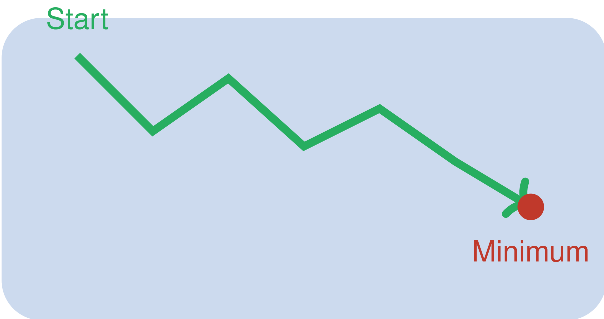
Deep Learning for Perception

Lecture 03: Stochastic Gradient Descent



Incremental Learning

Topics Covered in This Lecture:

- Review: Batch Gradient Descent
- Stochastic Gradient Descent (SGD)
- SGD Algorithm Step-by-Step
- SGD vs Batch GD Comparison
- Advantages of SGD
- Challenges & Solutions
- Mini-Batch Gradient Descent
- Multiple Numerical Examples

Instructor: Aqsa Younas

Department of Computer Science
FAST-NU, CFD Campus

Contents

Advance Organizer — What You'll Learn

Learning Objectives: By the end of this lecture, you will be able to:

1. **Explain** the motivation behind Stochastic Gradient Descent
2. **Implement** SGD weight updates step-by-step
3. **Compare** SGD with Batch Gradient Descent mathematically
4. **Calculate** weight updates for multiple epochs
5. **Understand** Mini-Batch GD as a compromise
6. **Solve** numerical problems involving SGD from scratch

Prior Knowledge Required:

- Batch Gradient Descent algorithm
- Weight update rule: $\Delta w_i = \eta(t - o)x_i$
- Loss functions (MSE)

1 Review: Batch Gradient Descent

Why It Matters

Before understanding SGD, we must clearly understand **Batch Gradient Descent** and its limitations. SGD was developed to address specific problems with Batch GD.

1.1 Batch GD Recap

Definition

Batch Gradient Descent computes the gradient using **ALL** training samples before making a single weight update.
Process:

1. Process **every sample** in the dataset
2. **Accumulate** all weight changes (Δw)
3. Update weights **ONCE** at the end

Key Formula

Batch GD Update Rule:

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) \cdot x_{id}$$

$$w_i \leftarrow w_i + \Delta w_i \quad \text{(update ONCE per epoch)}$$

1.2 Problems with Batch GD

Limitations of Batch Gradient Descent

1. **Computationally Expensive:** Must process ALL samples before one update
2. **Slow Convergence:** Only one weight update per epoch
3. **Memory Intensive:** Must store gradients for all samples
4. **Stuck in Local Minima:** Smooth path can get trapped
5. **Not Suitable for Large Datasets:** Impractical for millions of samples

2 Stochastic Gradient Descent (SGD)

Why It Matters

SGD was developed to make learning **faster** and more **practical** for large datasets. Instead of waiting to see all data, we update weights **immediately** after each sample!

2.1 Core Idea of SGD

Definition

Stochastic Gradient Descent (SGD) updates weights **incrementally** after processing **each individual training example**, rather than waiting for the entire dataset.
Key Insight: Each sample provides an **estimate** of the true gradient. We use this noisy estimate to make progress immediately.

Analogy — Think of It Like This

Batch GD vs SGD — A Navigation Analogy:
Batch GD: Like planning a road trip by studying the **entire map** before taking your first step. You know the exact direction, but it takes forever to start moving.
SGD: Like asking **one local person** at a time for directions. Each direction might be slightly off, but you start moving immediately and eventually reach your destination through many small corrections.

