Optimization Review

CSci 8980: ML at Large Scale and High Dimensions

Arindam Banerjee

January 29, 2014

Convex Optimization: Smooth Functions

- Smooth convex function f on domain S

Convex Optimization: Smooth Functions

- Smooth convex function f on domain S
  - f has a minimizer x^* in S

Convex Optimization: Smooth Functions

- Smooth convex function f on domain S
  - f has a minimizer x^* in S
  - f is convex and continuously differentiable on S

Convex Optimization: Smooth Functions

- Smooth convex function f on domain S
  - f has a minimizer x^* in S
  - f is convex and continuously differentiable on S
  - f is smooth, i.e., gradient ∇f is β-Lipschitz: $\forall x, y \in S$

$$\|\nabla f(x) - \nabla f(y)\| \leq \beta \|x - y\|$$

Convex Optimization: Smooth Functions

- Smooth convex function f on domain S
  - f has a minimizer x^* in S
  - f is convex and continuously differentiable on S
  - f is smooth, i.e., gradient ∇f is β-Lipschitz: $\forall x, y \in S$

$$\|\nabla f(x) - \nabla f(y)\| \leq \beta \|x - y\|$$

- Gradient descent for smooth functions:

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

Convex Optimization: Smooth Functions

- Smooth convex function f on domain S
  - f has a minimizer x^* in S
  - f is convex and continuously differentiable on S
  - f is smooth, i.e., gradient ∇f is β-Lipschitz: $\forall x, y \in S$

$$\|\nabla f(x) - \nabla f(y)\| \leq \beta \|x - y\|$$

- Gradient descent for smooth functions:

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

- With $\eta = \frac{1}{\beta}$, we have

$$f(x_T) - f(x^*) \leq \frac{2\beta \|x_0 - x^*\|^2}{T+4}$$

Convex Optimization: Smooth Functions

- Smooth convex function f on domain S
  - f has a minimizer x^* in S
  - f is convex and continuously differentiable on S
  - f is smooth, i.e., gradient ∇f is β-Lipschitz: $\forall x, y \in S$

$$\|\nabla f(x) - \nabla f(y)\| \leq \beta \|x - y\|$$

- Gradient descent for smooth functions:

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

- With $\eta = \frac{1}{\beta}$, we have

$$f(x_T) - f(x^*) \leq \frac{2\beta \|x_0 - x^*\|^2}{T + 4}$$

- Rate can be $O(\frac{1}{T^2})$ using "accelerated" gradient descent

Convex Optimization: Non-Smooth Functions

- Often work with "non-smooth" functions

Convex Optimization: Non-Smooth Functions

- Often work with “non-smooth” functions
  - Hinge loss, L_1 norm, etc.

Convex Optimization: Non-Smooth Functions

- Often work with "non-smooth" functions
  - Hinge loss, L_1 norm, etc.
- Consider a non-smooth function f on domain S

Convex Optimization: Non-Smooth Functions

- Often work with "non-smooth" functions
  - Hinge loss, L_1 norm, etc.
- Consider a non-smooth function f on domain S
- Sub-differential set $\partial f(x)$: $g \in \partial f(x)$ if

$$f(y) \geq f(x) + \langle y - x, g \rangle \ , \quad \forall y \in S$$

Convex Optimization: Non-Smooth Functions

- Often work with "non-smooth" functions
  - Hinge loss, L_1 norm, etc.
- Consider a non-smooth function f on domain S
- Sub-differential set $\partial f(x)$: $g \in \partial f(x)$ if

$$f(y) \geq f(x) + \langle y - x, g \rangle \ , \quad \forall y \in S$$

- A non-smooth function is convex if $\partial f(x) \neq \emptyset, \forall x \in S$

Convex Optimization: Non-Smooth Functions

- Often work with "non-smooth" functions
  - Hinge loss, L_1 norm, etc.
- Consider a non-smooth function f on domain S
- Sub-differential set $\partial f(x)$: $g \in \partial f(x)$ if

$$f(y) \geq f(x) + \langle y - x, g \rangle \ , \quad \forall y \in S$$

- A non-smooth function is convex if $\partial f(x) \neq \emptyset, \forall x \in S$
  - Sub-differential set $\partial f(x)$ is convex, compact

Convex Optimization: Non-Smooth Functions

- Often work with "non-smooth" functions
  - Hinge loss, L_1 norm, etc.
- Consider a non-smooth function f on domain S
- Sub-differential set $\partial f(x)$: $g \in \partial f(x)$ if

$$f(y) \geq f(x) + \langle y - x, g \rangle \ , \quad \forall y \in S$$

- A non-smooth function is convex if $\partial f(x) \neq \emptyset, \forall x \in S$
  - Sub-differential set $\partial f(x)$ is convex, compact
  - Each $g \in \partial f(x)$ is a sub-gardient

Convex Optimization: Non-Smooth Functions

- Often work with "non-smooth" functions
  - Hinge loss, L_1 norm, etc.
- Consider a non-smooth function f on domain S
- Sub-differential set $\partial f(x)$: $g \in \partial f(x)$ if

$$f(y) \geq f(x) + \langle y - x, g \rangle \ , \quad \forall y \in S$$

- A non-smooth function is convex if $\partial f(x) \neq \emptyset, \forall x \in S$
  - Sub-differential set $\partial f(x)$ is convex, compact
  - Each $g \in \partial f(x)$ is a sub-gardient
- Lipschitz convex functions f on domain S:

Convex Optimization: Non-Smooth Functions

- Often work with "non-smooth" functions
  - Hinge loss, L_1 norm, etc.
- Consider a non-smooth function f on domain S
- Sub-differential set $\partial f(x)$: $g \in \partial f(x)$ if

$$f(y) \geq f(x) + \langle y - x, g \rangle \ , \quad \forall y \in S$$

- A non-smooth function is convex if $\partial f(x) \neq \emptyset, \forall x \in S$
  - Sub-differential set $\partial f(x)$ is convex, compact
  - Each $g \in \partial f(x)$ is a sub-gardient
- Lipschitz convex functions f on domain S:
  - f has a minimizer x^* in S

Convex Optimization: Non-Smooth Functions

- Often work with "non-smooth" functions
  - Hinge loss, L_1 norm, etc.
- Consider a non-smooth function f on domain S
- Sub-differential set $\partial f(x)$: $g \in \partial f(x)$ if

$$f(y) \geq f(x) + \langle y - x, g \rangle \ , \quad \forall y \in S$$

- A non-smooth function is convex if $\partial f(x) \neq \emptyset, \forall x \in S$
  - Sub-differential set $\partial f(x)$ is convex, compact
  - Each $g \in \partial f(x)$ is a sub-gardient
- Lipschitz convex functions f on domain S:
  - f has a minimizer x^* in S
  - f is convex on S

Convex Optimization: Non-Smooth Functions

- Often work with "non-smooth" functions
  - Hinge loss, L_1 norm, etc.
- Consider a non-smooth function f on domain S
- Sub-differential set $\partial f(x)$: $g \in \partial f(x)$ if

$$f(y) \geq f(x) + \langle y - x, g \rangle \ , \quad \forall y \in S$$

- A non-smooth function is convex if $\partial f(x) \neq \emptyset, \forall x \in S$
  - Sub-differential set $\partial f(x)$ is convex, compact
  - Each $g \in \partial f(x)$ is a sub-gardient
- Lipschitz convex functions f on domain S:
  - f has a minimizer x^* in S
  - f is convex on S
  - f is G-Lipschitz on S, i.e., for any $g \in \partial f(x)$, we have $\|g\| \leq G$

