Gradient Descent Methods:
Full Batch, Stochastic and Mini-Batch

Youssef SALMAN

November 10, 2025

Introduction

In optimization for Machine Learning, we often want to minimize a loss function

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} L(\theta; x_i),$$

where $x_1, \ldots, x_n$ are the training samples.

To minimize this loss, we use **gradient descent**. The update rule is always:

$$\theta_{k+1} = \theta_k - \eta\, g_k,$$

where:

- θ_k is the current parameter vector,
- g_k is an estimate of the gradient at iteration k,
- η is the **learning rate**.

Depending on how we compute g_k, we obtain three versions of gradient descent.

1 1. Full Batch Gradient Descent

Idea

Use **all** the dataset $\{x_1, \ldots, x_n\}$ to compute the gradient. This gives an exact and stable update.

Mathematical Form

$$g_k = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta L(\theta_k; x_i).$$

Advantages

- Very stable (no randomness).
- Exact direction of steepest descent.

Disadvantages

- Very slow for large n (must compute gradient on all samples).
- Not suitable for big datasets.

Steps (Algorithm)

1. Compute the full gradient:
$$g_k = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta L(\theta_k; x_i).$$
2. Update parameters:
$$\theta_{k+1} = \theta_k - \eta g_k.$$
3. Repeat until convergence.

2 2. Stochastic Gradient Descent (SGD)

Idea

At each iteration, select **one random sample** z from the dataset, and compute the gradient only using this sample.

$$g_k = \nabla_\theta L(\theta_k; z).$$

This gives a very fast but noisy update.

Why does this work?

Because:
$$\mathbb{E}[\,\nabla L(\theta; z)\,] = \frac{1}{n}\sum_{i=1}^{n} \nabla L(\theta; x_i)$$
so the average behavior matches the true gradient.

Advantages

- Extremely fast updates.
- Works with huge datasets.
- Can escape shallow local minima.

Disadvantages

- Very noisy.
- May oscillate around the minimum.

Steps (Algorithm)

1. Randomly choose one sample:
$$z \sim \{x_1, \dots, x_n\}.$$
2. Compute stochastic gradient:
$$g_k = \nabla_\theta L(\theta_k; z).$$
3. Update:
$$\theta_{k+1} = \theta_k - \eta g_k.$$
4. Repeat.

Example: what does "choose one sample" mean in SGD? In Machine Learning, a sample is usually a **vector of features** (and possibly a label), not a single number. For example, consider the dataset:

$$x_1 = (2, 1, 3), \qquad x_2 = (0, 4, 2), \qquad x_3 = (1, 2, 5), \qquad x_4 = (3, 0, 1).$$

At each iteration of SGD, we select one full observation at random:

$$z \sim \{x_1, x_2, x_3, x_4\}.$$

A possible sequence of chosen samples may be:

$$\begin{aligned} &\text{Iteration 1: } z = x_3 = (1,2,5), \\ &\text{Iteration 2: } z = x_1 = (2,1,3), \\ &\text{Iteration 3: } z = x_4 = (3,0,1), \\ &\text{Iteration 4: } z = x_2 = (0,4,2). \end{aligned}$$

The stochastic gradient at iteration k is computed using only this sample z:

$$g_k = \nabla_\theta L(\theta_k; z),$$

and the parameters are updated by

$$\theta_{k+1} = \theta_k - \eta\, g_k.$$

This process is repeated for many iterations (or epochs) until convergence.

When do we stop? Stochastic Gradient Descent does not stop after a single update. We repeat the procedure until a stopping criterion is satisfied. Common stopping conditions include:

- a fixed number of **epochs** (one epoch = a full pass through all n samples),
- the loss $J(\theta)$ stops decreasing (convergence),
- the gradient norm becomes small:

$$\|\nabla J(\theta_k)\| < \varepsilon,$$

- a maximum number of iterations is reached.

In practice, SGD is usually run for several epochs because a single sample does not contain enough information to reach the minimum.

3 3. Mini-Batch Gradient Descent

Idea

At each iteration, randomly choose a **small subset** (a mini-batch) of size m:

$$B_k = \{x_{i_1}, \dots, x_{i_m}\}.$$

Then compute the average gradient over this mini-batch:

$$g_k = \frac{1}{m} \sum_{x \in B_k} \nabla_\theta L(\theta_k; x).$$

Why does mini-batch converge?

Even though each update uses only $m < n$ samples, each sample has a chance to appear in many batches. On average:

$$\mathbb{E}[g_k] = \text{true gradient}.$$

Thus it converges (with proper learning rate).

Advantages

- Faster than full batch.
- More stable than SGD.
- Works very well with GPUs.

Disadvantages

- Still contains some noise.
- Must choose batch size m (common choice: 32, 64, 128).

Steps (Algorithm)

1. Randomly choose a mini-batch B_k of size m.
2. Compute mini-batch gradient:
$$g_k = \frac{1}{m} \sum_{x \in B_k} \nabla_\theta L(\theta_k; x).$$
3. Update:
$$\theta_{k+1} = \theta_k - \eta g_k.$$
4. Repeat.

When do we stop? Mini-Batch Gradient Descent also repeats the update many times. We continue until a stopping criterion is satisfied. Typical stopping conditions are:

- a fixed number of **epochs** (one epoch = using all n samples, divided into batches of size m),
- the loss function $J(\theta)$ stops decreasing,
- the gradient norm becomes small:
$$\|\nabla J(\theta_k)\| < \varepsilon,$$
- a maximum number of mini-batch iterations is reached.

Because each mini-batch only uses $m < n$ samples, we must run several epochs. The random reshuffling of the data at each epoch ensures that every example contributes to the learning process.

4 4. The Learning Rate η

Definition

The quantity $\eta > 0$ is the **learning rate**. It controls how big the update step is:
$$\theta_{k+1} = \theta_k - \eta\, g_k.$$

Role of η

- If η is too large: updates diverge or oscillate.
- If η is too small: convergence becomes extremely slow.
- Correct choice of η balances speed and stability.

Interpretation

$$\eta \text{ is the "speed" of learning.}$$

A large η learns fast but may lose control. A small η learns safely but slowly.

Typical Choices

- Batch GD: constant η.
- SGD: decreasing schedule $\eta_k \to 0$.
- Mini-batch: constant η (standard in deep learning).

Summary Box

$$\boxed{\text{Batch uses all data. SGD uses 1 sample. Mini-batch uses } m \text{ samples.}}$$

$$\boxed{\eta \text{ controls how big the step is: small } \eta = \text{slow, big } \eta = \text{unstable.}}$$