STOR566: Introduction to Deep Learning

Lecture 4: Optimization

Yao Li
UNC Chapel Hill

Aug 29, 2024

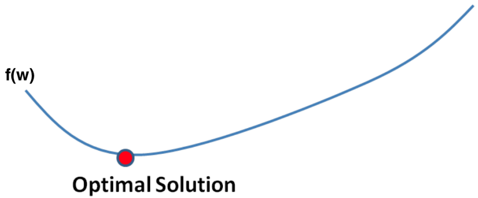
Materials are from *Learning from data (Caltech)* and *Deep Learning (UCLA)*

Optimization

- Goal: find the minimizer of a function

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

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



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 when α is sufficiently small

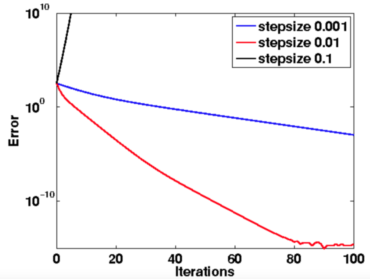
Gradient Descent

- 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 when α is sufficiently small
- Step size too large $\Rightarrow$ diverge; too small $\Rightarrow$ slow convergence



Convergence

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

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: $f(\boldsymbol{w} + \alpha \boldsymbol{d}) < f(\boldsymbol{w})$
  - 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

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

Why SGD works?

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

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

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?
- 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, 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
- pros:

  cheaper computation per iteration
- cons:

  less stable, slower final convergence

Momentum

- Gradient descent: only using current gradient (local information)
- Momentum: use previous gradient information

Momentum

- Gradient descent: only using current gradient (local information)
- Momentum: use previous gradient information
- The momentum update rule:

$$
\begin{aligned}
\boldsymbol{v}_t &= \beta \boldsymbol{v}_{t-1} + (1-\beta)\nabla f(\boldsymbol{w}_t) \\
\boldsymbol{w}_{t+1} &= \boldsymbol{w}_t - \alpha \boldsymbol{v}_t
\end{aligned}
$$

$\beta \in [0, 1)$: discount factors, α: step size

Momentum

- Gradient descent: only using current gradient (local information)
- Momentum: use previous gradient information
- The momentum update rule:

$$\begin{aligned} \boldsymbol{v}_t &= \beta \boldsymbol{v}_{t-1} + (1-\beta)\nabla f(\boldsymbol{w}_t) \\ \boldsymbol{w}_{t+1} &= \boldsymbol{w}_t - \alpha \boldsymbol{v}_t \end{aligned}$$

$\beta \in [0, 1)$: discount factors, α: step size

- Equivalent to using exponential moving average of gradients:

$$\boldsymbol{v}_t = (1-\beta)\nabla f(\boldsymbol{w}_t) + \beta(1-\beta)\nabla f(\boldsymbol{w}_{t-1}) + \beta^2(1-\beta)\nabla f(\boldsymbol{w}_{t-2}) + \cdots$$

Momentum

- Gradient descent: only using current gradient (local information)
- Momentum: use previous gradient information
- The momentum update rule:

$$\begin{aligned} \boldsymbol{v}_t &= \beta \boldsymbol{v}_{t-1} + (1-\beta)\nabla f(\boldsymbol{w}_t) \\ \boldsymbol{w}_{t+1} &= \boldsymbol{w}_t - \alpha \boldsymbol{v}_t \end{aligned}$$

  $\beta \in [0, 1)$: discount factors, α: step size
- Equivalent to using exponential moving average of gradients:

$$\boldsymbol{v}_t = (1-\beta)\nabla f(\boldsymbol{w}_t) + \beta(1-\beta)\nabla f(\boldsymbol{w}_{t-1}) + \beta^2(1-\beta)\nabla f(\boldsymbol{w}_{t-2}) + \cdots$$

Momentum

In practice: Usually replace $(1-\beta)$ with α

- The momentum update rule:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \boldsymbol{d}_t$$

Replace $(1-\beta)$ with α

$$\boldsymbol{d}_t = \alpha \nabla f(\boldsymbol{w}_t) + \alpha\beta \nabla f(\boldsymbol{w}_{t-1}) + \alpha\beta^2 \nabla f(\boldsymbol{w}_{t-2}) + \cdots$$

- $\boldsymbol{d}_t =$Gradient descent direction + Decayed Previous Descent Direction

Momentum

In practice: Usually replace $(1-\beta)$ with α

- The momentum update rule:

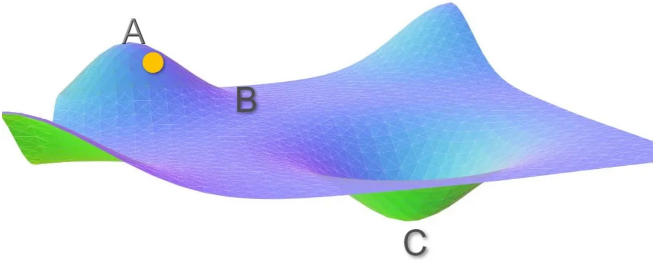
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{d}_t$$

Replace $(1-\beta)$ with α

$$\mathbf{d}_t = \alpha \nabla f(\mathbf{w}_t) + \alpha\beta \nabla f(\mathbf{w}_{t-1}) + \alpha\beta^2 \nabla f(\mathbf{w}_{t-2}) + \cdots$$

- $\mathbf{d}_t$ =Gradient descent direction + Decayed Previous Descent Direction
- Small β: Contribution from earlier gradients decreases rapidly
- Large β: Accommodate more gradients from the past.
- In practice: β is generally around 0.9

