CS260: Machine Learning Algorithms

Lecture 3: Optimization

Cho-Jui Hsieh
UCLA

Jan 14, 2019

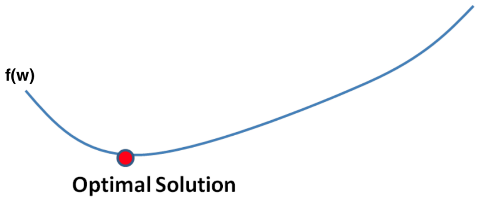
Optimization

- Goal: find the minimizer of a function

$$\min_{\boldsymbol{w}} f(\boldsymbol{w})$$

  For now we assume f is twice differentiable

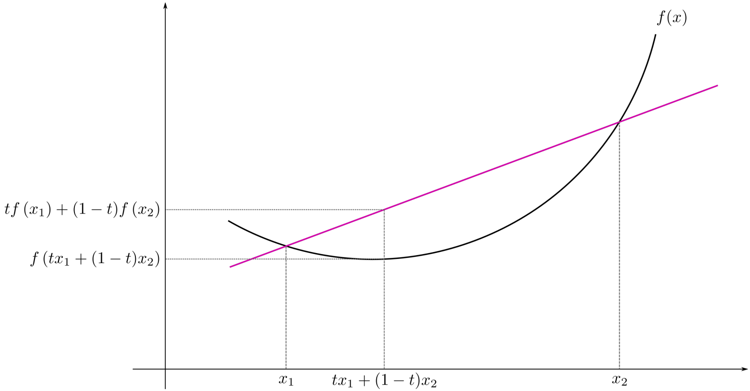
- Machine learning algorithm: find the hypothesis that **minimizes training error**



Convex function

- A function $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function
  $\Leftrightarrow$ the function f is below any line segment between two points on f:

$$\forall \boldsymbol{x}_1, \boldsymbol{x}_2,\ \forall t \in [0,1],\quad f(t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_2) \leq tf(\boldsymbol{x}_1) + (1-t)f(\boldsymbol{x}_2)$$

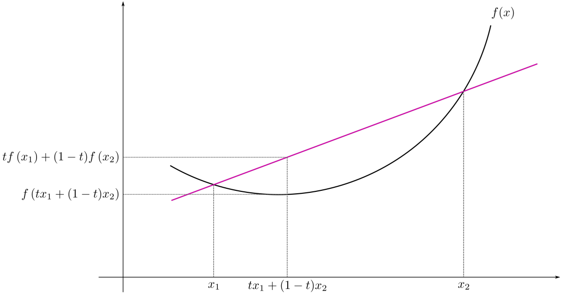


Convex function

- A function $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function

  $\Leftrightarrow$ the function f is below any line segment between two points on f:

$$\forall \boldsymbol{x}_1, \boldsymbol{x}_2, \ \forall t \in [0,1], \quad f(t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_2) \leq tf(\boldsymbol{x}_1) + (1-t)f(\boldsymbol{x}_2)$$

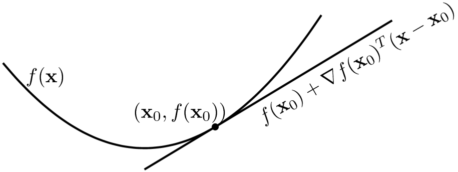


Strict convex: $f(t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_2) < tf(\boldsymbol{x}_1) + (1-t)f(\boldsymbol{x}_2)$

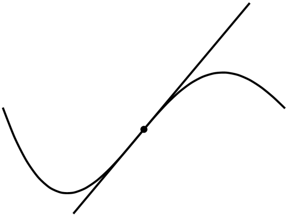
Convex function

- Another equivalent definition for differentiable function:

$$f \text{ is convex if and only if } f(\boldsymbol{x}) \geq f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)^T(\boldsymbol{x} - \boldsymbol{x}_0), \quad \forall \boldsymbol{x}, \boldsymbol{x}_0$$



convex function

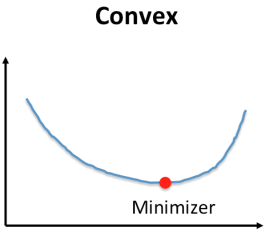


nonconvex function

Convex function

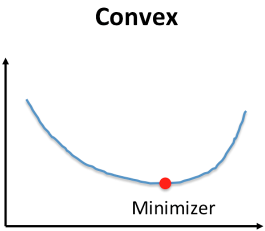
- Convex function:
  - (for differentiable function) $\nabla f(\boldsymbol{w}^*) = 0 \Leftrightarrow \boldsymbol{w}^*$ is a global minimum
  - If f is twice differentiable $\Rightarrow$

    f is convex if and only if $\nabla^2 f(\boldsymbol{w})$ **is positive semi-definite**
  - Example: linear regression, logistic regression, $\cdots$



