T||^2 \\ &\leq ||\mathbf{x}_{(1)} - \mathbf{x}_*||^2 - 2\sum_{t=1}^{T} \gamma_t (f(\mathbf{x}_t) - f(\mathbf{x}_*)) + \sum_{t=1}^{T} \gamma_t^2 ||\mathbf{h}_t||^2 \end{aligned}$$

$\leq l^2$

Subgradient Descent: Convergence Analysis (contd.)

And since this is lower bounded by 0, we have

$$0 \leq ||\mathbf{x}_{(T+1)} - \mathbf{x}_*||^2 \leq R^2 - 2\sum_{t=1}^{T} \gamma_i \left[f(\mathbf{x}_t) - f(\mathbf{x}_*)\right] + \sum_{t=1}^{T} \gamma_t^2 l^2$$

$$\Rightarrow 2\sum_{t=1}^{T} \gamma_t \left[f(\mathbf{x}_t) - f(\mathbf{x}_*)\right] \leq R^2 + \sum_{t=1}^{T} \gamma_t^2 l^2$$

$$\Rightarrow 2\left(\sum_{t=1}^{T} \gamma_t\right)\left[f(\mathbf{x}_T^{best}) - f(\mathbf{x}_*)\right] \leq R^2 + \sum_{t=1}^{T} \gamma_t^2 l^2$$

Subgradient Descent: Convergence Analysis (contd.)

And since this is lower bounded by 0, we have

$$0 \leq ||\mathbf{x}_{(T+1)} - \mathbf{x}_*||^2 \leq R^2 - 2\sum_{t=1}^{T} \gamma_i \left[f(\mathbf{x}_t) - f(\mathbf{x}_*)\right] + \sum_{t=1}^{T} \gamma_t^2 l^2$$

$$\Rightarrow 2\sum_{t=1}^{T} \gamma_t \left[f(\mathbf{x}_t) - f(\mathbf{x}_*)\right] \leq R^2 + \sum_{t=1}^{T} \gamma_t^2 l^2$$

$$\Rightarrow 2\left(\sum_{t=1}^{T} \gamma_t\right)\left[f(\mathbf{x}_T^{best}) - f(\mathbf{x}_*)\right] \leq R^2 + \sum_{t=1}^{T} \gamma_t^2 l^2$$

Conditions on γ_t? $\gamma_t \to 0$ as $t \to \infty$

Const $\gamma_t \Rightarrow$ $f(\mathbf{x}_T^{best}) - f(x^*) \leq \frac{R^2}{2T\gamma} + \frac{\gamma l^2}{2}$

Subgradient Descent: Convergence Analysis (contd.)

For a constant step size $\gamma_t = \gamma$:

$$\frac{R^2 + l^2\gamma^2 T}{2\gamma T} \to \frac{l^2\gamma}{2}, \text{ as } T \to \infty$$

and for diminishing step size, we have:

Subgradient Descent: Convergence Analysis (contd.)

For a constant step size $\gamma_t = \gamma$:

$$\frac{R^2 + l^2\gamma^2 T}{2\gamma T} \to \frac{l^2\gamma}{2}, \text{ as } T \to \infty$$

and for diminishing step size, we have:

$r_t \to 0$ as $t \to \infty$

Is $\sum r_t^2 < \infty$ reqd.

$$f(x_T^{best}) - f(x^*) \leq \frac{1}{2(\sum r_t)} \left[R^2 + l^2 \sum r_t^2\right]$$

$\sum r_t \to \infty$

Subgradient Descent: Convergence Analysis (contd.)

For a constant step size $\gamma_t = \gamma$:

$$\frac{R^2 + l^2\gamma^2 T}{2\gamma T} \to \frac{l^2\gamma}{2}, \text{ as } T \to \infty$$

and for diminishing step size, we have:

$$\sum_{t=0}^{T} \gamma_t^2 < \infty, \sum_{i=0}^{T} \gamma_t = \infty$$

therefore,

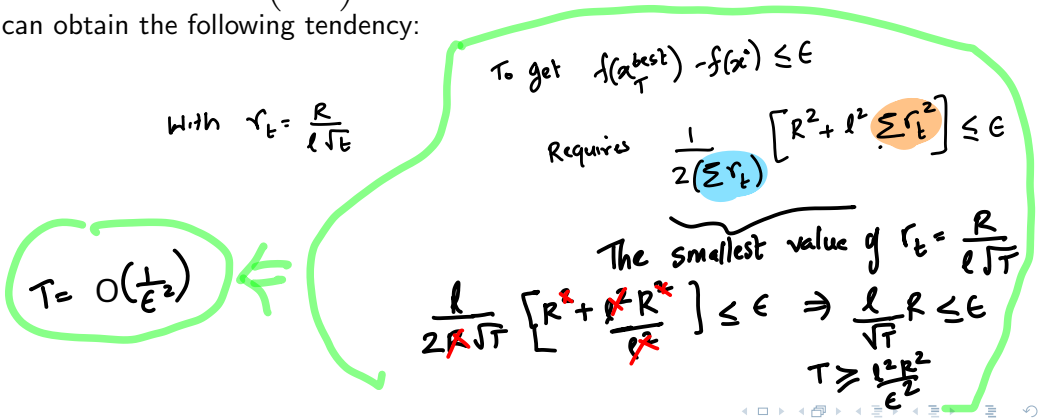
$$\frac{R^2 + l^2 \sum_{t=0}^{T} \gamma_t^2}{2\sum_{t=0}^{T} \gamma_t} \to 0, \text{ as } T \to \infty$$

Subgradient Descent: Convergence Analysis (contd.)

Consider taking $\gamma_t = R/\left(I\sqrt{T}\right)$ (or more generally, $\gamma_t = R/(I\sqrt{t})$), for all $t = 1, ..., T$. Then we can obtain the following tendency:

Subgradient Descent: Convergence Analysis (contd.)

Consider taking $\gamma_t = R/\left(l\sqrt{T}\right)$ (or more generally, $\gamma_t = R/(l\sqrt{t})$), for all $t = 1, ..., T$. Then we can obtain the following tendency:



Subgradient Descent: Convergence Analysis (contd.)

Consider taking $\gamma_t = R/\left(l\sqrt{T}\right)$ (or more generally, $\gamma_t = R/(l\sqrt{t})$), for all $t = 1, ..., T$. Then we can obtain the following tendency:

$$\frac{R^2 + l^2 \sum_{t=0}^{T} \gamma_t^2}{2\sum_{t=0}^{t} \gamma_t} = \frac{Rl}{\sqrt{T}}.$$

That is, subgradient method has convergence rate of $O(\frac{1}{\sqrt{T}})$, and to get $f(x_T^{best}) - f(x_*) \leq \epsilon$, needs $O(\frac{1}{\epsilon^2})$ iterations.

Recap: Summary after including subgradient descent...

- Lipschitz continuous functions (C): R^2B^2/ϵ^2 iterations
- Lipschitz continuous functions + Strongly Convex (CS): $2B^2/\epsilon - 1$ iterations
- Smooth Functions GD (SGD): $\frac{R^2L}{\epsilon}$ iterations.
- Smooth Functions AGD (SAGD): $\sqrt{\frac{2R^2L}{\epsilon}}$ iterations. Speeds up rate of convergence from $\frac{LR^2}{2T}$ to $\frac{2LR^2}{T^2}$ to match the lower bound of $\frac{3L}{32}\frac{R^2}{(T+1)^2}$
- Smooth + Strongly Convex (SS): $\frac{L}{\mu}\log(\frac{R^2L}{2\epsilon})$ iterations ($O\left(\log(\frac{1}{\epsilon})\right)$ even with heavy ball acceleration but with better factors).
- Subgradient method has convergence rate of $O(\frac{1}{\sqrt{T}})$, and to get $f(x_T^{best}) - f(x_*) \leq \epsilon$, needs $O(\frac{1}{\epsilon^2})$ iterations.

Recap: Summary after including subgradient descent...

- Lipschitz continuous functions (C): R^2B^2/ϵ^2 iterations
- Lipschitz continuous functions + Strongly Convex (CS): $2B^2/\epsilon - 1$ iterations
- Smooth Functions GD (SGD): $\frac{R^2L}{\epsilon}$ iterations.
- Smooth Functions AGD (SAGD): $\sqrt{\frac{2R^2L}{\epsilon}}$ iterations. Speeds up rate of convergence from $\frac{LR^2}{2T}$ to $\frac{2LR^2}{T^2}$ to match the lower bound of $\frac{3L}{32}\frac{R^2}{(T+1)^2}$
- Smooth + Strongly Convex (SS): $\frac{L}{\mu}\log(\frac{R^2L}{2\epsilon})$ iterations ($O\left(\log(\frac{1}{\epsilon})\right)$ even with heavy ball acceleration but with better factors).
- Subgradient method has convergence rate of $O(\frac{1}{\sqrt{T}})$, and to get $f(x_T^{best}) - f(x_*) \leq \epsilon$, needs $O(\frac{1}{\epsilon^2})$ iterations.

(1) Subgradient requires similar # of steps BUT with more assumptions on x_t !!