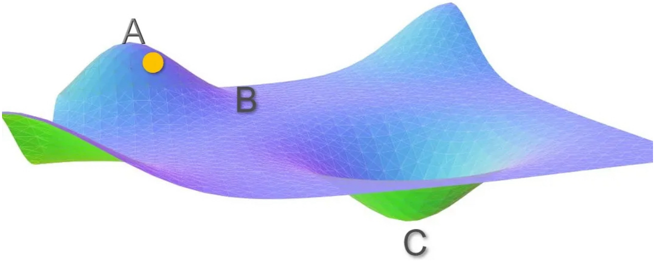
Momumtum Explanation



(Image from *Gradient Descent with Momentum* on Medium)

- Initial Point A, Saddle Point B, Global Minimum C

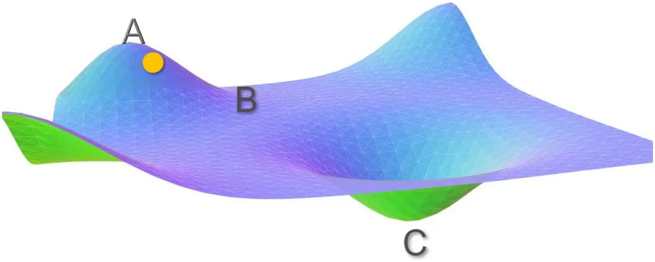
Momumtum Explanation



(Image from *Gradient Descent with Momentum* on Medium)

- Initial Point A, Saddle Point B, Global Minimum C
- Gradient Descent: Do well between AB, but stuck at Point B

Momumtum Explanation



(Image from *Gradient Descent with Momentum* on Medium)

- Initial Point A, Saddle Point B, Global Minimum C
- Gradient Descent: Do well between AB, but stuck at Point B
- Momentum: accumulated momentum push it across B

$$\boldsymbol{d}_t = \alpha \nabla f(\boldsymbol{w}_t) + \alpha\beta \nabla f(\boldsymbol{w}_{t-1}) + \alpha\beta^2 \nabla f(\boldsymbol{w}_{t-2}) + \cdots$$

Momentum gradient descent

Momentum gradient descent

- Initialize $\boldsymbol{w}_0, \boldsymbol{d}_0 = 0$
- For $t = 1, 2, \cdots$
  - Compute $\boldsymbol{d}_t \leftarrow \alpha \nabla f(\boldsymbol{w}_t) + \beta \boldsymbol{d}_{t-1}$
  - Update $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \boldsymbol{d}_t$

α: learning rate

β: discount factor ($\beta = 0$ means no momentum)

Momentum stochastic gradient descent

Optimizing $f(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{w})$

Momentum stochastic gradient descent

- Initialize $\boldsymbol{w}_0, \boldsymbol{d}_0 = 0$
- For $t = 1, 2, \cdots$
  - Sample a small batch $B \subseteq \{1, \cdots, N\}$
  - Compute $\boldsymbol{d}_t \leftarrow \alpha \frac{1}{|B|} \sum_{i \in B} \nabla f_i(\boldsymbol{w}_t) + \beta \boldsymbol{d}_{t-1}$
  - Update $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \boldsymbol{d}_t$

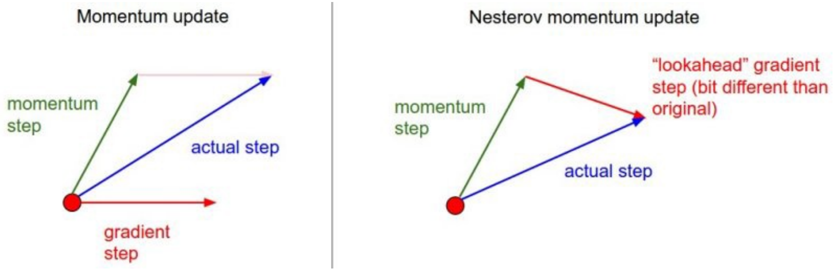
α: learning rate

β: discount factor ($\beta = 0$ means no momentum)

Nesterov accelerated gradient

- Using the "look-ahead" gradient

$$\mathbf{v}_t = \beta \mathbf{v}_{t-1} + (1-\beta)\nabla f(\mathbf{w}_t - \beta \mathbf{v}_{t-1})$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \mathbf{v}_t$$



(Figure from https://towardsdatascience.com)

Adagrad: Adaptive updates (2010)

- SGD update: same step size for all variables
- Adaptive algorithms: each dimension can have a different step size

Adagrad: Adaptive updates (2010)

- SGD update: same step size for all variables
- Adaptive algorithms: each dimension can have a different step size

Adagrad

- Initialize $\boldsymbol{w}^{(0)}$
- For $t = 1, 2, \cdots$
  - Sample an $i \in \{1, \cdots, N\}$
  - Compute $\boldsymbol{g}^{(t)} \leftarrow \nabla f_i(\boldsymbol{w}^{(t)})$
  - $G_j^{(t)} \leftarrow G_j^{(t-1)} + (g_j^{(t)})^2$ for all $j = 1, \ldots d$
  - Update $w_j^{(t+1)} \leftarrow w_j^{(t)} - \frac{\eta}{\sqrt{G_j^{(t)} + \epsilon}} g_j^{(t)}$

η: step size (constant)

ϵ: small constant to avoid division by 0

- Adam: Momentum + Adaptive updates (2015)

Conclusions

- Gradient descent
- Stochastic gradient descent
- Variants

Questions?