2.2 SGD Formula

Key Formula

SGD Update Rule (for each sample d):

$$\boxed{\Delta w_i = \eta(t_d - o_d) \cdot x_{id}}$$

$$\boxed{w_i \leftarrow w_i + \Delta w_i \quad \text{(update IMMEDIATELY)}}$$

Where:

- η = Learning rate
- t_d = Target value for sample d
- o_d = Output (prediction) for sample d
- x_{id} = Input feature i of sample d
- $(t_d - o_d)$ = Error for sample d

Critical Difference: Weights are updated **after EACH sample**, not after all samples!

2.3 SGD Error Function

Definition

In SGD, we define a **per-sample error function**:

$$E_d(\mathbf{w}) = \frac{1}{2}(t_d - o_d)^2$$

Instead of minimizing total error $E(\mathbf{w}) = \sum_d E_d(\mathbf{w})$, SGD takes steps based on individual $E_d(\mathbf{w})$ values.

3 SGD Algorithm: Step-by-Step

Stochastic Gradient Descent Algorithm

Input: Training data D, learning rate η, initial weights $\mathbf{w}$
Repeat (for each epoch):

1. **Shuffle** the training data (optional but recommended)
2. **For each** training sample (x, t) in D:
   a. Calculate output: $o = \sum_i w_i \cdot x_i$
   b. Calculate error: $e = t - o$
   c. **For each** weight w_i:
      - Calculate delta: $\Delta w_i = \eta \cdot e \cdot x_i$
      - **Update immediately:** $w_i \leftarrow w_i + \Delta w_i$

Until convergence (error below threshold or max epochs reached)

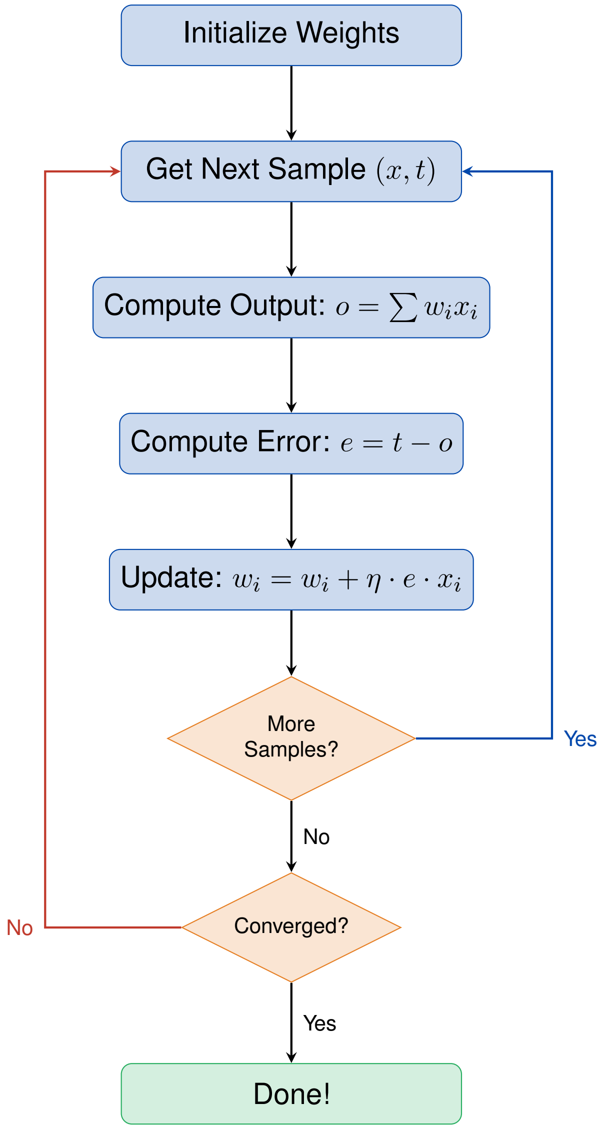


Figure 1: SGD Algorithm Flowchart: Update weights after EACH sample

Memory Hook — Remember This!