Convex function

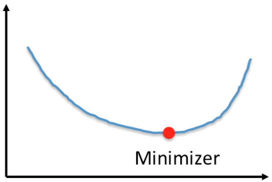
- Strict convex function:
  - $\nabla f(\boldsymbol{w}^*) = 0 \Leftrightarrow \boldsymbol{w}^*$ is the unique global minimum
    most algorithms only converge to gradient$= 0$
  - Example: Linear regression when $X^T X$ is invertible



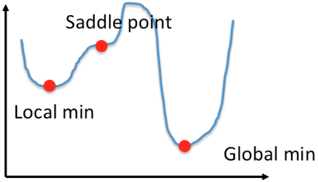
Convex vs Nonconvex

- Convex function:
  - $\nabla f(\boldsymbol{w}^*) = 0 \Leftrightarrow \boldsymbol{w}^*$ is a global minimum
  - Example: linear regression, logistic regression, $\cdots$
- Non-convex function:
  - $\nabla f(\boldsymbol{w}^*) = 0 \Leftrightarrow \boldsymbol{w}^*$ is Global min, local min, or saddle point
    (also called **stationary points**)
    most algorithms only converge to **stationary points**
  - Example: neural network, $\cdots$

Convex



Non-Convex



Gradient descent

Gradient Descent

- Gradient descent: repeatedly do

$$\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t - \alpha \nabla f(\boldsymbol{w}^t)$$

  $\alpha > 0$ is the step size

Gradient Descent

- Gradient descent: repeatedly do

$$\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t - \alpha \nabla f(\boldsymbol{w}^t)$$

  $\alpha > 0$ is the step size
- Generate the sequence $\boldsymbol{w}^1, \boldsymbol{w}^2, \cdots$

  converge to stationary points ($\lim_{t \to \infty} \|\nabla f(\boldsymbol{w}^t)\| = 0$)

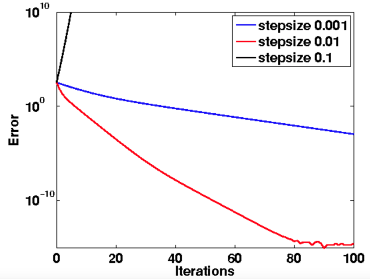
Gradient Descent

- Gradient descent: repeatedly do

$$\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t - \alpha \nabla f(\boldsymbol{w}^t)$$

  $\alpha > 0$ is the step size
- Generate the sequence $\boldsymbol{w}^1, \boldsymbol{w}^2, \cdots$

  converge to stationary points ($\lim_{t\to\infty} \|\nabla f(\boldsymbol{w}^t)\| = 0$)
- Step size too large $\Rightarrow$ diverge; too small $\Rightarrow$ slow convergence



Why gradient descent?

- Successive approximation view

  At each iteration, form an approximation function of $f(\cdot)$:

$$f(\boldsymbol{w}^t + \boldsymbol{d}) \approx g(\boldsymbol{d}) := f(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)^T \boldsymbol{d} + \frac{1}{2\alpha}\|\boldsymbol{d}\|^2$$

  Update solution by $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t + \boldsymbol{d}^*$

$$\boldsymbol{d}^* = \arg\min_{\boldsymbol{d}} g(\boldsymbol{d})$$

$$\nabla g(\boldsymbol{d}^*) = 0 \Rightarrow \nabla f(\boldsymbol{w}^t) + \frac{1}{\alpha}\boldsymbol{d}^* = 0 \Rightarrow \quad \boldsymbol{d}^* = -\alpha \nabla f(\boldsymbol{w}^t)$$

Why gradient descent?

- Successive approximation view

  At each iteration, form an approximation function of $f(\cdot)$:

$$f(\boldsymbol{w}^t + \boldsymbol{d}) \approx g(\boldsymbol{d}) := f(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)^T \boldsymbol{d} + \frac{1}{2\alpha}\|\boldsymbol{d}\|^2$$

  Update solution by $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t + \boldsymbol{d}^*$

$$\boldsymbol{d}^* = \arg\min_{\boldsymbol{d}} g(\boldsymbol{d})$$

$$\nabla g(\boldsymbol{d}^*) = 0 \Rightarrow \nabla f(\boldsymbol{w}^t) + \frac{1}{\alpha}\boldsymbol{d}^* = 0 \Rightarrow \quad \boldsymbol{d}^* = -\alpha \nabla f(\boldsymbol{w}^t)$$

- $\boldsymbol{d}^*$ will decrease $f(\cdot)$ if α (step size) is sufficiently small