Non-Smooth functions: Subgradient Descent

- Assume $\|x^*\| \leq R$

Non-Smooth functions: Subgradient Descent

- Assume $\|x^*\| \leq R$
- Assume f is G-Lipschitz in the R-ball, i.e., $\|g\| \leq G$ for $g \in \partial f(x)$ for $\|x\| \leq R$

Non-Smooth functions: Subgradient Descent

- Assume $\|x^*\| \leq R$
- Assume f is G-Lipschitz in the R-ball, i.e., $\|g\| \leq G$ for $g \in \partial f(x)$ for $\|x\| \leq R$
- Projected sub-gradient descent

$$y_{t+1} = x_t - \eta g_t \ , \quad \text{where } g_t \in \partial f(x_t)$$

$$x_{t+1} = \begin{cases} y_{t+1} \ , & \text{if } \ \|y_{t+1}\| \leq R \\ \frac{R}{\|y_{t+1}\|} y_{t+1} \ , & \text{if } \ \|y_{t+1}\| > R \end{cases}$$

Non-Smooth functions: Subgradient Descent

- Assume $\|x^*\| \leq R$
- Assume f is G-Lipschitz in the R-ball, i.e., $\|g\| \leq G$ for $g \in \partial f(x)$ for $\|x\| \leq R$
- Projected sub-gradient descent

$$y_{t+1} = x_t - \eta g_t \ , \quad \text{where } g_t \in \partial f(x_t)$$

$$x_{t+1} = \begin{cases} y_{t+1} \ , & \text{if } \ \|y_{t+1}\| \leq R \\ \frac{R}{\|y_{t+1}\|} y_{t+1} \ , & \text{if } \ \|y_{t+1}\| > R \end{cases}$$

- With $\eta = \frac{R}{G\sqrt{T}}$, $\bar{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t$ satisfies

$$f(\bar{x}_T) - f(x^*) \leq \frac{RG}{\sqrt{T}}$$

Non-Smooth functions: Subgradient Descent

- Assume $\|x^*\| \leq R$
- Assume f is G-Lipschitz in the R-ball, i.e., $\|g\| \leq G$ for $g \in \partial f(x)$ for $\|x\| \leq R$
- Projected sub-gradient descent

$$y_{t+1} = x_t - \eta g_t \ , \quad \text{where } g_t \in \partial f(x_t)$$

$$x_{t+1} = \begin{cases} y_{t+1} \ , & \text{if } \ \|y_{t+1}\| \leq R \\ \frac{R}{\|y_{t+1}\|} y_{t+1} \ , & \text{if } \ \|y_{t+1}\| > R \end{cases}$$

- With $\eta = \frac{R}{G\sqrt{T}}$, $\bar{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t$ satisfies

$$f(\bar{x}_T) - f(x^*) \leq \frac{RG}{\sqrt{T}}$$

  - Step-size is more conservative, compared to smooth functions

Non-Smooth functions: Subgradient Descent

- Assume $\|x^*\| \leq R$
- Assume f is G-Lipschitz in the R-ball, i.e., $\|g\| \leq G$ for $g \in \partial f(x)$ for $\|x\| \leq R$
- Projected sub-gradient descent

$$y_{t+1} = x_t - \eta g_t \ , \quad \text{where } g_t \in \partial f(x_t)$$

$$x_{t+1} = \begin{cases} y_{t+1} \ , & \text{if } \ \|y_{t+1}\| \leq R \\ \frac{R}{\|y_{t+1}\|} y_{t+1} \ , & \text{if } \ \|y_{t+1}\| > R \end{cases}$$

- With $\eta = \frac{R}{G\sqrt{T}}$, $\bar{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t$ satisfies

$$f(\bar{x}_T) - f(x^*) \leq \frac{RG}{\sqrt{T}}$$

  - Step-size is more conservative, compared to smooth functions
  - Rate cannot be improved by “acceleration”

Non-Smooth functions: Subgradient Descent

- Assume $\|x^*\| \leq R$
- Assume f is G-Lipschitz in the R-ball, i.e., $\|g\| \leq G$ for $g \in \partial f(x)$ for $\|x\| \leq R$
- Projected sub-gradient descent

$$y_{t+1} = x_t - \eta g_t \ , \quad \text{where } g_t \in \partial f(x_t)$$

$$x_{t+1} = \begin{cases} y_{t+1} \ , & \text{if } \ \|y_{t+1}\| \leq R \\ \frac{R}{\|y_{t+1}\|} y_{t+1} \ , & \text{if } \ \|y_{t+1}\| > R \end{cases}$$

- With $\eta = \frac{R}{G\sqrt{T}}$, $\bar{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t$ satisfies

$$f(\bar{x}_T) - f(x^*) \leq \frac{RG}{\sqrt{T}}$$

  - Step-size is more conservative, compared to smooth functions
  - Rate cannot be improved by "acceleration"
  - Bound holds for $\tilde{x}_T = \text{argmin}_{1 \leq t \leq T} f(x_t)$

Iteration Complexity, Oracle Complexity

- Algorithm needs access to an oracle

Iteration Complexity, Oracle Complexity

- Algorithm needs access to an oracle
  - 0^{th} order: Given x, what is $f(x)$

Iteration Complexity, Oracle Complexity

- Algorithm needs access to an oracle
  - 0^{th} order: Given x, what is $f(x)$
  - 1^{st} order: Given x, what is $\nabla f(x)$ (or sub-gradient)

Iteration Complexity, Oracle Complexity

- Algorithm needs access to an oracle
  - 0^{th} order: Given x, what is $f(x)$
  - 1^{st} order: Given x, what is $\nabla f(x)$ (or sub-gradient)
- An algorithm with a 1^{st} order oracle is a mapping:

$$x_t = \phi_t(\{x_\tau, f(x_\tau), \nabla f(x_\tau)\}, \tau = 0, \ldots, t-1)$$

Iteration Complexity, Oracle Complexity

- Algorithm needs access to an oracle
  - 0^{th} order: Given x, what is $f(x)$
  - 1^{st} order: Given x, what is $\nabla f(x)$ (or sub-gradient)
- An algorithm with a 1^{st} order oracle is a mapping:

$$x_t = \phi_t(\{x_\tau, f(x_\tau), \nabla f(x_\tau)\}, \tau = 0, \ldots, t-1)$$

- Iteration complexity: T to get $f(x_T) - f(x^*) \leq \epsilon$

Iteration Complexity, Oracle Complexity

- Algorithm needs access to an oracle
  - 0^{th} order: Given x, what is $f(x)$
  - 1^{st} order: Given x, what is $\nabla f(x)$ (or sub-gradient)
- An algorithm with a 1^{st} order oracle is a mapping:

$$x_t = \phi_t(\{x_\tau, f(x_\tau), \nabla f(x_\tau)\}, \tau = 0, \ldots, t-1)$$

- Iteration complexity: T to get $f(x_T) - f(x^*) \leq \epsilon$
  - GD for smooth functions: $T = O(\frac{1}{\epsilon})$