The Key Difference to Remember:

Batch GD	**SGD**
See ALL samples $\rightarrow$ Update ONCE	See ONE sample $\rightarrow$ Update ONCE
n samples = 1 update	n samples = n updates

4 Numerical Example 1: Basic SGD

Solved Example 1: SGD — One Epoch

Given Data:

- Learning rate: $\eta = 0.5$
- Initial weights: $w_0 = -0.3$ (bias), $w_1 = 0.5$, $w_2 = 0.5$
- Input format: $\mathbf{x} = [1, x_1, x_2]$ (first element is 1 for bias)

Training Data:

Sample	Input $\mathbf{x}$	Target t
1	[1, 1, 0]	1
2	[1, 1, 1]	0
3	[1, 0, 0]	0
4	[1, 0, 1]	1

Task: Perform ONE epoch of SGD. Show all weight updates.

Solution:
Output formula: $o = w_0 \cdot x_0 + w_1 \cdot x_1 + w_2 \cdot x_2$
Update rule: $\Delta w_i = \eta \cdot (t - o) \cdot x_i$ (apply immediately!)

STEP 1: Process Sample 1 — $\mathbf{x} = [1, 1, 0], t = 1$
Current weights: $[w_0, w_1, w_2] = [-0.3, 0.5, 0.5]$
Forward Pass:

$$o = (-0.3)(1) + (0.5)(1) + (0.5)(0) = -0.3 + 0.5 + 0 = \mathbf{0.2}$$

Error:

$$e = t - o = 1 - 0.2 = \mathbf{0.8}$$

Weight Updates:

$$\begin{aligned} \Delta w_0 &= 0.5 \times 0.8 \times 1 = \mathbf{0.4} & w_0 &= -0.3 + 0.4 = \mathbf{0.1} \\ \Delta w_1 &= 0.5 \times 0.8 \times 1 = \mathbf{0.4} & w_1 &= 0.5 + 0.4 = \mathbf{0.9} \\ \Delta w_2 &= 0.5 \times 0.8 \times 0 = \mathbf{0} & w_2 &= 0.5 + 0 = \mathbf{0.5} \end{aligned}$$

$\boxed{\text{Updated weights: } [0.1, 0.9, 0.5]}$

STEP 2: Process Sample 2 — $\mathbf{x} = [1, 1, 1], t = 0$
Current weights: $[0.1, 0.9, 0.5]$ (using UPDATED weights!)
Forward Pass:

$$o = (0.1)(1) + (0.9)(1) + (0.5)(1) = 0.1 + 0.9 + 0.5 = \mathbf{1.5}$$

Error:

$$e = t - o = 0 - 1.5 = \mathbf{-1.5}$$

Weight Updates:

$$\begin{aligned} \Delta w_0 &= 0.5 \times (-1.5) \times 1 = \mathbf{-0.75} & w_0 &= 0.1 - 0.75 = \mathbf{-0.65} \\ \Delta w_1 &= 0.5 \times (-1.5) \times 1 = \mathbf{-0.75} & w_1 &= 0.9 - 0.75 = \mathbf{0.15} \\ \Delta w_2 &= 0.5 \times (-1.5) \times 1 = \mathbf{-0.75} & w_2 &= 0.5 - 0.75 = \mathbf{-0.25} \end{aligned}$$

$\boxed{\text{Updated weights: } [-0.65, 0.15, -0.25]}$

STEP 3: Process Sample 3 — $\mathbf{x} = [1, 0, 0], t = 0$
Current weights: $[-0.65, 0.15, -0.25]$
Forward Pass:

$$o = (-0.65)(1) + (0.15)(0) + (-0.25)(0) = \mathbf{-0.65}$$

Error:

$$e = t - o = 0 - (-0.65) = \mathbf{0.65}$$

Weight Updates:

$$\begin{aligned} \Delta w_0 &= 0.5 \times 0.65 \times 1 = \mathbf{0.325} & w_0 &= -0.65 + 0.325 = \mathbf{-0.325} \\ \Delta w_1 &= 0.5 \times 0.65 \times 0 = \mathbf{0} & w_1 &= 0.15 + 0 = \mathbf{0.15} \\ \Delta w_2 &= 0.5 \times 0.65 \times 0 = \mathbf{0} & w_2 &= -0.25 + 0 = \mathbf{-0.25} \end{aligned}$$