Illustration of gradient descent

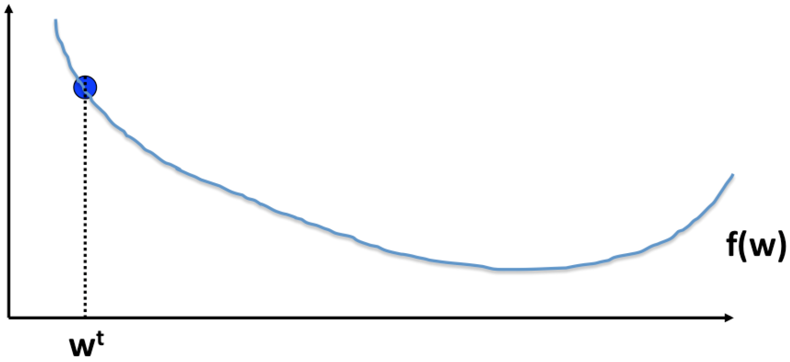
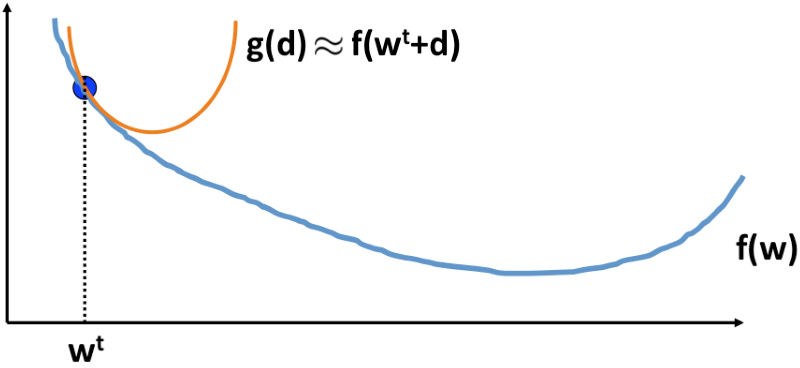


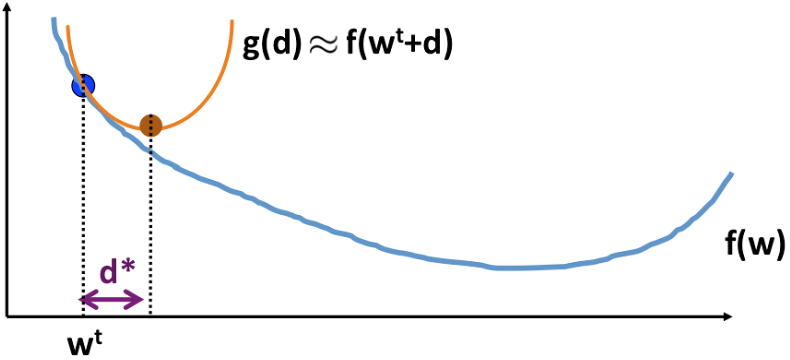
Illustration of gradient descent



Form a quadratic approximation

$$f(\boldsymbol{w}^t + \boldsymbol{d}) \approx g(\boldsymbol{d}) = f(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)^T \boldsymbol{d} + \frac{1}{2\alpha} \|\boldsymbol{d}\|^2$$

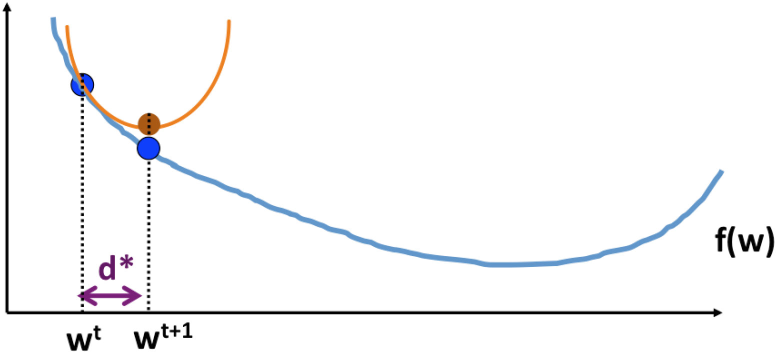
Illustration of gradient descent



Minimize $g(\boldsymbol{d})$:

$$\nabla g(\boldsymbol{d}^*) = 0 \Rightarrow \nabla f(\boldsymbol{w}^t) + \frac{1}{\alpha}\boldsymbol{d}^* = 0 \Rightarrow \boldsymbol{d}^* = -\alpha \nabla f(\boldsymbol{w}^t)$$

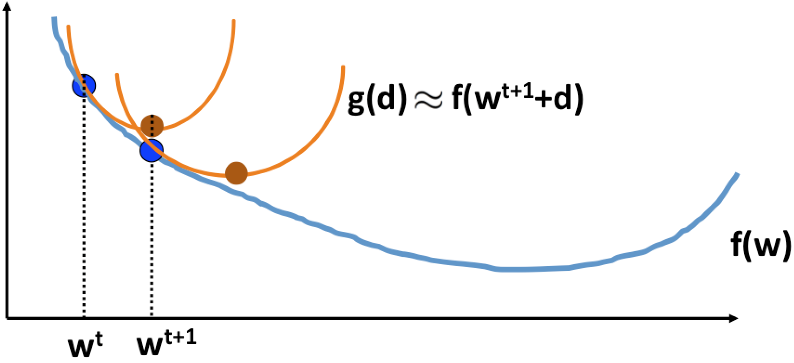
Illustration of gradient descent



Update

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \boldsymbol{d}^* = \boldsymbol{w}^t - \alpha \nabla f(\boldsymbol{w}^t)$$