Iteration Complexity, Oracle Complexity

- Algorithm needs access to an oracle
  - 0^{th} order: Given x, what is $f(x)$
  - 1^{st} order: Given x, what is $\nabla f(x)$ (or sub-gradient)
- An algorithm with a 1^{st} order oracle is a mapping:

$$x_t = \phi_t(\{x_\tau, f(x_\tau), \nabla f(x_\tau)\}, \tau = 0, \ldots, t-1)$$

- Iteration complexity: T to get $f(x_T) - f(x^*) \leq \epsilon$
  - GD for smooth functions: $T = O(\frac{1}{\epsilon})$
  - AGD for smooth functions: $T = O(\frac{1}{\sqrt{\epsilon}})$

Iteration Complexity, Oracle Complexity

- Algorithm needs access to an oracle
  - 0^{th} order: Given x, what is $f(x)$
  - 1^{st} order: Given x, what is $\nabla f(x)$ (or sub-gradient)
- An algorithm with a 1^{st} order oracle is a mapping:

$$x_t = \phi_t(\{x_\tau, f(x_\tau), \nabla f(x_\tau)\}, \tau = 0, \ldots, t-1)$$

- Iteration complexity: T to get $f(x_T) - f(x^*) \leq \epsilon$
  - GD for smooth functions: $T = O(\frac{1}{\epsilon})$
  - AGD for smooth functions: $T = O(\frac{1}{\sqrt{\epsilon}})$
  - 'GD' for non-smooth functions: $T = O(\frac{1}{\epsilon^2})$

Iteration Complexity, Oracle Complexity

- Algorithm needs access to an oracle
  - 0^{th} order: Given x, what is $f(x)$
  - 1^{st} order: Given x, what is $\nabla f(x)$ (or sub-gradient)
- An algorithm with a 1^{st} order oracle is a mapping:

$$x_t = \phi_t(\{x_\tau, f(x_\tau), \nabla f(x_\tau)\}, \tau = 0, \ldots, t-1)$$

- Iteration complexity: T to get $f(x_T) - f(x^*) \leq \epsilon$
  - GD for smooth functions: $T = O(\frac{1}{\epsilon})$
  - AGD for smooth functions: $T = O(\frac{1}{\sqrt{\epsilon}})$
  - 'GD' for non-smooth functions: $T = O(\frac{1}{\epsilon^2})$
- The minimax optimization error for function class $\mathcal{F}$

$$OC_t(\mathcal{F}) = \inf_{\phi_0,\ldots,\phi_t} \sup_{f \in \mathcal{F}} \left(f(x_t) - \inf_{x \in \mathcal{X}} f(x) \right)$$

Iteration Complexity, Oracle Complexity

- Algorithm needs access to an oracle
  - 0^{th} order: Given x, what is $f(x)$
  - 1^{st} order: Given x, what is $\nabla f(x)$ (or sub-gradient)
- An algorithm with a 1^{st} order oracle is a mapping:

$$x_t = \phi_t(\{x_\tau, f(x_\tau), \nabla f(x_\tau)\}, \tau = 0, \ldots, t-1)$$

- Iteration complexity: T to get $f(x_T) - f(x^*) \leq \epsilon$
  - GD for smooth functions: $T = O(\frac{1}{\epsilon})$
  - AGD for smooth functions: $T = O(\frac{1}{\sqrt{\epsilon}})$
  - 'GD' for non-smooth functions: $T = O(\frac{1}{\epsilon^2})$
- The minimax optimization error for function class $\mathcal{F}$

$$OC_t(\mathcal{F}) = \inf_{\phi_0,\ldots,\phi_t} \sup_{f \in \mathcal{F}} \left(f(x_t) - \inf_{x \in \mathcal{X}} f(x) \right)$$

- Oracle complexity: T so that $OC_T(\mathcal{F}) \leq \epsilon$

Iteration Complexity, Oracle Complexity

- Algorithm needs access to an oracle
  - 0^{th} order: Given x, what is $f(x)$
  - 1^{st} order: Given x, what is $\nabla f(x)$ (or sub-gradient)
- An algorithm with a 1^{st} order oracle is a mapping:

$$x_t = \phi_t(\{x_\tau, f(x_\tau), \nabla f(x_\tau)\}, \tau = 0, \ldots, t-1)$$

- Iteration complexity: T to get $f(x_T) - f(x^*) \leq \epsilon$
  - GD for smooth functions: $T = O(\frac{1}{\epsilon})$
  - AGD for smooth functions: $T = O(\frac{1}{\sqrt{\epsilon}})$
  - 'GD' for non-smooth functions: $T = O(\frac{1}{\epsilon^2})$
- The minimax optimization error for function class $\mathcal{F}$

$$OC_t(\mathcal{F}) = \inf_{\phi_0,\ldots,\phi_t} \sup_{f \in \mathcal{F}} \left(f(x_t) - \inf_{x \in \mathcal{X}} f(x) \right)$$

- Oracle complexity: T so that $OC_T(\mathcal{F}) \leq \epsilon$
  - Smooth functions: $T = O(\frac{1}{\sqrt{\epsilon}})$

Iteration Complexity, Oracle Complexity

- Algorithm needs access to an oracle
  - 0^{th} order: Given x, what is $f(x)$
  - 1^{st} order: Given x, what is $\nabla f(x)$ (or sub-gradient)
- An algorithm with a 1^{st} order oracle is a mapping:

$$x_t = \phi_t(\{x_\tau, f(x_\tau), \nabla f(x_\tau)\}, \tau = 0, \dots, t-1)$$

- Iteration complexity: T to get $f(x_T) - f(x^*) \leq \epsilon$
  - GD for smooth functions: $T = O(\frac{1}{\epsilon})$
  - AGD for smooth functions: $T = O(\frac{1}{\sqrt{\epsilon}})$
  - 'GD' for non-smooth functions: $T = O(\frac{1}{\epsilon^2})$
- The minimax optimization error for function class $\mathcal{F}$

$$OC_t(\mathcal{F}) = \inf_{\phi_0,\dots,\phi_t} \sup_{f \in \mathcal{F}} \left(f(x_t) - \inf_{x \in \mathcal{X}} f(x) \right)$$

- Oracle complexity: T so that $OC_T(\mathcal{F}) \leq \epsilon$
  - Smooth functions: $T = O(\frac{1}{\sqrt{\epsilon}})$
  - Non-smooth functions: $T = O(\frac{1}{\epsilon^2})$

Stochastic Gradient Descent (SGD)

- Can we do better than gradient descent?

Stochastic Gradient Descent (SGD)

- Can we do better than gradient descent?
- Gradient descent for smooth functions: $O(\frac{m}{\epsilon})$

Stochastic Gradient Descent (SGD)

- Can we do better than gradient descent?
- Gradient descent for smooth functions: $O(\frac{m}{\epsilon})$
  - Number of iterations $O(\frac{1}{\epsilon})$

Stochastic Gradient Descent (SGD)

- Can we do better than gradient descent?
- Gradient descent for smooth functions: $O(\frac{m}{\epsilon})$
  - Number of iterations $O(\frac{1}{\epsilon})$
  - Runtime in each iteration m

Stochastic Gradient Descent (SGD)

- Can we do better than gradient descent?
- Gradient descent for smooth functions: $O(\frac{m}{\epsilon})$
  - Number of iterations $O(\frac{1}{\epsilon})$
  - Runtime in each iteration m
- Sub-gradient descent for non-smooth functions: $O(\frac{m}{\epsilon^2})$

Stochastic Gradient Descent (SGD)

