CS281A: Statistical Learning Theory **Fall 2025**

Lecture 11 — October 2

Lecturer: Benjamin Recht *Scribe: Shizhe Zhang*

11.1 Introduction

Gradient-based optimization lies at the core of modern machine learning. When the objective function is expressed as an average over many data points, computing the full gradient can be prohibitively expensive.

Stochastic Gradient Descent (SGD) is a randomized alternative to full-batch gradient descent that replaces the exact gradient with a stochastic approximation computed using a single sample or a small batch of data points. This makes SGD particularly suitable for large-scale optimization tasks such as training deep neural networks.

11.2 Problem Setup

We consider the general optimization problem of empirical risk minimization:

$$\min_{\theta \in \mathbb{R}^d} \mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, \theta),$$

where:

- $\theta \in \mathbb{R}^d$ denotes the parameter vector we wish to learn.
- $\{x_i\}_{i=1}^n$ are the observed training data.
- $\ell(x_i, \theta)$ is the loss function measuring prediction error on x_i.

In many statistical learning problems, n can be extremely large (e.g., millions of data points). Computing the gradient

$$\nabla \mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \nabla \ell(x_i, \theta)$$

requires a full pass over the dataset, which is computationally expensive.

This motivates the use of stochastic approximations that rely on fewer data points per update.

11.3 Gradient Descent Recap

Classical Gradient Descent iteratively updates the parameter vector θ_t according to the rule

$$\theta_{t+1} = \theta_t - \eta_t \nabla \mathcal{L}(\theta_t),$$

where $\eta_t > 0$ is the learning rate or step size at iteration t.

$$\nabla \mathcal{L}(\theta_t) = \frac{1}{n} \sum_{i=1}^{n} \nabla_{\theta_t} L(f(x_i; \theta_t), y_i)$$

$$= \frac{1}{n}\sum_{i=1}^{n} \frac{\partial L(f(x_i;\theta_t), y_i)}{\partial f}|_{f(x_i;\theta_t)} \cdot \nabla_{\theta_t} f(x_i;\theta_t)$$

$$:= \frac{1}{n}\sum_{i=1}^{n} e(\hat{y}_i, y_i) \cdot \nabla_{\theta_t} f(x_i;\theta_t)$$

Remark 1. The learning rate η_t controls how aggressively the algorithm moves in the direction of steepest descent. Too large a step may cause divergence, while too small a step may lead to extremely slow progress.

Limitations.

- Each update requires computing the full gradient $\nabla f(\theta_t)$, which involves all n data points.
- When n is very large, a single gradient computation may be as costly as training a model once.
- This makes GD impractical for large-scale machine learning tasks.

These limitations motivate the need for stochastic methods that approximate the gradient using a small subset of the data at each step.

11.4 Learning Rate Schedules

The step size (learning rate) η_t is crucial for the performance of SGD. Different schedules can lead to very different convergence behavior.

Constant Learning Rate

$$\eta_t = \eta \quad \text{for all } t.$$

- Fast initial decrease in loss.
- May oscillate around the minimum instead of converging exactly.

Diminishing Learning Rate

$$\eta_t = \frac{\eta_0}{1+\alpha t} \quad \text{or} \quad \eta_t = \frac{\eta_0}{\sqrt{t}} \quad \text{or} \quad \eta_t = 0.9\eta_{t-1}$$

- Guarantees convergence to the optimum for convex functions.
- Slower practical progress if η_t decays too fast.

Example 1. Consider this example with time-variant learning rate:

$$\min \frac{1}{2n}\sum_{i=1}^{n}(w - y_i)^2$$

where $\nabla \frac{1}{2}(w-y)^2 = (w-y)$, $\eta_t = \frac{1}{t}$

Thus,

$$w_1 = 0$$

$$w_2 = w_1 - \eta_1(w_1 - y_1) = \eta_1 y_1 = y_1$$
$$w_3 = w_2 - \eta_2(w_2 - y_2) = \frac{1}{2}(y_1 + y_2)$$
$$\vdots$$
$$w_k = \frac{1}{k-1}\sum_{i=1}^{k-1} y_i$$

Adaptive Learning Rates

Modern optimizers adapt learning rates dynamically:

- **AdaGrad:** scales coordinates inversely by the square root of the sum of past squared gradients.
- **RMSProp:** uses an exponential moving average of squared gradients.
- **Adam:** combines momentum with adaptive learning rates.

Remark. While diminishing step sizes are important for theoretical guarantees, constant or cyclical learning rate schedules are often used in practice, especially in deep learning, where they improve empirical performance.

Definition 1 (SGD Update Rule)**.** At iteration t, sample an index $i_t \in \{1, \dots, n\}$ uniformly at random and update

$$\theta_{t+1} = \theta_t - \eta_t \nabla \ell(x_{i_t}, \theta_t).$$

In practice, mini-batches of size $b \ll n$ are often used to reduce variance:

$$\theta_{t+1} = \theta_t - \eta_t \frac{1}{b}\sum_{j=1}^{b} \nabla \ell(x_{i_t^{(j)}}, \theta_t).$$

Algorithm: Stochastic Gradient Descent

Algorithm 1 Stochastic Gradient Descent

1: Initialize θ_0
2: **for** $t = 0, 1, \dots, T-1$ **do**
3: Sample index i_t uniformly from $\{1, \dots, n\}$
4: Compute stochastic gradient $g_t \leftarrow \nabla \ell(x_{i_t}, \theta_t)$
5: Update $\theta_{t+1} \leftarrow \theta_t - \eta_t g_t$
6: **end for**

Key Property. The stochastic gradient is an unbiased estimator of the true gradient:

$$\mathbb{E}[\nabla \ell(x_{i_t}, \theta_t)] = \nabla \mathcal{L}(\theta_t).$$

This unbiasedness ensures that, on average, SGD follows the true descent direction, even though individual updates are noisy.