Illustration of gradient descent

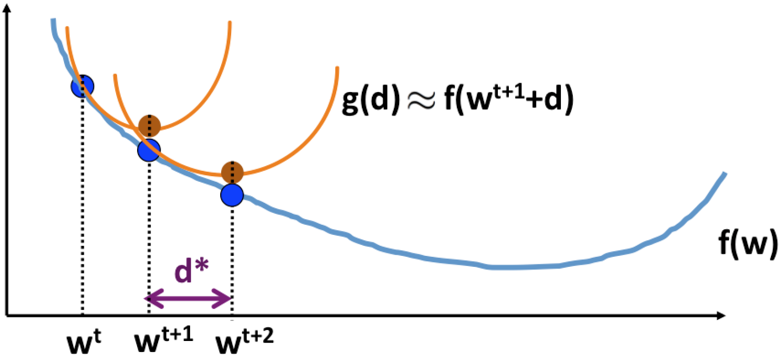


Form another quadratic approximation

$$f(\boldsymbol{w}^{t+1} + \boldsymbol{d}) \approx g(\boldsymbol{d}) = f(\boldsymbol{w}^{t+1}) + \nabla f(\boldsymbol{w}^{t+1})^T \boldsymbol{d} + \frac{1}{2\alpha}\|\boldsymbol{d}\|^2$$

$$\boldsymbol{d}^* = -\alpha \nabla f(\boldsymbol{w}^{t+1})$$

Illustration of gradient descent

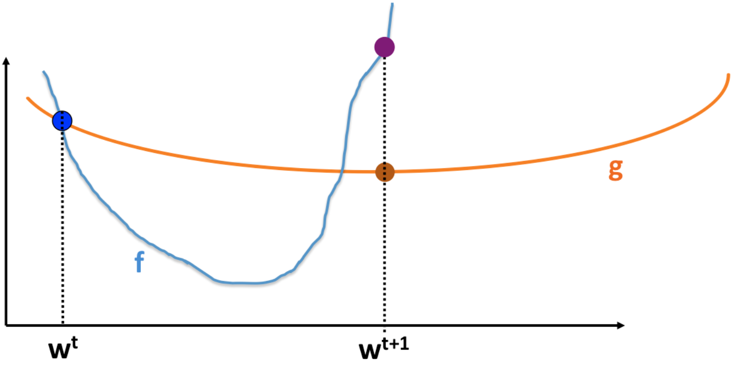


Update

$$\boldsymbol{w}^{t+2} = \boldsymbol{w}^{t+1} + \boldsymbol{d}^{*} = \boldsymbol{w}^{t+1} - \alpha \nabla f(\boldsymbol{w}^{t+1})$$

When will it diverge?

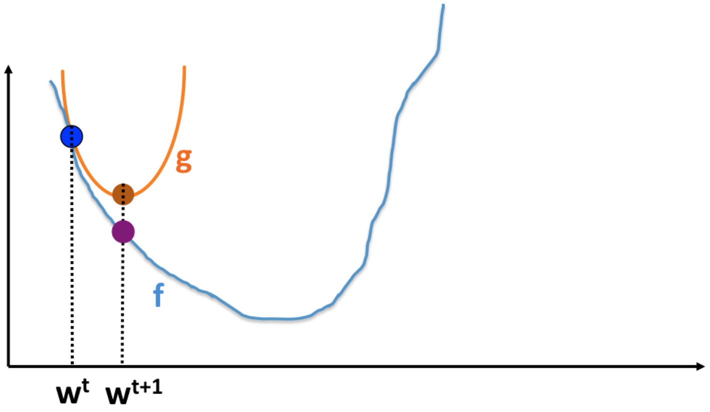
Can diverge ($f(\boldsymbol{w}^t) < f(\boldsymbol{w}^{t+1})$) if g is not an upperbound of f



f(w^t) < f(w^{t+1}), diverge because g's curvature is too small

When will it converge?