$\boxed{\text{Updated weights: } [-0.325, 0.15, -0.25]}$

Final Answer (After 1 Epoch of SGD):

$$\boxed{w_0 = 0.4625, \quad w_1 = 0.15, \quad w_2 = 0.5375}$$

Key Observation: Notice how each step used the **updated weights** from the previous step. This is what makes SGD different from Batch GD!

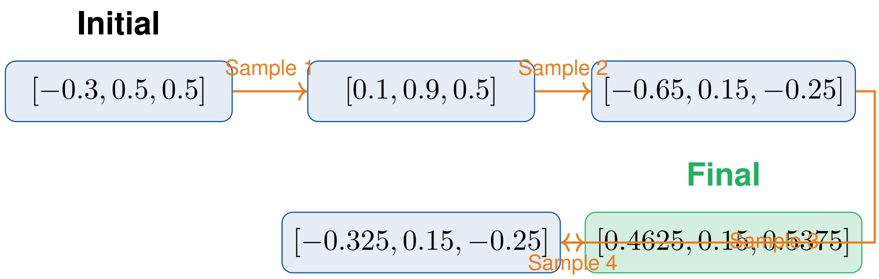


Figure 2: Weight evolution through SGD: 4 updates in 1 epoch

5 SGD vs Batch Gradient Descent

Why It Matters

Understanding the differences between SGD and Batch GD helps you choose the right algorithm for your problem and understand why modern deep learning prefers SGD-based methods.

5.1 Visual Comparison of Convergence Paths

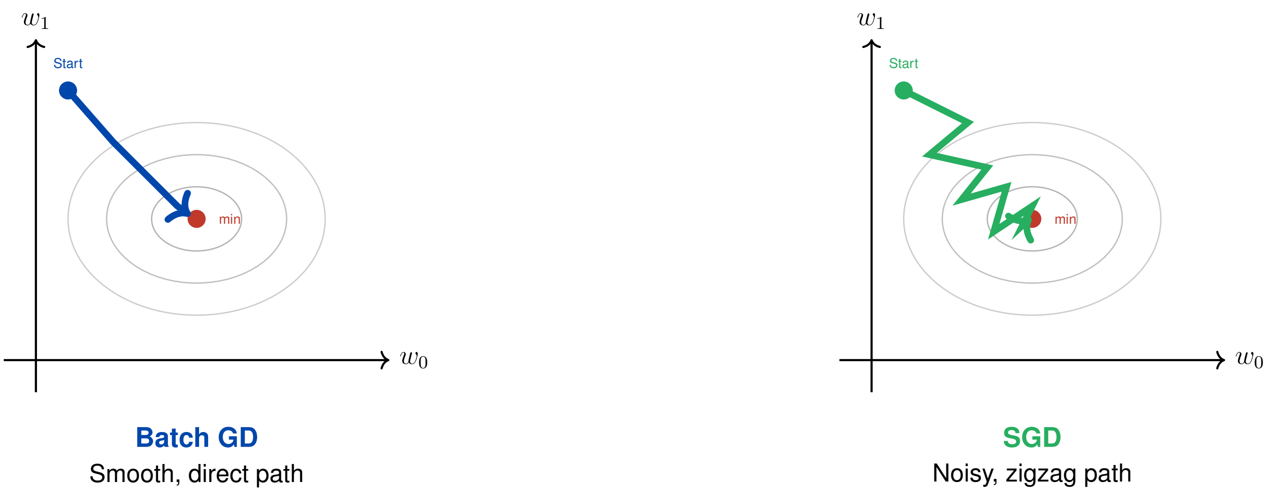


Figure 3: Convergence paths: Batch GD (smooth) vs SGD (noisy but often faster)