- Can we do better than gradient descent?
- Gradient descent for smooth functions: $O(\frac{m}{\epsilon})$
  - Number of iterations $O(\frac{1}{\epsilon})$
  - Runtime in each iteration m
- Sub-gradient descent for non-smooth functions: $O(\frac{m}{\epsilon^2})$
  - Number of iterations $O(\frac{1}{\epsilon^2})$

Stochastic Gradient Descent (SGD)

- Can we do better than gradient descent?
- Gradient descent for smooth functions: $O(\frac{m}{\epsilon})$
  - Number of iterations $O(\frac{1}{\epsilon})$
  - Runtime in each iteration m
- Sub-gradient descent for non-smooth functions: $O(\frac{m}{\epsilon^2})$
  - Number of iterations $O(\frac{1}{\epsilon^2})$
  - Runtime in each iteration m

Stochastic Gradient Descent (SGD)

- Can we do better than gradient descent?
- Gradient descent for smooth functions: $O(\frac{m}{\epsilon})$
  - Number of iterations $O(\frac{1}{\epsilon})$
  - Runtime in each iteration m
- Sub-gradient descent for non-smooth functions: $O(\frac{m}{\epsilon^2})$
  - Number of iterations $O(\frac{1}{\epsilon^2})$
  - Runtime in each iteration m
- Main idea:

Stochastic Gradient Descent (SGD)

- Can we do better than gradient descent?
- Gradient descent for smooth functions: $O(\frac{m}{\epsilon})$
  - Number of iterations $O(\frac{1}{\epsilon})$
  - Runtime in each iteration m
- Sub-gradient descent for non-smooth functions: $O(\frac{m}{\epsilon^2})$
  - Number of iterations $O(\frac{1}{\epsilon^2})$
  - Runtime in each iteration m
- Main idea:
  - Decease the runtime in each iteration

Stochastic Gradient Descent (SGD)

- Can we do better than gradient descent?
- Gradient descent for smooth functions: $O(\frac{m}{\epsilon})$
  - Number of iterations $O(\frac{1}{\epsilon})$
  - Runtime in each iteration m
- Sub-gradient descent for non-smooth functions: $O(\frac{m}{\epsilon^2})$
  - Number of iterations $O(\frac{1}{\epsilon^2})$
  - Runtime in each iteration m
- Main idea:
  - Decease the runtime in each iteration
  - Possibly increase the number of iterations

Stochastic Gradient Descent (SGD)

- Can we do better than gradient descent?
- Gradient descent for smooth functions: $O(\frac{m}{\epsilon})$
  - Number of iterations $O(\frac{1}{\epsilon})$
  - Runtime in each iteration m
- Sub-gradient descent for non-smooth functions: $O(\frac{m}{\epsilon^2})$
  - Number of iterations $O(\frac{1}{\epsilon^2})$
  - Runtime in each iteration m
- Main idea:
  - Decease the runtime in each iteration
  - Possibly increase the number of iterations
  - The decrease should be more than the increase

Stochastic Gradient Descent (SGD)

- Can we do better than gradient descent?
- Gradient descent for smooth functions: $O(\frac{m}{\epsilon})$
  - Number of iterations $O(\frac{1}{\epsilon})$
  - Runtime in each iteration m
- Sub-gradient descent for non-smooth functions: $O(\frac{m}{\epsilon^2})$
  - Number of iterations $O(\frac{1}{\epsilon^2})$
  - Runtime in each iteration m
- Main idea:
  - Decease the runtime in each iteration
  - Possibly increase the number of iterations
  - The decrease should be more than the increase
- Simplest case: $m = 1$, i.e., compute only 1 gradient

Stochastic Gradient Descent (SGD)

- Can we do better than gradient descent?
- Gradient descent for smooth functions: $O(\frac{m}{\epsilon})$
  - Number of iterations $O(\frac{1}{\epsilon})$
  - Runtime in each iteration m
- Sub-gradient descent for non-smooth functions: $O(\frac{m}{\epsilon^2})$
  - Number of iterations $O(\frac{1}{\epsilon^2})$
  - Runtime in each iteration m
- Main idea:
  - Decease the runtime in each iteration
  - Possibly increase the number of iterations
  - The decrease should be more than the increase
- Simplest case: $m = 1$, i.e., compute only 1 gradient
- Questions: What is the algorithm? Will this converge?

Stochastic Gradient Descent (SGD)

- Assume: Samples $(\mathbf{x}_i, y_i)$ are i.i.d., consider

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} \ell((\mathbf{x}_i, y_i), \mathbf{w})$$

Stochastic Gradient Descent (SGD)

- Assume: Samples $(\mathbf{x}_i, y_i)$ are i.i.d., consider

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} \ell((\mathbf{x}_i, y_i), \mathbf{w})$$

- Stochastic gradient descent:

Stochastic Gradient Descent (SGD)

- Assume: Samples $(\mathbf{x}_i, y_i)$ are i.i.d., consider

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} \ell((\mathbf{x}_i, y_i), \mathbf{w})$$

- Stochastic gradient descent:
  - For $t = 1, \ldots, T$

Stochastic Gradient Descent (SGD)

- Assume: Samples $(\mathbf{x}_i, y_i)$ are i.i.d., consider

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} \ell((\mathbf{x}_i, y_i), \mathbf{w})$$

- Stochastic gradient descent:
  - For $t = 1, \ldots, T$
    - Randomly draw $i \in \{1, \ldots, m\}$

Stochastic Gradient Descent (SGD)

- Assume: Samples $(\mathbf{x}_i, y_i)$ are i.i.d., consider

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} \ell((\mathbf{x}_i, y_i), \mathbf{w})$$

- Stochastic gradient descent:
  - For $t = 1, \dots, T$
    - Randomly draw $i \in \{1, \dots, m\}$
    - Compute (sub)gradient $g_t = \nabla \ell((\mathbf{x}_i, y_i), \mathbf{w}_t)$

Stochastic Gradient Descent (SGD)

- Assume: Samples $(\mathbf{x}_i, y_i)$ are i.i.d., consider

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} \ell((\mathbf{x}_i, y_i), \mathbf{w})$$

- Stochastic gradient descent:
  - For $t = 1, \ldots, T$
    - Randomly draw $i \in \{1, \ldots, m\}$
    - Compute (sub)gradient $g_t = \nabla \ell((\mathbf{x}_i, y_i), \mathbf{w}_t)$
    - $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t g_t$

Stochastic Gradient Descent (SGD)

- Assume: Samples $(\mathbf{x}_i, y_i)$ are i.i.d., consider

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} \ell((\mathbf{x}_i, y_i), \mathbf{w})$$

- Stochastic gradient descent:
  - For $t = 1, \ldots, T$
    - Randomly draw $i \in \{1, \ldots, m\}$
    - Compute (sub)gradient $g_t = \nabla \ell((\mathbf{x}_i, y_i), \mathbf{w}_t)$
    - $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t g_t$
  - Output $\bar{\mathbf{w}}_T = \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t$

Stochastic Gradient Descent (SGD)

- Assume: Samples $(\mathbf{x}_i, y_i)$ are i.i.d., consider

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} \ell((\mathbf{x}_i, y_i), \mathbf{w})$$

- Stochastic gradient descent:
  - For $t = 1, \ldots, T$
    - Randomly draw $i \in \{1, \ldots, m\}$
    - Compute (sub)gradient $g_t = \nabla \ell((\mathbf{x}_i, y_i), \mathbf{w}_t)$
    - $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t g_t$
  - Output $\bar{\mathbf{w}}_T = \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t$
- Choosing step-size: Assume $E[\|g\|^2] \leq G^2$