Always converge ($f(\boldsymbol{w}^t) > f(\boldsymbol{w}^{t+1})$) when g is an upperbound of f



$f(w^t) > f(w^{t+1})$, converge when g's curvature is large enough

Convergence

- Let L be the Lipchitz constant

  $(\nabla^2 f(\boldsymbol{x}) \preceq LI$ for all $\boldsymbol{x})$
- **Theorem:** gradient descent converges if $\alpha < \frac{1}{L}$

Convergence

- Let L be the Lipchitz constant

  $(\nabla^2 f(\boldsymbol{x}) \preceq LI$ for all $\boldsymbol{x})$
- **Theorem:** gradient descent converges if $\alpha < \frac{1}{L}$
- Why?
  - When $\alpha < 1/L$, for any $\boldsymbol{d}$,

$$
\begin{aligned}
g(\boldsymbol{d}) &= f(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)^T \boldsymbol{d} + \frac{1}{2\alpha}\|\boldsymbol{d}\|^2 \\
&> f(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)^T \boldsymbol{d} + \frac{L}{2}\|\boldsymbol{d}\|^2 \\
&\geq f(\boldsymbol{w}^t + \boldsymbol{d})
\end{aligned}
$$

Convergence

- Let L be the Lipchitz constant
  $(\nabla^2 f(\boldsymbol{x}) \preceq LI$ for all $\boldsymbol{x})$
- **Theorem:** gradient descent converges if $\alpha < \frac{1}{L}$
- Why?
  - When $\alpha < 1/L$, for any $\boldsymbol{d}$,

$$\begin{aligned} g(\boldsymbol{d}) &= f(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)^T \boldsymbol{d} + \frac{1}{2\alpha}\|\boldsymbol{d}\|^2 \\ &> f(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)^T \boldsymbol{d} + \frac{L}{2}\|\boldsymbol{d}\|^2 \\ &\geq f(\boldsymbol{w}^t + \boldsymbol{d}) \end{aligned}$$

  - So, $f(\boldsymbol{w}^t + \boldsymbol{d}^*) < g(\boldsymbol{d}^*) \leq g(0) = f(\boldsymbol{w}^t)$

Convergence

- Let L be the Lipchitz constant

  $(\nabla^2 f(\boldsymbol{x}) \preceq LI$ for all $\boldsymbol{x})$
- **Theorem:** gradient descent converges if $\alpha < \frac{1}{L}$
- Why?
  - When $\alpha < 1/L$, for any $\boldsymbol{d}$,

$$
\begin{aligned}
g(\boldsymbol{d}) &= f(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)^T \boldsymbol{d} + \frac{1}{2\alpha}\|\boldsymbol{d}\|^2 \\
&> f(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)^T \boldsymbol{d} + \frac{L}{2}\|\boldsymbol{d}\|^2 \\
&\geq f(\boldsymbol{w}^t + \boldsymbol{d})
\end{aligned}
$$

  - So, $f(\boldsymbol{w}^t + \boldsymbol{d}^*) < g(\boldsymbol{d}^*) \leq g(0) = f(\boldsymbol{w}^t)$
  - In formal proof, need to show $f(\boldsymbol{w}^t + \boldsymbol{d}^*)$ is sufficiently smaller than $f(\boldsymbol{w}^t)$

Applying to Logistic regression

gradient descent for logistic regression

- Initialize the weights $\boldsymbol{w}_0$
- For $t = 1, 2, \cdots$
  - Compute the gradient

$$\nabla f(\boldsymbol{w}) = -\frac{1}{N}\sum_{n=1}^{N} \frac{y_n \boldsymbol{x}_n}{1 + e^{y_n \boldsymbol{w}^T \boldsymbol{x}_n}}$$

  - Update the weights: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \nabla f(\boldsymbol{w})$
- Return the final weights $\boldsymbol{w}$

Applying to Logistic regression

gradient descent for logistic regression

- Initialize the weights $\boldsymbol{w}_0$
- For $t = 1, 2, \cdots$
  - Compute the gradient

$$\nabla f(\boldsymbol{w}) = -\frac{1}{N}\sum_{n=1}^{N}\frac{y_n\boldsymbol{x}_n}{1+e^{y_n\boldsymbol{w}^T\boldsymbol{x}_n}}$$

  - Update the weights: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha\nabla f(\boldsymbol{w})$
- Return the final weights $\boldsymbol{w}$

When to stop?

- Fixed number of iterations, or
- Stop when $\|\nabla f(\boldsymbol{w})\| < \epsilon$

Line Search

- In practice, we do not know $L \cdots$

  need to tune step size when running gradient descent

Line Search

- In practice, we do not know $L \cdots$

  need to tune step size when running gradient descent
- Line Search: Select step size automatically (for gradient descent)

Line Search

- The back-tracking line search:
  - Start from some large α_0
  - Try $\alpha = \alpha_0, \frac{\alpha_0}{2}, \frac{\alpha_0}{4}, \cdots$

    Stop when α satisfies some sufficient decrease condition

Line Search

- The back-tracking line search:
  - Start from some large α_0
  - Try $\alpha = \alpha_0, \frac{\alpha_0}{2}, \frac{\alpha_0}{4}, \cdots$
    Stop when α satisfies some sufficient decrease condition
  - A simple condition: $f(\boldsymbol{w} + \alpha \boldsymbol{d}) < f(\boldsymbol{w})$

