Neural Network Training on MNIST

Parth Bansal

June 19, 2025

Abstract

We present a three-layer neural network trained on the MNIST dataset of handwritten digits. To accelerate pure-NumPy training, we switch from a full-batch loop to an *epoch + mini-batch* scheme, significantly reducing per-step cost and improving convergence control.

1 Algorithm Overview

Training Algorithm

Input: $X \in \mathbb{R}^{n_x \times m}$, $Y \in \{0,1\}^{n_y \times m}$, layer sizes $(n_x, n_{h1}, n_{h2}, n_y)$, epochs E, batch size B, learning rate α

Output: Parameters $\{W^{[l]}, b^{[l]}\}_{l=1}^{3}$

Initialize (He): $W^{[l]} \sim \mathcal{N}(0, \frac{2}{n_{l-1}})$, $b^{[l]} \leftarrow 0$
for epoch $= 1$ **to** E **do**
  Shuffle columns of (X, Y)
  for each mini-batch (X_b, Y_b) of size B **do**
    Forward:

$$Z^{[1]} = W^{[1]} X_b + b^{[1]},\ A^{[1]} = \mathrm{ReLU}(Z^{[1]})$$

$$Z^{[2]} = W^{[2]} A^{[1]} + b^{[2]},\ A^{[2]} = \mathrm{ReLU}(Z^{[2]})$$

$$Z^{[3]} = W^{[3]} A^{[2]} + b^{[3]},\ A^{[3]} = \mathrm{softmax}(Z^{[3]})$$

    Loss: $\mathcal{L} = -\frac{1}{B} \sum_{i=1}^{B} \sum_{j=1}^{n_y} Y_{j,i} \log A^{[3]}_{j,i}$
    Backward: compute $dW^{[l]}, db^{[l]}$ via standard backprop
    Update: $W^{[l]} -= \alpha\, dW^{[l]},\ b^{[l]} -= \alpha\, db^{[l]}$
  end for
end for

Output: return $\{W^{[l]}, b^{[l]}\}_{l=1}^{3}$

2 Batching Comparison

Full-Batch vs. Epoch+Mini-Batch

	Full-Batch	**Epoch+Mini-Batch**
Data per update	60 000	64
Updates per pass	1	$\approx 60000/64 \approx 938$
Total per epoch	60 000	60 000
Speed per update	Very slow	Fast
Control	Low	High

3 Key Takeaways

- **Mini-batches** dramatically reduce per-step computation and add beneficial gradient noise.
- **Epochs** let you monitor progress over full passes and stop when performance plateaus.
- **He initialization** and numerically-stable softmax are essential for stable training.