Stochastic Gradient Descent (SGD)

- Assume: Samples $(\mathbf{x}_i, y_i)$ are i.i.d., consider

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} \ell((\mathbf{x}_i, y_i), \mathbf{w})$$

- Stochastic gradient descent:
  - For $t = 1, \dots, T$
    - Randomly draw $i \in \{1, \dots, m\}$
    - Compute (sub)gradient $g_t = \nabla \ell((\mathbf{x}_i, y_i), \mathbf{w}_t)$
    - $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t g_t$
  - Output $\bar{\mathbf{w}}_T = \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t$
- Choosing step-size: Assume $E[\|g\|^2] \leq G^2$
  - Fixed: $\eta_t = \frac{\|\mathbf{w}^*\|_2}{G\sqrt{T}} \;\Rightarrow\; E[f(\bar{\mathbf{w}}_T)] - f(\mathbf{w}^*) \leq \frac{G\|\mathbf{w}^*\|_2}{\sqrt{T}}$

Stochastic Gradient Descent (SGD)

- Assume: Samples $(\mathbf{x}_i, y_i)$ are i.i.d., consider

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} \ell((\mathbf{x}_i, y_i), \mathbf{w})$$

- Stochastic gradient descent:
  - For $t = 1, \ldots, T$
    - Randomly draw $i \in \{1, \ldots, m\}$
    - Compute (sub)gradient $g_t = \nabla \ell((\mathbf{x}_i, y_i), \mathbf{w}_t)$
    - $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t g_t$
  - Output $\bar{\mathbf{w}}_T = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$
- Choosing step-size: Assume $E[\|g\|^2] \leq G^2$
  - Fixed: $\eta_t = \frac{\|\mathbf{w}^*\|_2}{G\sqrt{T}} \;\Rightarrow\; E[f(\bar{\mathbf{w}}_T)] - f(\mathbf{w}^*) \leq \frac{G\|\mathbf{w}^*\|_2}{\sqrt{T}}$
  - Decaying: $\eta_t = \frac{\|\mathbf{w}^*\|_2}{G\sqrt{t}} \;\Rightarrow\; E[f(\bar{\mathbf{w}}_T)] - f(\mathbf{w}^*) \leq \frac{4G\|\mathbf{w}^*\|_2}{\sqrt{T}}$

Stochastic Gradient Descent (SGD)

- Assume: Samples $(\mathbf{x}_i, y_i)$ are i.i.d., consider
$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} \ell((\mathbf{x}_i, y_i), \mathbf{w})$$
- Stochastic gradient descent:
  - For $t = 1, \ldots, T$
    - Randomly draw $i \in \{1, \ldots, m\}$
    - Compute (sub)gradient $g_t = \nabla \ell((\mathbf{x}_i, y_i), \mathbf{w}_t)$
    - $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t g_t$
  - Output $\bar{\mathbf{w}}_T = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$
- Choosing step-size: Assume $E[\|g\|^2] \leq G^2$
  - Fixed: $\eta_t = \frac{\|\mathbf{w}^*\|_2}{G\sqrt{T}} \Rightarrow E[f(\bar{\mathbf{w}}_T)] - f(\mathbf{w}^*) \leq \frac{G\|\mathbf{w}^*\|_2}{\sqrt{T}}$
  - Decaying: $\eta_t = \frac{\|\mathbf{w}^*\|_2}{G\sqrt{t}} \Rightarrow E[f(\bar{\mathbf{w}}_T)] - f(\mathbf{w}^*) \leq \frac{4G\|\mathbf{w}^*\|_2}{\sqrt{T}}$
  - Unknown $G, \|\mathbf{w}^*\|$:
    $\eta_t = \frac{\beta\|\mathbf{w}^*\|_2}{G\sqrt{t}} \Rightarrow E[f(\bar{\mathbf{w}}_T)] - f(\mathbf{w}^*) \leq \frac{4G\|\mathbf{w}^*\|_2}{\sqrt{T}} \max(\beta, \frac{1}{\beta})$

Smooth Functions: SGD vs GD

- SGD convergence rate:

$$\mathbb{E}[f(\bar{\mathbf{w}}_T)] - f(\mathbf{w}^*) \leq O\left(\frac{1}{\sqrt{T}}\right)$$

$$\text{Iteration complexity } T = O\left(\frac{1}{\epsilon^2}\right)$$

Smooth functions	GD	SGD
Number of iterations	$O(\frac{1}{\epsilon})$	$O(\frac{1}{\epsilon^2})$
Each iteration	m	1
Total runtime	$O(\frac{m}{\epsilon})$	$O(\frac{1}{\epsilon^2})$
$m = 10^6$, $\epsilon = 10^{-2}$	10^8	10^4

Smooth Functions: SGD vs GD

- SGD convergence rate:

$$\mathbb{E}[f(\bar{\mathbf{w}}_T)] - f(\mathbf{w}^*) \leq O\left(\frac{1}{\sqrt{T}}\right)$$

$$\text{Iteration complexity } T = O\left(\frac{1}{\epsilon^2}\right)$$

Smooth functions	GD	SGD
Number of iterations	$O(\frac{1}{\epsilon})$	$O(\frac{1}{\epsilon^2})$
Each iteration	m	1
Total runtime	$O(\frac{m}{\epsilon})$	$O(\frac{1}{\epsilon^2})$
$m = 10^6$, $\epsilon = 10^{-2}$	10^8	10^4

- GD vs SGD: full gradient vs random gradient

Smooth Functions: SGD vs GD

- SGD convergence rate:

$$\mathbb{E}[f(\bar{\mathbf{w}}_T)] - f(\mathbf{w}^*) \leq O\left(\frac{1}{\sqrt{T}}\right)$$

$$\text{Iteration complexity } T = O\left(\frac{1}{\epsilon^2}\right)$$

Smooth functions	GD	SGD
Number of iterations	$O(\frac{1}{\epsilon})$	$O(\frac{1}{\epsilon^2})$
Each iteration	m	1
Total runtime	$O(\frac{m}{\epsilon})$	$O(\frac{1}{\epsilon^2})$
$m = 10^6$, $\epsilon = 10^{-2}$	10^8	10^4

- GD vs SGD: full gradient vs random gradient
- SGD is memory efficient, extends to mini-batches

Non-smooth Functions: SGD vs GD

- SGD convergence rate:

$$\mathbb{E}[f(\bar{\mathbf{w}}_T)] - f(\mathbf{w}^*) \leq O\left(\frac{1}{\sqrt{T}}\right)$$

$$\text{Iteration complexity } T = O\left(\frac{1}{\epsilon^2}\right)$$

Non-smooth functions	GD	SGD
Number of iterations	$O(\frac{1}{\epsilon^2})$	$O(\frac{1}{\epsilon^2})$
Each iteration	m	1
Total runtime	$O(\frac{m}{\epsilon^2})$	$O(\frac{1}{\epsilon^2})$
$m = 10^6$, $\epsilon = 10^{-2}$	10^{10}	10^4

Non-smooth Functions: SGD vs GD

- SGD convergence rate:

$$\mathbb{E}[f(\bar{\mathbf{w}}_T)] - f(\mathbf{w}^*) \leq O\left(\frac{1}{\sqrt{T}}\right)$$

$$\text{Iteration complexity } T = O\left(\frac{1}{\epsilon^2}\right)$$

Non-smooth functions	GD	SGD
Number of iterations	$O(\frac{1}{\epsilon^2})$	$O(\frac{1}{\epsilon^2})$
Each iteration	m	1
Total runtime	$O(\frac{m}{\epsilon^2})$	$O(\frac{1}{\epsilon^2})$
$m = 10^6$, $\epsilon = 10^{-2}$	10^{10}	10^4