Line Search

- The back-tracking line search:
  - Start from some large α_0
  - Try $\alpha = \alpha_0, \frac{\alpha_0}{2}, \frac{\alpha_0}{4}, \cdots$

    Stop when α satisfies some sufficient decrease condition
  - A simple condition: $f(\boldsymbol{w} + \alpha \boldsymbol{d}) < f(\boldsymbol{w})$

    often works in practice but doesn't work in theory

Line Search

- The back-tracking line search:
  - Start from some large α_0
  - Try $\alpha = \alpha_0, \frac{\alpha_0}{2}, \frac{\alpha_0}{4}, \cdots$
    Stop when α satisfies some sufficient decrease condition
  - A simple condition: $f(\boldsymbol{w} + \alpha \boldsymbol{d}) < f(\boldsymbol{w})$
    often works in practice but doesn't work in theory
  - A (provable) sufficient decrease condition:

$$f(\boldsymbol{w} + \alpha \boldsymbol{d}) \leq f(\boldsymbol{w}) + \sigma \alpha \nabla f(\boldsymbol{w})^T \boldsymbol{d}$$

  for a constant $\sigma \in (0, 1)$

Line Search

gradient descent with backtracking line search

- Initialize the weights $\boldsymbol{w}_0$
- For $t = 1, 2, \cdots$
  - Compute the gradient
    $$\boldsymbol{d} = -\nabla f(\boldsymbol{w})$$
  - For $\alpha = \alpha_0, \alpha_0/2, \alpha_0/4, \cdots$
    Break if $f(\boldsymbol{w} + \alpha\boldsymbol{d}) \leq f(\boldsymbol{w}) + \sigma\alpha\nabla f(\boldsymbol{w})^T\boldsymbol{d}$
  - Update $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\boldsymbol{d}$
- Return the final solution $\boldsymbol{w}$

Stochastic Gradient descent

Large-scale Problems

- Machine learning: usually minimizing the training loss

$$\min_{\boldsymbol{w}} \{ \frac{1}{N} \sum_{n=1}^{N} \ell(\boldsymbol{w}^T \boldsymbol{x}_n, y_n) \} := f(\boldsymbol{w}) \text{ (linear model)}$$

$$\min_{\boldsymbol{w}} \{ \frac{1}{N} \sum_{n=1}^{N} \ell(h_{\boldsymbol{w}}(\boldsymbol{x}_n), y_n) \} := f(\boldsymbol{w}) \text{ (general hypothesis)}$$

  ℓ: loss function (e.g., $\ell(a, b) = (a - b)^2$)

- Gradient descent:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \underbrace{\nabla f(\boldsymbol{w})}_{\text{Main computation}}$$

Large-scale Problems

- Machine learning: usually minimizing the training loss

$$\min_{\boldsymbol{w}} \{ \frac{1}{N} \sum_{n=1}^{N} \ell(\boldsymbol{w}^T \boldsymbol{x}_n, y_n) \} := f(\boldsymbol{w}) \text{ (linear model)}$$

$$\min_{\boldsymbol{w}} \{ \frac{1}{N} \sum_{n=1}^{N} \ell(h_{\boldsymbol{w}}(\boldsymbol{x}_n), y_n) \} := f(\boldsymbol{w}) \text{ (general hypothesis)}$$

ℓ: loss function (e.g., $\ell(a, b) = (a - b)^2$)

- Gradient descent:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \underbrace{\nabla f(\boldsymbol{w})}_{\text{Main computation}}$$

- In general, $f(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} f_n(\boldsymbol{w})$,

  each $f_n(\boldsymbol{w})$ only depends on $(\boldsymbol{x}_n, y_n)$

Stochastic gradient

- Gradient:

$$\nabla f(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N} \nabla f_n(\boldsymbol{w})$$

- Each gradient computation needs to go through all training samples

  slow when millions of samples

- Faster way to compute "approximate gradient"?

Stochastic gradient

- Gradient:

$$\nabla f(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}\nabla f_n(\boldsymbol{w})$$

- Each gradient computation needs to go through all training samples

    slow when millions of samples

- Faster way to compute "approximate gradient"?
- Use stochastic sampling:
    - Sample a small subset $B \subseteq \{1, \cdots, N\}$
    - Estimated gradient

$$\nabla f(\boldsymbol{w}) \approx \frac{1}{|B|}\sum_{n \in B}\nabla f_n(\boldsymbol{w})$$

    $|B|$: batch size

Stochastic gradient descent

Stochastic Gradient Descent (SGD)

- Input: training data $\{\boldsymbol{x}_n, y_n\}_{n=1}^{N}$
- Initialize $\boldsymbol{w}$ (zero or random)
- For $t = 1, 2, \cdots$
  - Sample a small batch $B \subseteq \{1, \cdots, N\}$
  - Update parameter