5.2 Detailed Comparison Table

Aspect	Batch GD	Stochastic GD
Update Frequency	Once per epoch	After every sample
Updates per Epoch	1 update	n updates (where n = samples)
Gradient Used	True gradient (exact)	Noisy estimate (approximate)
Convergence Path	Smooth, direct	Noisy, oscillating
Speed (large data)	Very slow	Much faster
Memory Usage	High (store all gradients)	Low (one sample at a time)
Local Minima	Can get stuck	Can escape (noise helps!)
Learning Rate	Can be larger	Usually needs smaller
Best For	Small datasets, convex problems	Large datasets, neural networks

Table 1: Comprehensive comparison of Batch GD vs SGD

5.3 Advantages of SGD

Why SGD is Preferred in Deep Learning

1. **Faster Progress:** n weight updates per epoch instead of 1
2. **Memory Efficient:** Only need to store one sample at a time
3. **Escapes Local Minima:** Noise in gradient can jump out of shallow valleys
4. **Online Learning:** Can learn from streaming data in real-time
5. **Regularization Effect:** Noise acts as implicit regularization

6 Numerical Example 2: SGD vs Batch GD Comparison

Solved Example 2: Comparing SGD and Batch GD Results

Given Data (same for both):

- Learning rate: $\eta = 0.5$
- Initial weights: $w_0 = -0.3$, $w_1 = 0.5$, $w_2 = 0.5$
- Same 4 training samples as Example 1

Task: Compare final weights after 1 epoch using both methods.

Method 1: Batch GD (accumulate all, update once)
Process all samples with **original weights** $[-0.3, 0.5, 0.5]$:

Sample	x	t	o	error	Δw_0	Δw_1, Δw_2
1	[1,1,0]	1	0.2	0.8	0.4	0.4, 0
2	[1,1,1]	0	0.7	-0.7	-0.35	-0.35, -0.35
3	[1,0,0]	0	-0.3	0.3	0.15	0, 0
4	[1,0,1]	1	0.2	0.8	0.4	0, 0.4
			Sum of Δw:		**0.6**	**0.05, 0.05**

Batch GD Final Weights:

$$w_0 = -0.3 + 0.6 = \mathbf{0.3}$$
$$w_1 = 0.5 + 0.05 = \mathbf{0.55}$$
$$w_2 = 0.5 + 0.05 = \mathbf{0.55}$$

Method 2: SGD (from Example 1)
Weights evolve: $[-0.3, 0.5, 0.5] \rightarrow [0.1, 0.9, 0.5] \rightarrow [-0.65, 0.15, -0.25] \rightarrow [-0.325, 0.15, -0.25] \rightarrow [0.4625, 0.15, 0.5375]$
SGD Final Weights: $w_0 = 0.4625$, $w_1 = 0.15$, $w_2 = 0.5375$

Calculate MSE for Both:
Batch GD with weights $[0.3, 0.55, 0.55]$:

Sample 1: $o = 0.3 + 0.55 + 0 = 0.85, \quad e^2 = (1 - 0.85)^2 = 0.0225$
Sample 2: $o = 0.3 + 0.55 + 0.55 = 1.4, \quad e^2 = (0 - 1.4)^2 = 1.96$
Sample 3: $o = 0.3 + 0 + 0 = 0.3, \quad e^2 = (0 - 0.3)^2 = 0.09$
Sample 4: $o = 0.3 + 0 + 0.55 = 0.85, \quad e^2 = (1 - 0.85)^2 = 0.0225$

$$\text{MSE} = \frac{0.0225 + 1.96 + 0.09 + 0.0225}{4} = \mathbf{0.5238}$$

SGD with weights $[0.4625, 0.15, 0.5375]$:

Sample 1: $o = 0.4625 + 0.15 + 0 = 0.6125, \quad e^2 = (1 - 0.6125)^2 = 0.1502$
Sample 2: $o = 0.4625 + 0.15 + 0.5375 = 1.15, \quad e^2 = (0 - 1.15)^2 = 1.3225$
Sample 3: $o = 0.4625, \quad e^2 = (0 - 0.4625)^2 = 0.2139$
Sample 4: $o = 0.4625 + 0 + 0.5375 = 1.0, \quad e^2 = (1 - 1.0)^2 = 0$

$$\text{MSE} = \frac{0.1502 + 1.3225 + 0.2139 + 0}{4} = \mathbf{0.4216}$$

Comparison Results:

Method	w_0	w_1	w_2	**MSE**
Batch GD	0.3	0.55	0.55	0.5238
SGD	0.4625	0.15	0.5375	**0.4216**

Observation: After just 1 epoch, SGD achieved **lower MSE** (0.4216 vs 0.5238). This is because SGD made 4 weight updates while Batch GD made only 1!

7 SGD Over Multiple Epochs

Why It Matters

In practice, SGD runs for **multiple epochs**. The noisy updates gradually lead to convergence. Let's see how weights evolve over several passes through the data.

Solved Example 3: SGD for 3 Epochs

Given Data: Same as Examples 1 and 2.
Task: Perform 3 complete epochs of SGD and track weight evolution.

Solution:
We continue from the same initial weights and show the summary for each epoch.
EPOCH 1: (Already computed in Example 1)

Step	Input	t	Error	Weights After
1	[1,1,0]	1	0.8	[0.1, 0.9, 0.5]
2	[1,1,1]	0	-1.5	[-0.65, 0.15, -0.25]
3	[1,0,0]	0	0.65	[-0.325, 0.15, -0.25]
4	[1,0,1]	1	1.575	[0.4625, 0.15, 0.5375]

End of Epoch 1: $[0.4625, 0.15, 0.5375]$

EPOCH 2: Starting with $[0.4625, 0.15, 0.5375]$
Step 1: $x = [1, 1, 0], t = 1$

- $o = 0.4625 + 0.15 + 0 = 0.6125$
- Error $= 1 - 0.6125 = 0.3875$
- $\Delta w = [0.1938, 0.1938, 0]$
- Updated: $[0.6563, 0.3438, 0.5375]$

Step 2: $x = [1, 1, 1], t = 0$

- $o = 0.6563 + 0.3438 + 0.5375 = 1.5375$
- Error $= 0 - 1.5375 = -1.5375$
- Updated: $[-0.1125, -0.4250, -0.2313]$

Step 3: $x = [1, 0, 0], t = 0$

- $o = -0.1125$, Error $= 0.1125$
- Updated: $[-0.0563, -0.4250, -0.2313]$

Step 4: $x = [1, 0, 1], t = 1$

- $o = -0.0563 + 0 - 0.2313 = -0.2875$
- Error $= 1 - (-0.2875) = 1.2875$
- Updated: $[0.5875, -0.4250, 0.4125]$

End of Epoch 2: $[0.5875, -0.4250, 0.4125]$

EPOCH 3: Starting with $[0.5875, -0.4250, 0.4125]$
(Following same process...)
End of Epoch 3: $[0.7219, -0.7125, 0.2781]$

Weight Evolution Summary:

Epoch	w_0	w_1	w_2
0 (Initial)	-0.3	0.5	0.5
1	0.4625	0.15	0.5375
2	0.5875	-0.425	0.4125
3	0.7219	-0.7125	0.2781

Observation: Weights continue to change significantly even after multiple epochs. SGD's noisy path gradually refines the solution.

8 Mini-Batch Gradient Descent

Why It Matters

Mini-Batch GD is a **compromise** between Batch GD and SGD. It offers the benefits of both: stable gradient estimates like Batch GD and frequent updates like SGD.