- GD is $O(m)$ slower than SGD

Non-smooth Functions: SGD vs GD

- SGD convergence rate:

$$\mathbb{E}[f(\bar{\mathbf{w}}_T)] - f(\mathbf{w}^*) \leq O\left(\frac{1}{\sqrt{T}}\right)$$

$$\text{Iteration complexity } T = O\left(\frac{1}{\epsilon^2}\right)$$

Non-smooth functions	GD	SGD
Number of iterations	$O(\frac{1}{\epsilon^2})$	$O(\frac{1}{\epsilon^2})$
Each iteration	m	1
Total runtime	$O(\frac{m}{\epsilon^2})$	$O(\frac{1}{\epsilon^2})$
$m = 10^6$, $\epsilon = 10^{-2}$	10^{10}	10^4

- GD is $O(m)$ slower than SGD
- Examples: Hinge loss (SVMs)

Online Convex Optimization (OCO)

- ‘Sequential’ optimization with convex losses

Online Convex Optimization (OCO)

- 'Sequential' optimization with convex losses
- For $t = 1, 2, \ldots, T$

Online Convex Optimization (OCO)

- 'Sequential' optimization with convex losses
- For $t = 1, 2, \ldots, T$
  - Learner picks a vector $\mathbf{w}_t \in S$

Online Convex Optimization (OCO)

- 'Sequential' optimization with convex losses
- For $t = 1, 2, \ldots, T$
  - Learner picks a vector $\mathbf{w}_t \in S$
  - Receive convex function $f_t : S \mapsto \mathbb{R}$

Online Convex Optimization (OCO)

- 'Sequential' optimization with convex losses
- For $t = 1, 2, \ldots, T$
  - Learner picks a vector $\mathbf{w}_t \in S$
  - Receive convex function $f_t : S \mapsto \mathbb{R}$
  - Incur loss $f_t(\mathbf{w}_t)$

Online Convex Optimization (OCO)

- 'Sequential' optimization with convex losses
- For $t = 1, 2, \ldots, T$
  - Learner picks a vector $\mathbf{w}_t \in S$
  - Receive convex function $f_t : S \mapsto \mathbb{R}$
  - Incur loss $f_t(\mathbf{w}_t)$
- Regret w.r.t. comparator class $\mathcal{U}$

$$Regret_T(\mathcal{U}) = \sum_{t=1}^{T} f_t(\mathbf{w}_t) - \min_{\mathbf{u}\in\mathcal{U}} \sum_{t=1}^{T} f_t(\mathbf{u})$$

Online Gradient Descent (OGD)

- Apply gradient descent for OCO

Online Gradient Descent (OGD)

- Apply gradient descent for OCO
- FoReL minimizes $\sum_{\tau=1}^{t-1} f_\tau(\mathbf{w})$

Online Gradient Descent (OGD)

- Apply gradient descent for OCO
- FoReL minimizes $\sum_{\tau=1}^{t-1} f_\tau(\mathbf{w})$
  - May be difficult for more complicated f_τ's

Online Gradient Descent (OGD)

- Apply gradient descent for OCO
- FoReL minimizes $\sum_{\tau=1}^{t-1} f_\tau(\mathbf{w})$
  - May be difficult for more complicated f_τ's
  - Need to maintain all functions

Online Gradient Descent (OGD)

- Apply gradient descent for OCO
- FoReL minimizes $\sum_{\tau=1}^{t-1} f_\tau(\mathbf{w})$
  - May be difficult for more complicated f_τ's
  - Need to maintain all functions
- Use simple (sub)gradient descent

Online Gradient Descent (OGD)

- Apply gradient descent for OCO
- FoReL minimizes $\sum_{\tau=1}^{t-1} f_\tau(\mathbf{w})$
  - May be difficult for more complicated f_τ's
  - Need to maintain all functions
- Use simple (sub)gradient descent

Online Gradient Descent (OGD)

- Apply gradient descent for OCO
- FoReL minimizes $\sum_{\tau=1}^{t-1} f_\tau(\mathbf{w})$
  - May be difficult for more complicated f_τ's
  - Need to maintain all functions
- Use simple (sub)gradient descent

Algorithm Online Gradient Descent (OGD)

Set: $\eta > 0$
Initialize $\mathbf{w}_1 = 0$
for $t = 1, 2, 3, \ldots$ **do**
  $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla f_t(\mathbf{w}_t)$
end for

OGD: Regret Bounds

- Let f_t be general convex functions

OGD: Regret Bounds

- Let f_t be general convex functions
  - Sequence of vectors produced by OGD: $\mathbf{w}_1, \mathbf{w}_2, \ldots$

OGD: Regret Bounds

- Let f_t be general convex functions
  - Sequence of vectors produced by OGD: $\mathbf{w}_1, \mathbf{w}_2, \ldots$
  - Then

$$Regret_T = \sum_{t=1}^{T} f_t(\mathbf{w}_t) - \min_{\mathbf{u} \in S} \sum_{t=1}^{T} f_t(\mathbf{u}) = O(\sqrt{T})$$

OGD: Regret Bounds

- Let f_t be general convex functions
  - Sequence of vectors produced by OGD: $\mathbf{w}_1, \mathbf{w}_2, \ldots$
  - Then

$$Regret_T = \sum_{t=1}^{T} f_t(\mathbf{w}_t) - \min_{\mathbf{u} \in S} \sum_{t=1}^{T} f_t(\mathbf{u}) = O(\sqrt{T})$$

  - Example: Hinge loss

OGD: Regret Bounds

- Let f_t be general convex functions
  - Sequence of vectors produced by OGD: $\mathbf{w}_1, \mathbf{w}_2, \ldots$
  - Then

$$Regret_T = \sum_{t=1}^{T} f_t(\mathbf{w}_t) - \min_{\mathbf{u} \in S} \sum_{t=1}^{T} f_t(\mathbf{u}) = O(\sqrt{T})$$

  - Example: Hinge loss
- Let f_t be strongly convex functions

OGD: Regret Bounds

- Let f_t be general convex functions
  - Sequence of vectors produced by OGD: $\mathbf{w}_1, \mathbf{w}_2, \ldots$
  - Then

$$Regret_T = \sum_{t=1}^{T} f_t(\mathbf{w}_t) - \min_{\mathbf{u} \in S} \sum_{t=1}^{T} f_t(\mathbf{u}) = O(\sqrt{T})$$

  - Example: Hinge loss
- Let f_t be strongly convex functions
  - Sequence of vectors produced by OGD: $\mathbf{w}_1, \mathbf{w}_2, \ldots$

OGD: Regret Bounds

- Let f_t be general convex functions
  - Sequence of vectors produced by OGD: $\mathbf{w}_1, \mathbf{w}_2, \ldots$
  - Then

$$Regret_T = \sum_{t=1}^{T} f_t(\mathbf{w}_t) - \min_{\mathbf{u} \in S} \sum_{t=1}^{T} f_t(\mathbf{u}) = O(\sqrt{T})$$