(2) Also subgradient (descent) analysis is in terms of x_T^{best}

Our running Colab Notebook:
CS769_sms_RunningNotebook_from_GradientDescent-WithUnsupervisedAndSubsetSelection.ipynb
https://colab.research.google.com/drive/1f1y8_ZkL_WS7Kfd7q-KB9Yt2VME1wV35#scrollTo=r50pPWrwLO-1

Loss Functions in Machine Learning

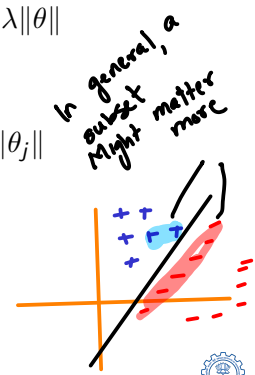
- L1/L2 Reg Logistic Regression: $L(\theta) = \sum_{i=1}^{n} \log(1 + \exp(-y_i \theta^T x_i)) + \lambda \|\theta\|$
- L1/L2 Reg SVMs: $L(\theta) = \sum_{i=1}^{n} \max\{0, 1 - y_i \theta^T x_i\} + \lambda \|\theta\|$
- L1/L2 Reg Multi-class Logistic Regression: $L(\theta_1, \cdots, \theta_k) = \sum_{i=1}^{n} -\theta_{y_i}^T x_i + \log(\sum_{c=1}^{k} \exp(\theta_c^T x_i))\} + \sum_{i=1}^{c} \lambda \sum_{j=1}^{m} \|\theta_j\|$
- L1/L2 Reg Least Squares (Lasso): $L(\theta) = \sum_{i=1}^{n} (\theta^T x_i - y_i)^2 + \lambda \|\theta\|$
- Matrix Completion: $L(X) = \sum_{i=1}^{n} ||y_i - A_i(X)||_2^2 + ||X||_*$
- Soft-Max Contextual Bandits: $L(\theta) = \sum_{i=1}^{n} \frac{r_i}{p_i} \frac{\exp(\theta^T x_i^{a_i})}{\sum_{j=1}^{k} \exp(\theta^T x_i^j)} + \lambda \|\theta\|$

Loss Functions in Machine Learning

While subgradient descent is somewhat specific to machine learning and so is lot of analysis based on L-smoothness and continuity, can we be even more specific to Machine Learning?

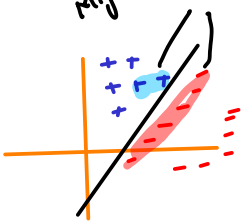
- L1/L2 Reg Logistic Regression: $L(\theta) = \sum_{i=1}^{n} \log(1 + \exp(-y_i \theta^T x_i)) + \lambda \|\theta\|$
- L1/L2 Reg SVMs: $L(\theta) = \sum_{i=1}^{n} \max\{0, 1 - y_i \theta^T x_i\} + \lambda \|\theta\|$
- L1/L2 Reg Multi-class Logistic Regression: $L(\theta_1, \cdots, \theta_k) = \sum_{i=1}^{n} -\theta_{y_i}^T x_i + \log(\sum_{c=1}^{k} \exp(\theta_c^T x_i))\} + \sum_{i=1}^{c} \lambda \sum_{j=1}^{m} \|\theta_j\|$
- L1/L2 Reg Least Squares (Lasso): $L(\theta) = \sum_{i=1}^{n} (\theta^T x_i - y_i)^2 + \lambda \|\theta\|$
- Matrix Completion: $L(X) = \sum_{i=1}^{n} ||y_i - A_i(X)||_2^2 + ||X||_*$
- Soft-Max Contextual Bandits: $L(\theta) = \sum_{i=1}^{n} \frac{r_i}{p_i} \frac{\exp(\theta^T x_i^{a_i})}{\sum_{j=1}^{k} \exp(\theta^T x_i^j)} + \lambda \|\theta\|$

In all these optimization objectives, we sum over all examples



1 and 3 are mixed discrete & continuous optimization formulations

In general, a subset might matter more



Question: Which subset of data points might matter more

1) Perhaps points close the boundary. But here is a chicken and egg problem.
To find the boundary, we need to learn the model. So who gives you the boundary
Such clustering/subset selection can also be computed on Kernel functions (see submodlib)
Possibly current (flaky or evolving) boundary?
Refer: General CORE SET based approaches

Gradmatch $\min_{S \subseteq X} \| \sum w_i \nabla L(x_i) - \nabla L_T \|$

2) Points with high gradient magnitude (say L2 norm) with the understanding that gradient norm is the uncertainty associated with that example since the function changes drastically at such points.
Refer: Curriculum based approaches

3) Points which are the shortest distance from the points of the opposite class or as in CRAIG in ICML 2020 which computes in K-mediod clusters in gradient space) so that remaining points are determined to be noisy data points

Refer: CRUST in Neurips 2019 - Such clustering is done in each class

Jacobian $J = \begin{bmatrix} \nabla f(x_1) \\ \nabla f(x_2) \\ \vdots \\ \nabla f(x_m) \end{bmatrix}$

Crust intends to decompose the Jacobian into a Noise and information space through eigenvector decomposition to get a Macro view

A simple and strong start: Since above approaches leverage data redundancy, why not pick a random subset of points?