8.1 Mini-Batch GD Concept

Definition

Mini-Batch Gradient Descent divides the training data into small **batches** of size B. Weights are updated after processing each batch.

$$\Delta w_i = \eta \sum_{d \in \text{batch}} (t_d - o_d) \cdot x_{id}$$

Common batch sizes: 16, 32, 64, 128, 256

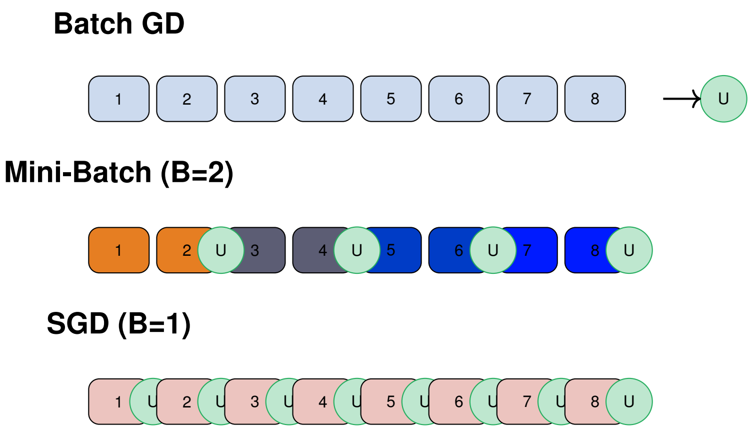


Figure 4: Comparison: Batch GD (1 update), Mini-Batch (4 updates), SGD (8 updates)

8.2 Mini-Batch GD Advantages

Why Mini-Batch is Often Best

- **Better gradient estimate** than SGD (averages over batch)
- **More updates** than Batch GD (faster progress)
- **GPU-friendly:** Batches can be processed in parallel
- **Memory efficient:** Only load batch into memory
- **Sweet spot:** Balances noise and stability

Solved Example 4: Mini-Batch GD (Batch Size = 2)

Given Data:

- Learning rate: $\eta = 0.5$, Batch size: $B = 2$
- Initial weights: $w_0 = -0.3$, $w_1 = 0.5$, $w_2 = 0.5$
- Same 4 training samples

Task: Perform 1 epoch of Mini-Batch GD.

Solution:
Divide 4 samples into 2 mini-batches of size 2.
Mini-Batch 1: Samples 1 and 2
Using weights $[-0.3, 0.5, 0.5]$:
Sample 1: $x = [1, 1, 0], t = 1$

- $o = -0.3 + 0.5 + 0 = 0.2$, Error $= 0.8$
- $\Delta w = [0.4, 0.4, 0]$

Sample 2: $x = [1, 1, 1], t = 0$

- $o = -0.3 + 0.5 + 0.5 = 0.7$, Error $= -0.7$
- $\Delta w = [-0.35, -0.35, -0.35]$

Accumulated Δw**:** $[0.4 - 0.35, 0.4 - 0.35, 0 - 0.35] = [0.05, 0.05, -0.35]$
Update: $[-0.3 + 0.05, 0.5 + 0.05, 0.5 - 0.35] = \boxed{[-0.25, 0.55, 0.15]}$

Mini-Batch 2: Samples 3 and 4
Using weights $[-0.25, 0.55, 0.15]$:
Sample 3: $x = [1, 0, 0], t = 0$

- $o = -0.25$, Error $= 0.25$
- $\Delta w = [0.125, 0, 0]$

Sample 4: $x = [1, 0, 1], t = 1$

- $o = -0.25 + 0 + 0.15 = -0.1$, Error $= 1.1$
- $\Delta w = [0.55, 0, 0.55]$

Accumulated Δw**:** $[0.125 + 0.55, 0, 0 + 0.55] = [0.675, 0, 0.55]$
Update: $[-0.25 + 0.675, 0.55 + 0, 0.15 + 0.55] = \boxed{[0.425, 0.55, 0.7]}$

Final Answer (Mini-Batch GD, B=2):

$$\boxed{w_0 = 0.425, \quad w_1 = 0.55, \quad w_2 = 0.7}$$

Comparison (all after 1 epoch):

Method	w_0	w_1	w_2	**Updates**
Batch GD	0.3	0.55	0.55	1
Mini-Batch (B=2)	0.425	0.55	0.7	2
SGD	0.4625	0.15	0.5375	4

9 Practical Considerations for SGD

9.1 Learning Rate Selection

Memory Hook — Remember This!

