ECE 595: Machine Learning I
Lecture 5.2: Stochastic Gradient Descent

Spring 2020

Stanley Chan

School of Electrical and Computer Engineering Purdue University

Outline

Mathematical Background

- Lecture 4: Intro to Optimization
- Lecture 5: Gradient Descent

Lecture 5: Gradient Descent

- Gradient Descent
  - Descent Direction
  - Step Size
  - Convergence
- Stochastic Gradient Descent
  - Difference between GD and SGD
  - Why does SGD work?

Stochastic Gradient Descent

Most loss functions in machine learning problems are **separable**:

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}(g_{\boldsymbol{\theta}}(\boldsymbol{x}^n), y^n) = \frac{1}{N}\sum_{n=1}^{N} J_n(\boldsymbol{\theta}). \tag{1}$$

For example,

- Square-loss:

$$J(\boldsymbol{\theta}) = \sum_{n=1}^{N} (g_{\boldsymbol{\theta}}(\boldsymbol{x}^n) - y^n)^2$$

- Cross-entropy loss:

$$J(\boldsymbol{\theta}) = -\sum_{n=1}^{N} \left\{ y^n \log g_{\boldsymbol{\theta}}(\boldsymbol{x}^n) + (1 - y^n) \log(1 - g_{\boldsymbol{\theta}}(\boldsymbol{x}^n)) \right\}$$

- Logistic loss:

$$J(\boldsymbol{\theta}) = \sum_{n=1}^{N} \log(1 + e^{-y^n \boldsymbol{\theta}^T \boldsymbol{x}^n})$$

Full Gradient VS Partial Gradient

Vanilla **gradient descent**:

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} - \underbrace{\eta^{t} \nabla J(\boldsymbol{\theta}^{t})}_{\text{main computation}} . \tag{2}$$

The full gradient of the loss is

$$\nabla J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} \nabla J_n(\boldsymbol{\theta}) \tag{3}$$

Stochastic gradient descent:

$$\nabla J(\boldsymbol{\theta}) \approx \frac{1}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \nabla J_n(\boldsymbol{\theta}) \tag{4}$$

where $\mathcal{B} \subseteq \{1, \ldots, N\}$ is a random subset. $|\mathcal{B}|$ = batch size.

SGD Algorithm

Algorithm (Stochastic Gradient Descent)

1. *Given* $\{(\boldsymbol{x}^n, y^n) \mid n = 1, \ldots, N\}$.
2. *Initialize* $\boldsymbol{\theta}$ *(zero or random)*
3. *For* $t = 1, 2, 3, \ldots$
   - *Draw a random subset* $\mathcal{B} \subseteq \{1, \ldots, N\}$.
   - *Update*

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \eta^t \frac{1}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \nabla J_n(\boldsymbol{\theta}) \tag{5}$$

- If $|\mathcal{B}| = 1$, then use only one sample at a time.
- The approximate gradient is **unbiased**: (See Appendix for Proof)

$$\mathbb{E}\left[\frac{1}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \nabla J_n(\boldsymbol{\theta})\right] = \nabla J(\boldsymbol{\theta}).$$

Interpreting SGD

- Just showed that the SGD step is unbiased:

$$\mathbb{E}\left[\frac{1}{|\mathcal{B}|}\sum_{n\in\mathcal{B}}\nabla J_n(\boldsymbol{\theta})\right] = \nabla J(\boldsymbol{\theta}).$$

- Unbiased gradient implies that each update is

gradient + zero-mean noise

- Step size: SGD with constant step size **does not converge**.
- If $\boldsymbol{\theta}^*$ is a minimizer, then $J(\boldsymbol{\theta}^*) = \frac{1}{N}\sum_{n=1}^{N} J_n(\boldsymbol{\theta}^*) = 0$. But

$$\frac{1}{|\mathcal{B}|}\sum_{n\in\mathcal{B}} J_n(\boldsymbol{\theta}^*) \neq 0, \qquad \text{since } \mathcal{B} \text{ is a subset.}$$

- Typical strategy: Start with large step size and gradually decrease: $\eta^t \to 0$, e.g., $\eta^t = t^{-a}$ for some constant a.

Perspectives of SGD

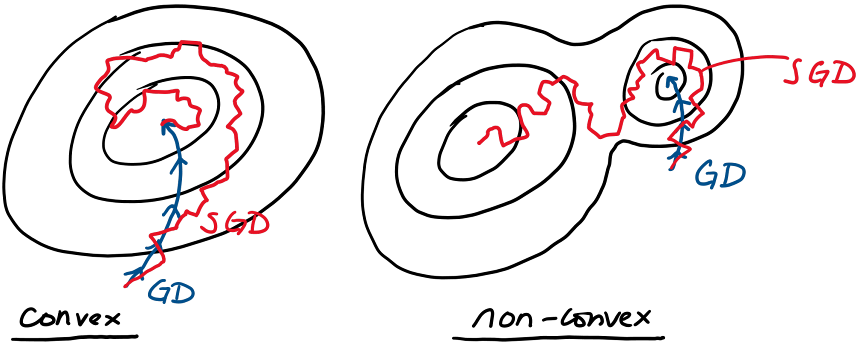
Classical optimization literature have the following observations.