  - Example: Hinge loss
- Let f_t be strongly convex functions
  - Sequence of vectors produced by OGD: $\mathbf{w}_1, \mathbf{w}_2, \ldots$
  - Then

$$Regret_T = \sum_{t=1}^{T} f_t(\mathbf{w}_t) - \min_{\mathbf{u} \in S} \sum_{t=1}^{T} f_t(\mathbf{u}) = O(\log(T))$$

OGD: Regret Bounds

- Let f_t be general convex functions
  - Sequence of vectors produced by OGD: $\mathbf{w}_1, \mathbf{w}_2, \ldots$
  - Then

$$Regret_T = \sum_{t=1}^{T} f_t(\mathbf{w}_t) - \min_{\mathbf{u} \in S} \sum_{t=1}^{T} f_t(\mathbf{u}) = O(\sqrt{T})$$

  - Example: Hinge loss
- Let f_t be strongly convex functions
  - Sequence of vectors produced by OGD: $\mathbf{w}_1, \mathbf{w}_2, \ldots$
  - Then

$$Regret_T = \sum_{t=1}^{T} f_t(\mathbf{w}_t) - \min_{\mathbf{u} \in S} \sum_{t=1}^{T} f_t(\mathbf{u}) = O(\log(T))$$

  - Example: Least square regression

Gradient Descent, Mirror Descent

- Gradient Descent: $\min_{\mathbf{w}} f(\mathbf{w})$

$$\mathbf{w}_{k+1} = \operatorname{argmin}_{\mathbf{w}} f(\mathbf{w}_k) + \langle \nabla f(\mathbf{w}_k), \mathbf{w} - \mathbf{w}_k \rangle + \frac{1}{2\alpha_k} \|\mathbf{w} - \mathbf{w}_k\|_2^2$$

$$= \operatorname{argmin}_{\mathbf{w}} \langle \nabla f(\mathbf{w}_k), \mathbf{w} \rangle + \frac{1}{2\alpha_k} \|\mathbf{w} - \mathbf{w}_k\|_2^2$$

Gradient Descent, Mirror Descent

- Gradient Descent: $\min_{\mathbf{w}} f(\mathbf{w})$

$$\mathbf{w}_{k+1} = \text{argmin}_{\mathbf{w}} f(\mathbf{w}_k) + \langle \nabla f(\mathbf{w}_k), \mathbf{w} - \mathbf{w}_k \rangle + \frac{1}{2\alpha_k} \|\mathbf{w} - \mathbf{w}_k\|_2^2$$

$$= \text{argmin}_{\mathbf{w}} \langle \nabla f(\mathbf{w}_k), \mathbf{w} \rangle + \frac{1}{2\alpha_k} \|\mathbf{w} - \mathbf{w}_k\|_2^2$$

- Setting the derivative to zero yields $\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \nabla f(\mathbf{w}_k)$

Gradient Descent, Mirror Descent

- Gradient Descent: $\min_{\mathbf{w}} f(\mathbf{w})$

$$\mathbf{w}_{k+1} = \text{argmin}_{\mathbf{w}} f(\mathbf{w}_k) + \langle \nabla f(\mathbf{w}_k), \mathbf{w} - \mathbf{w}_k \rangle + \frac{1}{2\alpha_k} \|\mathbf{w} - \mathbf{w}_k\|_2^2$$

$$= \text{argmin}_{\mathbf{w}} \langle \nabla f(\mathbf{w}_k), \mathbf{w} \rangle + \frac{1}{2\alpha_k} \|\mathbf{w} - \mathbf{w}_k\|_2^2$$

- Setting the derivative to zero yields $\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \nabla f(\mathbf{w}_k)$
- Replace the quadratic term by other functions?

Gradient Descent, Mirror Descent

- Gradient Descent: $\min_{\mathbf{w}} f(\mathbf{w})$

$$\mathbf{w}_{k+1} = \text{argmin}_{\mathbf{w}} f(\mathbf{w}_k) + \langle \nabla f(\mathbf{w}_k), \mathbf{w} - \mathbf{w}_k \rangle + \frac{1}{2\alpha_k} \|\mathbf{w} - \mathbf{w}_k\|_2^2$$

$$= \text{argmin}_{\mathbf{w}} \langle \nabla f(\mathbf{w}_k), \mathbf{w} \rangle + \frac{1}{2\alpha_k} \|\mathbf{w} - \mathbf{w}_k\|_2^2$$

- Setting the derivative to zero yields $\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \nabla f(\mathbf{w}_k)$
- Replace the quadratic term by other functions?
- Mirror descent

$$\mathbf{w}_{k+1} = \text{argmin}_{\mathbf{w}} \langle \nabla f(\mathbf{w}_k), \mathbf{w} \rangle + \frac{1}{\alpha_k} B_\phi(\mathbf{w}, \mathbf{w}_k)$$

Gradient Descent, Mirror Descent

- Gradient Descent: $\min_{\mathbf{w}} f(\mathbf{w})$

$$\mathbf{w}_{k+1} = \operatorname{argmin}_{\mathbf{w}} f(\mathbf{w}_k) + \langle \nabla f(\mathbf{w}_k), \mathbf{w} - \mathbf{w}_k \rangle + \frac{1}{2\alpha_k} \|\mathbf{w} - \mathbf{w}_k\|_2^2$$

$$= \operatorname{argmin}_{\mathbf{w}} \langle \nabla f(\mathbf{w}_k), \mathbf{w} \rangle + \frac{1}{2\alpha_k} \|\mathbf{w} - \mathbf{w}_k\|_2^2$$

- Setting the derivative to zero yields $\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \nabla f(\mathbf{w}_k)$
- Replace the quadratic term by other functions?
- Mirror descent

$$\mathbf{w}_{k+1} = \operatorname{argmin}_{\mathbf{w}} \langle \nabla f(\mathbf{w}_k), \mathbf{w} \rangle + \frac{1}{\alpha_k} B_\phi(\mathbf{w}, \mathbf{w}_k)$$

  - The convergence rate is the same as GD (SGD)

Gradient Descent, Mirror Descent

- Gradient Descent: $\min_{\mathbf{w}} f(\mathbf{w})$

$$\mathbf{w}_{k+1} = \text{argmin}_{\mathbf{w}} f(\mathbf{w}_k) + \langle \nabla f(\mathbf{w}_k), \mathbf{w} - \mathbf{w}_k \rangle + \frac{1}{2\alpha_k} \|\mathbf{w} - \mathbf{w}_k\|_2^2$$

$$= \text{argmin}_{\mathbf{w}} \langle \nabla f(\mathbf{w}_k), \mathbf{w} \rangle + \frac{1}{2\alpha_k} \|\mathbf{w} - \mathbf{w}_k\|_2^2$$

- Setting the derivative to zero yields $\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \nabla f(\mathbf{w}_k)$
- Replace the quadratic term by other functions?
- Mirror descent

$$\mathbf{w}_{k+1} = \text{argmin}_{\mathbf{w}} \langle \nabla f(\mathbf{w}_k), \mathbf{w} \rangle + \frac{1}{\alpha_k} B_\phi(\mathbf{w}, \mathbf{w}_k)$$

  - The convergence rate is the same as GD (SGD)
- Let distance-generating function ϕ be differentiable, strictly convex function, Bregman divergence is defined as

$$B_\phi(\mathbf{w}, \mathbf{w}_k) = \phi(\mathbf{w}) - \phi(\mathbf{w}_k) - \langle \nabla \phi(\mathbf{w}_k), \mathbf{w} - \mathbf{w}_k \rangle$$

Bregman Divergence

- $B_\phi(\mathbf{w}, \mathbf{w}_k) = \phi(\mathbf{w}) - \phi(\mathbf{w}_k) - \langle \nabla\phi(\mathbf{w}_k), \mathbf{w} - \mathbf{w}_k \rangle$

Bregman Divergence

- $B_\phi(\mathbf{w}, \mathbf{w}_k) = \phi(\mathbf{w}) - \phi(\mathbf{w}_k) - \langle \nabla\phi(\mathbf{w}_k), \mathbf{w} - \mathbf{w}_k \rangle$
- Examples: Squared loss, relative entropy, etc.