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta^t \frac{1}{|B|} \sum_{n \in B} \nabla f_n(\boldsymbol{w})$$

Stochastic gradient descent

Stochastic Gradient Descent (SGD)

- Input: training data $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$
- Initialize $\boldsymbol{w}$ (zero or random)
- For $t = 1, 2, \cdots$
  - Sample a small batch $B \subseteq \{1, \cdots, N\}$
  - Update parameter

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta^t \frac{1}{|B|} \sum_{n \in B} \nabla f_n(\boldsymbol{w})$$

Extreme case: $|B| = 1 \Rightarrow$ **Sample one training data at a time**

Logistic Regression by SGD

- Logistic regression:

$$\min_{\boldsymbol{w}} \frac{1}{N} \sum_{n=1}^{N} \underbrace{\log(1 + e^{-y_n \boldsymbol{w}^T \boldsymbol{x}_n})}_{f_n(\boldsymbol{w})}$$

SGD for Logistic Regression

- Input: training data $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$
- Initialize $\boldsymbol{w}$ (zero or random)
- For $t = 1, 2, \cdots$
  - Sample a batch $B \subseteq \{1, \cdots, N\}$
  - Update parameter

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta^t \frac{1}{|B|} \sum_{i \in B} \underbrace{\frac{-y_n \boldsymbol{x}_n}{1 + e^{y_n \boldsymbol{w}^T \boldsymbol{x}_n}}}_{\nabla f_n(\boldsymbol{w})}$$

Why SGD works?

- Stochastic gradient is an unbiased estimator of full gradient:

$$
\begin{aligned}
E[\frac{1}{|B|}\sum_{n\in B}\nabla f_n(\boldsymbol{w})] &= \frac{1}{N}\sum_{n=1}^{N}\nabla f_n(\boldsymbol{w}) \\
&= \nabla f(\boldsymbol{w})
\end{aligned}
$$

Why SGD works?

- Stochastic gradient is an unbiased estimator of full gradient:

$$E[\frac{1}{|B|}\sum_{n\in B}\nabla f_n(\boldsymbol{w})] = \frac{1}{N}\sum_{n=1}^{N}\nabla f_n(\boldsymbol{w})$$
$$= \nabla f(\boldsymbol{w})$$

- Each iteration updated by

gradient + zero-mean noise

Stochastic gradient descent

- In gradient descent, η (step size) is a fixed constant
- Can we use fixed step size for SGD?

Stochastic gradient descent

- In gradient descent, η (step size) is a fixed constant
- Can we use fixed step size for SGD?
- SGD with fixed step size cannot converge to global/local minimizers

Stochastic gradient descent

- In gradient descent, η (step size) is a fixed constant
- Can we use fixed step size for SGD?
- SGD with fixed step size cannot converge to global/local minimizers
- If $\boldsymbol{w}^*$ is the minimizer, $\nabla f(\boldsymbol{w}^*) = \frac{1}{N}\sum_{n=1}^{N} \nabla f_n(\boldsymbol{w}^*) = 0$,

Stochastic gradient descent

- In gradient descent, η (step size) is a fixed constant
- Can we use fixed step size for SGD?
- SGD with fixed step size cannot converge to global/local minimizers
- If $\boldsymbol{w}^*$ is the minimizer, $\nabla f(\boldsymbol{w}^*) = \frac{1}{N}\sum_{n=1}^{N} \nabla f_n(\boldsymbol{w}^*) = 0$,

$$\text{but} \quad \frac{1}{|B|}\sum_{n \in B} \nabla f_n(\boldsymbol{w}^*) \neq 0 \quad \text{if } B \text{ is a subset}$$

Stochastic gradient descent

- In gradient descent, η (step size) is a fixed constant
- Can we use fixed step size for SGD?
- SGD with fixed step size cannot converge to global/local minimizers
- If $\boldsymbol{w}^*$ is the minimizer, $\nabla f(\boldsymbol{w}^*) = \frac{1}{N}\sum_{n=1}^{N} \nabla f_n(\boldsymbol{w}^*) = 0$,

$$\text{but} \quad \frac{1}{|B|}\sum_{n\in B} \nabla f_n(\boldsymbol{w}^*) \neq 0 \quad \text{if } B \text{ is a subset}$$

(Even if we got minimizer, SGD will move away from it)

Stochastic gradient descent, step size

- To make SGD converge:

  Step size should decrease to 0

  $\eta^t \to 0$

  Usually with polynomial rate: $\eta^t \approx t^{-a}$ with constant a

Stochastic gradient descent vs Gradient descent

Stochastic gradient descent:

- pros:

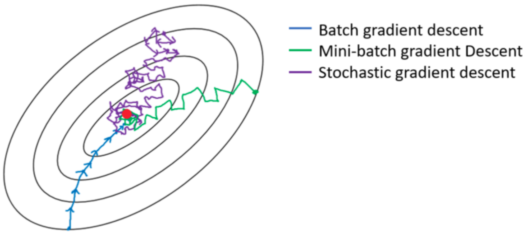
  cheaper computation per iteration

  faster convergence in the beginning

- cons:

  less stable, slower final convergence

  hard to tune step size



(Figure from https://medium.com/@ImadPhd/gradient-descent-algorithm-and-its-variants-10f652806a3)

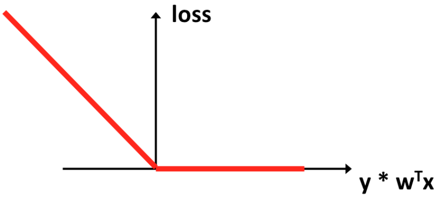
Revisit perceptron Learning Algorithm

- Given a classification data $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$
- Learning a linear model:

$$\min_{\boldsymbol{w}} \frac{1}{N} \sum_{n=1}^{N} \ell(\boldsymbol{w}^T \boldsymbol{x}_n, y_n)$$

- Consider the loss:

$$\ell(\boldsymbol{w}^T \boldsymbol{x}_n, y_n) = \max(0, -y_n \boldsymbol{w}^T \boldsymbol{x}_n)$$



What's the gradient?

Revisit perceptron Learning Algorithm

$$\ell(\boldsymbol{w}^T \boldsymbol{x}_n, y_n) = \max(0, -y_n \boldsymbol{w}^T \boldsymbol{x}_n)$$

Consider two cases:

- Case I: $y_n \boldsymbol{w}^T \boldsymbol{x}_n > 0$ (prediction correct)
  - $\ell(\boldsymbol{w}^T \boldsymbol{x}_n, y_n) = 0$
  - $\frac{\partial}{\partial \boldsymbol{w}} \ell(\boldsymbol{w}^T \boldsymbol{x}_n, y_n) = 0$

Revisit perceptron Learning Algorithm

$$\ell(\boldsymbol{w}^T\boldsymbol{x}_n, y_n) = \max(0, -y_n\boldsymbol{w}^T\boldsymbol{x}_n)$$

Consider two cases:

- Case I: $y_n\boldsymbol{w}^T\boldsymbol{x}_n > 0$ (prediction correct)
  - $\ell(\boldsymbol{w}^T\boldsymbol{x}_n, y_n) = 0$
  - $\frac{\partial}{\partial \boldsymbol{w}}\ell(\boldsymbol{w}^T\boldsymbol{x}_n, y_n) = 0$
- Case II: $y_n\boldsymbol{w}^T\boldsymbol{x}_n < 0$ (prediction wrong)
  - $\ell(\boldsymbol{w}^T\boldsymbol{x}_n, y_n) = -y_n\boldsymbol{w}^T\boldsymbol{x}_n$
  - $\frac{\partial}{\partial \boldsymbol{w}}\ell(\boldsymbol{w}^T\boldsymbol{x}_n, y_n) = -y_n\boldsymbol{x}_n$

Revisit perceptron Learning Algorithm

$$\ell(\boldsymbol{w}^T \boldsymbol{x}_n, y_n) = \max(0, -y_n \boldsymbol{w}^T \boldsymbol{x}_n)$$

Consider two cases:

- Case I: $y_n \boldsymbol{w}^T \boldsymbol{x}_n > 0$ (prediction correct)
  - $\ell(\boldsymbol{w}^T \boldsymbol{x}_n, y_n) = 0$
  - $\frac{\partial}{\partial \boldsymbol{w}} \ell(\boldsymbol{w}^T \boldsymbol{x}_n, y_n) = 0$
- Case II: $y_n \boldsymbol{w}^T \boldsymbol{x}_n < 0$ (prediction wrong)
  - $\ell(\boldsymbol{w}^T \boldsymbol{x}_n, y_n) = -y_n \boldsymbol{w}^T \boldsymbol{x}_n$
  - $\frac{\partial}{\partial \boldsymbol{w}} \ell(\boldsymbol{w}^T \boldsymbol{x}_n, y_n) = -y_n \boldsymbol{x}_n$

SGD update rule: Sample an index n

$$\boldsymbol{w}^{t+1} \leftarrow \begin{cases} \boldsymbol{w}^t & \text{if } y_n \boldsymbol{w}^T \boldsymbol{x}_n \geq 0 \text{ (predict correct)} \\ \boldsymbol{w}^t + \eta^t y_n \boldsymbol{x}_n & \text{if } y_n \boldsymbol{w}^T \boldsymbol{x}_n < 0 \text{ (predict wrong)} \end{cases}$$

Equivalent to Perceptron Learning Algorithm when $\eta^t = 1$

Conclusions

- Gradient descent
- Stochastic gradient descent

Questions?