Learning Rate Guidelines:

- **Too large:** Overshoots minimum, may diverge
- **Too small:** Converges very slowly
- **Typical values:** 0.001 to 0.1
- **Start large, decay over time** (learning rate scheduling)

Figure 5: Effect of learning rate on SGD convergence

9.2 When SGD Can Escape Local Minima

Definition

Local Minima Escape:
Because SGD uses different gradients $\nabla E_d(\mathbf{w})$ for each sample (instead of the true gradient $\nabla E(\mathbf{w})$), the noisy updates can "jump" out of shallow local minima.
This is particularly useful in deep learning where the error surface has many local minima.

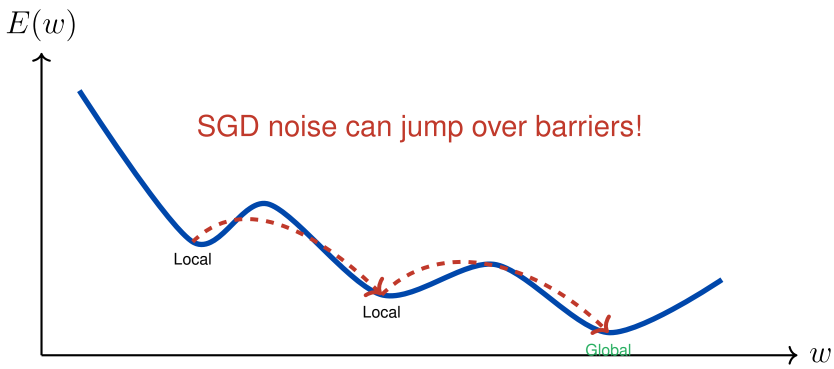


Figure 6: SGD's noisy updates can escape local minima

10 Summary: SGD Calculation Checklist

SGD Step-by-Step Checklist

For EACH sample in the dataset:

1. **Forward Pass:** Calculate output

$$o = \sum_{i=0}^{n} w_i \cdot x_i = w_0x_0 + w_1x_1 + \ldots + w_nx_n$$

2. **Error Calculation:**

$$e = t - o$$

3. **Weight Updates:** For each weight w_i:

$$\Delta w_i = \eta \cdot e \cdot x_i$$

$$w_i \leftarrow w_i + \Delta w_i \quad \text{(UPDATE IMMEDIATELY!)}$$

4. **Move to next sample** with the **updated weights**

Repeat for multiple epochs until convergence.

Self-Test — Check Your Understanding

Quick Practice:
Given: $\eta = 0.2$, weights $[w_0, w_1] = [0.5, -0.3]$, sample $x = [1, 2]$, target $t = 1$

1. Calculate output o
2. Calculate error e
3. Calculate Δw_0 and Δw_1
4. What are the updated weights?

Answers:

1. $o = 0.5(1) + (-0.3)(2) = 0.5 - 0.6 = -0.1$
2. $e = 1 - (-0.1) = 1.1$
3. $\Delta w_0 = 0.2 \times 1.1 \times 1 = 0.22$; $\Delta w_1 = 0.2 \times 1.1 \times 2 = 0.44$
4. $[0.5 + 0.22, -0.3 + 0.44] = [0.72, 0.14]$

11 Glossary

Term	Definition
SGD	Stochastic Gradient Descent — updates weights after each sample
Batch GD	Updates weights once after processing all samples
Mini-Batch GD	Updates weights after processing a small batch of samples
Epoch	One complete pass through all training data
Learning Rate (η)	Controls the step size of weight updates
Gradient	Direction of steepest increase in error
Local Minimum	A point where error is lower than nearby points but not globally lowest
Convergence	When weights stop changing significantly
Batch Size	Number of samples processed before one weight update