Gradmatch $\min_{S \subseteq X} \| \sum w_i \nabla L(x_i) - \nabla L_T \|$

A simple and strong start:
Since above approaches leverage data redundancy,
why not pick a random subset of points?

Framework of Loss Functions in ML

- Machine Learning Loss functions are often a special class of continuous functions.
- They involve a sum of a large number of components: $f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$ where f_i is the cost function of the ith observation, n is the total number of training examples
- *Eg:*, $f_i(w) = L(x_i, y_i, w)$ where (x_i, y_i) is i^{th} training point for supervised learning, L is a Loss function
- Evaluating the full gradient ∇f of such an objective involves $O(n)$ computation
- In many applications, n is of the order of a million to a hundreds of millions of data points!

Notation: In what follows, we use the following new (and old for consistency with previous discussions)

- i as index into the example,
- w (so far x) to represent the function parameter
- t as index into the iteration

Framework of Loss Functions in ML

- Machine Learning Loss functions are often a special class of continuous functions.
- They involve a sum of a large number of components: $f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$ where f_i is the cost function of the ith observation, n is the total number of training examples
- *Eg:*, $f_i(w) = L(x_i, y_i, w)$ where (x_i, y_i) is i^{th} training point for supervised learning, L is a Loss function
- Evaluating the full gradient ∇f of such an objective involves $O(n)$ computation
- In many applications, n is of the order of a million to a hundreds of millions of data points!

Moreover it is redundant

Notation: In what follows, we use the following new (and old for consistency with previous discussions)

- i as index into the example,
- w (so far x) to represent the function parameter
- t as index into the iteration

Stochastic Gradient Descent Algorithm

For times $t = 0, 1, \ldots T$, and stepsizes $\gamma_t > 0$

- Choose $w_0 \in \Re^d$.
- sample $i \in [n]$ uniformly at random
- Compute the vector $g_t^i := \nabla f_i(w_t)$ - a stochastic gradient (a vector of d random variables). We will often drop the index i and call it g_t.
- $w_{t+1} := w_t - \gamma_t \nabla g_t^i = w_t - \gamma_t \nabla f_i(w_t)$ - Only update with the gradient of $f_i = f(x_i)$ instead of the full gradient!

Iteration is n times cheaper than full gradient descent!

Stochastic Gradient Descent Algorithm

RECAP:
A simple and strong start: Since previous approaches leverage data redundancy, why not pick a random subset of points?

For times $t = 0, 1, \ldots T$, and stepsizes $\gamma_t > 0$

- Choose $w_0 \in \Re^d$.
- sample $i \in [n]$ uniformly at random
- Compute the vector $g_t^i := \nabla f_i(w_t)$ - a stochastic gradient (a vector of d random variables). We will often drop the index i and call it g_t.
- $w_{t+1} := w_t - \gamma_t \nabla g_t^i = w_t - \gamma_t \nabla f_i(w_t)$ - Only update with the gradient of $f_i = f(x_i)$ instead of the full gradient!

Iteration is n times cheaper than full gradient descent!

Unbiasedness in SGD

Can we use convexity for vanilla analysis? What inequality holds in (1) consistently?

$$f(w_t) - f(w_*) \quad \text{----} \quad g_t^T (w_t - w_*) \tag{1}$$

Unbiasedness in SGD

because of choosing points "Uniformly at random" (with replacement is default in practice? without replacement is default in theory?)

Can we use convexity for vanilla analysis? What inequality holds in (1) consistently?

$$f(w_t) - f(w_*) \;\; ? \;\; g_t^T (w_t - w_*) \qquad (1)$$

Total loss — stochastic grad

May not hold since f is total loss but g_t is for i^{th} pt only

$$\sum_i (f_i(w_t) - f_i(w_*)) \;\; ? \;\; \sum (g_t)_i^T (w_t - w_*)$$

$$\sum_i (f_i(w_t) - f_i(w_*)) \leq \sum_i \bar{g}_t^T (w_t - w_*) \rightarrow \text{Holds for complete loss if } \bar{g}_t \text{ were complete gradient}$$

$$f(w_*) \geq f(w_t) + \bar{g}_t^T (w_* - w_t)$$

$$\bar{g}_t = \nabla \left(\sum_i f_i(w_t) \right)$$

$$f_i(w_t) - f_i(w^*) \leq (g_t)_i^T (w_t - w^*)$$

This holds

Loss on i^{th} pt

stochastic gradient