- Compared to GD in **convex** problems:
- SGD offers a **trade-off** between accuracy and efficiency
- More iterations
- Less gradient evaluation per iteration
- Noise is a by-product

Recent studies of SGD for **non-convex** problems found that

- SGD for training deep neural networks works
- SGD finds solution faster
- SGD find a better local minima
- Noise matters

GD compared to SGD



Smoothing the Landscape

Analyzing SGD is an active research topic. Here is one by Kleinberg et al. (`https://arxiv.org/pdf/1802.06175.pdf` ICML 2018)

- The SGD step can be written as GD + noise:

$$\begin{aligned} \boldsymbol{x}^{t+1} &= \boldsymbol{x}^t - \eta(\nabla f(\boldsymbol{x}^t) + \boldsymbol{w}^t) \\ &= \underbrace{\boldsymbol{x}^t - \eta\nabla f(\boldsymbol{x}^t)}_{\overset{\text{def}}{=}\boldsymbol{y}^t} - \eta \boldsymbol{w}^t. \end{aligned}$$

- $\boldsymbol{y}^t$ is the "ideal" location returned by GD.
- Let us analyze $\boldsymbol{y}^{t+1}$:

$$\begin{aligned} \boldsymbol{y}^{t+1} &\overset{\text{def}}{=} \boldsymbol{x}^{t+1} - \eta\nabla f(\boldsymbol{x}^{t+1}) \\ &= (\boldsymbol{y}^t - \eta\boldsymbol{w}^t) - \eta\nabla f(\boldsymbol{y}^t - \eta\boldsymbol{w}^t) \end{aligned}$$

- Assume $\mathbb{E}[\boldsymbol{w}] = 0$, then

$$\mathbb{E}[\boldsymbol{y}^{t+1}] = \boldsymbol{y}^t - \eta\nabla\mathbb{E}[f(\boldsymbol{y}^t - \eta\boldsymbol{w}^t)]$$

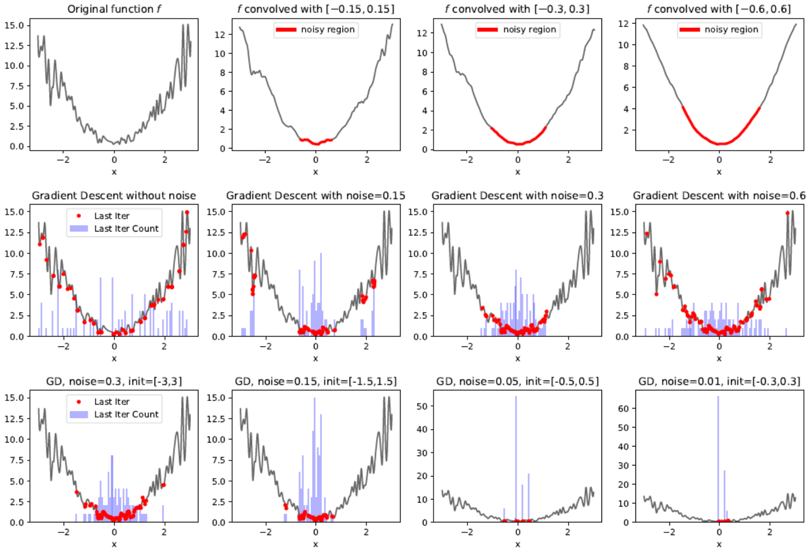
Smoothing the Landscape

- Let us look at $\mathbb{E}[f(\boldsymbol{y}^t - \eta \boldsymbol{w}^t)]$:

$$\mathbb{E}[f(\boldsymbol{y} - \eta \boldsymbol{w})] = \int f(\boldsymbol{y} - \eta \boldsymbol{w}) p(\boldsymbol{w}) \, d\boldsymbol{w},$$

  where $p(\boldsymbol{w})$ is the distribution of $\boldsymbol{w}$.
- $\int f(\boldsymbol{y} - \eta \boldsymbol{w}) p(\boldsymbol{w}) \, d\boldsymbol{w}$ is the **convolution** between f and p.
- $p(\boldsymbol{w}) \geq 0$ for all $\boldsymbol{w}$, so the convolution always **smoothes** the function.
- Learning rate controls the smoothness
- Too small: Under-smooth. You have not yet escaped from bad local minimum.
- Too large: Over-smooth. You may miss a local minimum.

Smoothing the Landscape



Reading List

Gradient Descent

- S. Boyd and L. Vandenberghe, "Convex Optimization", Chapter 9.2-9.4.
- J. Nocedal and S. Wright, "Numerical Optimization", Chapter 3.1-3.3.
- Y. Nesterov, "Introductory lectures on convex optimization", Chapter 2.
- CMU 10.725 Lecture https://www.stat.cmu.edu/~ryantibs/convexopt/lectures/grad-descent.pdf

Stochastic Gradient Descent

- CMU 10.725 Lecture https://www.stat.cmu.edu/~ryantibs/convexopt/lectures/stochastic-gd.pdf
- Kleinberg et al. (2018) "When Does SGD Escape Local Minima", https://arxiv.org/pdf/1802.06175.pdf