Bregman Divergence

- $B_\phi(\mathbf{w}, \mathbf{w}_k) = \phi(\mathbf{w}) - \phi(\mathbf{w}_k) - \langle \nabla\phi(\mathbf{w}_k), \mathbf{w} - \mathbf{w}_k \rangle$
- Examples: Squared loss, relative entropy, etc.
- Quadratic: $\phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$

Bregman Divergence

- $B_\phi(\mathbf{w}, \mathbf{w}_k) = \phi(\mathbf{w}) - \phi(\mathbf{w}_k) - \langle \nabla\phi(\mathbf{w}_k), \mathbf{w} - \mathbf{w}_k \rangle$
- Examples: Squared loss, relative entropy, etc.
- Quadratic: $\phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$
- Squared loss:

$$\begin{aligned} B_\phi(\mathbf{w}, \mathbf{w}_k) &= \frac{1}{2}\|\mathbf{w}\|_2^2 - \frac{1}{2}\|\mathbf{w}_k\|_2^2 - \langle \mathbf{w}_k, \mathbf{w} - \mathbf{w}_k \rangle \\ &= \frac{1}{2}\|\mathbf{w} - \mathbf{w}_k\|_2^2 \end{aligned}$$

Bregman Divergence

- $B_\phi(\mathbf{w}, \mathbf{w}_k) = \phi(\mathbf{w}) - \phi(\mathbf{w}_k) - \langle \nabla\phi(\mathbf{w}_k), \mathbf{w} - \mathbf{w}_k \rangle$
- Examples: Squared loss, relative entropy, etc.
- Quadratic: $\phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$
- Squared loss:

$$B_\phi(\mathbf{w}, \mathbf{w}_k) = \frac{1}{2}\|\mathbf{w}\|_2^2 - \frac{1}{2}\|\mathbf{w}_k\|_2^2 - \langle \mathbf{w}_k, \mathbf{w} - \mathbf{w}_k \rangle$$
$$= \frac{1}{2}\|\mathbf{w} - \mathbf{w}_k\|_2^2$$

- Entropy: $\phi(\mathbf{w}) = \sum_{i=1}^d \mathbf{w}(i) \log(\mathbf{w}(i))$, $\mathbf{w}(i)$ is the i-th entry

Bregman Divergence

- $B_\phi(\mathbf{w}, \mathbf{w}_k) = \phi(\mathbf{w}) - \phi(\mathbf{w}_k) - \langle \nabla\phi(\mathbf{w}_k), \mathbf{w} - \mathbf{w}_k \rangle$
- Examples: Squared loss, relative entropy, etc.
- Quadratic: $\phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$
- Squared loss:

$$B_\phi(\mathbf{w}, \mathbf{w}_k) = \frac{1}{2}\|\mathbf{w}\|_2^2 - \frac{1}{2}\|\mathbf{w}_k\|_2^2 - \langle \mathbf{w}_k, \mathbf{w} - \mathbf{w}_k \rangle$$
$$= \frac{1}{2}\|\mathbf{w} - \mathbf{w}_k\|_2^2$$

- Entropy: $\phi(\mathbf{w}) = \sum_{i=1}^{d} \mathbf{w}(i)\log(\mathbf{w}(i))$, $\mathbf{w}(i)$ is the i-th entry
- Relative entropy (un-normalized)

$$B_\phi(\mathbf{w}, \mathbf{w}_k) = \sum_{i=1}^{n} \left\{ \mathbf{w}_i \log\left(\frac{\mathbf{w}(i)}{\mathbf{w}_k(i)}\right) - \mathbf{w}(i) + \mathbf{w}_k(i) \right\}$$

Mirror Descent

- Mirror descent update:

$$\mathbf{w}_{k+1} = \text{argmin}_{\mathbf{w}} \langle \nabla f(\mathbf{w}_k), \mathbf{w} \rangle + \frac{1}{\alpha_k} (\phi(\mathbf{w}) - \phi(\mathbf{w}_k) - \langle \nabla \phi(\mathbf{w}_k), \mathbf{w} - \mathbf{w}_k \rangle$$

Mirror Descent

- Mirror descent update:

$$\mathbf{w}_{k+1} = \text{argmin}_{\mathbf{w}} \langle \nabla f(\mathbf{w}_k), \mathbf{w} \rangle + \frac{1}{\alpha_k} (\phi(\mathbf{w}) - \phi(\mathbf{w}_k) - \langle \nabla \phi(\mathbf{w}_k), \mathbf{w} - \mathbf{w}_k \rangle$$

- Setting the derivative to zero yields

$$\nabla f(\mathbf{w}_k) + \frac{1}{\alpha_k} (\nabla \phi(\mathbf{w}_{k+1}) - \nabla \phi(\mathbf{w}_k)) = 0$$

$$\Rightarrow \quad \mathbf{w}_{k+1} = \nabla \phi^{-1} (\nabla \phi(\mathbf{w}_k) - \alpha_k \nabla f(\mathbf{w}_k))$$

Mirror Descent

- Mirror descent update:

$$\mathbf{w}_{k+1} = \text{argmin}_{\mathbf{w}} \langle \nabla f(\mathbf{w}_k), \mathbf{w} \rangle + \frac{1}{\alpha_k}(\phi(\mathbf{w}) - \phi(\mathbf{w}_k) - \langle \nabla \phi(\mathbf{w}_k), \mathbf{w} - \mathbf{w}_k \rangle$$

- Setting the derivative to zero yields

$$\nabla f(\mathbf{w}_k) + \frac{1}{\alpha_k}(\nabla \phi(\mathbf{w}_{k+1}) - \nabla \phi(\mathbf{w}_k)) = 0$$

$$\Rightarrow \quad \mathbf{w}_{k+1} = \nabla \phi^{-1}(\nabla \phi(\mathbf{w}_k) - \alpha_k \nabla f(\mathbf{w}_k))$$

Mirror Descent

- Mirror descent update:

$$\mathbf{w}_{k+1} = \text{argmin}_{\mathbf{w}} \langle \nabla f(\mathbf{w}_k), \mathbf{w} \rangle + \frac{1}{\alpha_k} (\phi(\mathbf{w}) - \phi(\mathbf{w}_k) - \langle \nabla \phi(\mathbf{w}_k), \mathbf{w} - \mathbf{w}_k \rangle$$

- Setting the derivative to zero yields

$$\nabla f(\mathbf{w}_k) + \frac{1}{\alpha_k}(\nabla \phi(\mathbf{w}_{k+1}) - \nabla \phi(\mathbf{w}_k)) = 0$$

$$\Rightarrow \quad \mathbf{w}_{k+1} = \nabla \phi^{-1}(\nabla \phi(\mathbf{w}_k) - \alpha_k \nabla f(\mathbf{w}_k))$$