11.5 Convergence Analysis

The convergence of SGD depends heavily on the assumptions made about the objective function $f(\theta)$ and the choice of step size η_t.

Convex Case

If $f(\theta)$ is convex, and the learning rate satisfies

$$\sum_{t=1}^{\infty} \eta_t = \infty, \quad \sum_{t=1}^{\infty} \eta_t^2 < \infty,$$

then SGD converges to the global optimum in expectation [1].

Convergence Rate. For strongly convex functions, the expected error decays at rate

$$\mathbb{E}[f(\theta_T) - f(\theta^*)] = O\left(\frac{1}{T}\right).$$

For general convex functions, the rate is slower:

$$\mathbb{E}[f(\theta_T) - f(\theta^*)] = O\left(\frac{1}{\sqrt{T}}\right).$$

Non-Convex Case

In deep learning, the objective is typically non-convex. While classical theory does not guarantee convergence to a global minimum, it has been shown that SGD often converges to *stationary points*:

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \|\nabla f(\theta_t)\|^2 = 0.$$

Practical Implications

- The noise in SGD updates can help escape saddle points, which are abundant in high-dimensional non-convex optimization.
- Properly chosen step sizes lead to good practical convergence, though theoretical guarantees are weaker compared to convex settings.

11.6 Variants of SGD

Over time, many improvements have been proposed to enhance the performance and stability of SGD.

Momentum

Momentum accelerates convergence by accumulating a velocity term:

$$v_{t+1} = \mu v_t + \eta_t \nabla \ell(x_{i_t}, \theta_t), \quad \theta_{t+1} = \theta_t - v_{t+1},$$

where $\mu \in (0, 1)$ is the momentum coefficient. This helps overcome oscillations in narrow valleys of the loss landscape.

Nesterov Accelerated Gradient (NAG)

A variant of momentum that looks ahead before computing the gradient:

$$v_{t+1} = \mu v_t + \eta_t \nabla \ell(x_{i_t}, \theta_t - \mu v_t).$$

AdaGrad

Adapts learning rates per coordinate using the accumulated squared gradients:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t,$$

where G_t is the diagonal matrix of past squared gradients and $\odot$ denotes elementwise multiplication.

RMSProp

Uses an exponential moving average of squared gradients to avoid AdaGrad's rapid decay of learning rates.

Adam

Combines RMSProp with momentum by maintaining both first and second moment estimates of the gradient. Widely used in deep learning due to its strong empirical performance.

Remark. While these adaptive methods often work well in practice, recent studies [5] suggest they may generalize worse than vanilla SGD with momentum, raising questions about their long-term reliability.

11.7 Applications

Stochastic Gradient Descent is the workhorse of large-scale machine learning. It has found widespread application across classical models and modern deep learning.

Linear Regression

Given data $\{(x_i, y_i)\}_{i=1}^n$, the loss function is the mean squared error

$$\mathcal{L}(x_i, y_i; \theta) = \big(y_i - \theta^\top x_i\big)^2.$$

SGD updates the parameter vector by computing the gradient on a single sample (or mini-batch):

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta \mathcal{L}(x_{i_t}, y_{i_t}; \theta_t).$$

Logistic Regression

For binary classification with labels $y_i \in \{0, 1\}$, the logistic loss is

$$\mathcal{L}(x_i, y_i; \theta) = -\big[y_i \log \sigma(\theta^\top x_i) + (1 - y_i) \log(1 - \sigma(\theta^\top x_i))\big],$$

where $\sigma(z) = 1/(1+e^{-z})$ is the sigmoid. SGD updates naturally extend to this probabilistic setting.

Neural Networks

In deep learning, $\mathcal{L}$ is typically the cross-entropy loss or mean squared error applied to outputs of multi-layer neural networks. SGD (often with momentum or Adam) is the standard method for training models with millions of parameters on large-scale datasets.

11.8 Advantages and Limitations

- **Advantages:** scalable to large datasets, efficient in memory.
- **Limitations:** noisy updates, sensitive to hyperparameter tuning.

11.9 Critique and Further Perspectives

While the lecture emphasized the classical convergence properties of SGD under convex assumptions, much of modern machine learning uses SGD in highly non-convex settings. The theory presented (e.g., diminishing learning rates) does not fully capture practical heuristics used in deep learning, where constant or cyclical learning rates often perform better [2].

Furthermore, recent work has questioned the effectiveness of adaptive optimizers such as Adam: although they achieve rapid training loss reduction, they may fail to converge to optimal generalization performance [5]. Practitioners should reconsider the use of adaptive methods to train neural networks.

In summary, SGD is a scalable optimization method essential for modern machine learning. It approximates gradients using random samples, converges under mild conditions, and forms the backbone of training algorithms for neural networks. However, its performance depends heavily on learning rate schedules and variance control. A more comprehensive view should integrate modern critiques and improvements to SGD that reflect both theoretical advances and empirical practices in large-scale machine learning.

Bibliography

[1] Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*.

[2] Bottou, L. (2010). Large-Scale Machine Learning with Stochastic Gradient Descent. In *Proceedings of COMPSTAT*.

[3] Bottou, L., Curtis, F., & Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review*.

[4] Kingma, D., & Ba, J. (2015). Adam: A method for stochastic optimization. *ICLR*.

[5] Wilson, A. et al. (2017). The marginal value of adaptive gradient methods in machine learning. *NeurIPS*.

[6] Johnson, R., & Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. *NeurIPS*.