Ashutosh Kumar Jha 12340390 Homework 2

Ashutosh Kumar Jha

February 2025

1 Question 1

The colab notebook for the solution is attached below: Colab Notebook

1.1 Q1. How does creating batches impact the training process?

1.1.1 Computational Efficiency

- Processing data in batches (instead of one sample at a time) makes efficient use of hardware resources, especially GPUs and TPUs, as they can process matrix operations in parallel.
- Large batches utilize vectorized operations better, leading to faster training.

of Training

- Large batches provide more stable gradient estimates, leading to smoother convergence.
- Small batches introduce more noise into gradient updates, which can help escape local minima but might slow convergence.

1.1.2 Memory Constraints

- Training on entire datasets at once is infeasible for large datasets due to memory limitations. Batching allows models to be trained on limited GPU/CPU memory.

1.1.3 Convergence Speed

- Smaller batches introduce more variance in gradient updates, which can lead to faster generalization but slower overall convergence.
- Larger batches have more accurate gradient estimates but may result in models stuck in sharp local minima.

1.1.4 Generalization

- Very small batches may cause excessive noise, preventing good generalization.
- Very large batches might overfit to training data and fail to generalize to unseen data.

1.1.5 Types of Batch Processing

- **Mini-batch gradient descent** (most common): Uses a subset of data to compute gradients.
- **Batch gradient descent**: Uses the entire dataset to compute gradients (slow but accurate).
- **Stochastic gradient descent (SGD)**: Uses one data point at a time, introducing high variance in updates.

1.1.6 Choosing the Right Batch Size

- **Small batches** (e.g., 8-32): Better generalization but noisier updates.
- **Medium batches** (e.g., 64-256): Balance between stability and speed.
- **Large batches** (e.g., 512-4096+): Faster convergence but may need careful tuning (e.g., learning rate scaling).

1.2 Q2. How does the model generalize when we use different sized batches?

1.2.1 Small Batches (e.g., 8 - 64)

Pros: Better Generalization
Cons: Slower Convergence & Noisier Updates

- **More Noise in Gradients:** Small batches introduce stochasticity (randomness) in gradient updates, preventing the model from settling into sharp, narrow minima (which often lead to overfitting).
- **Better Exploration:** The noise in gradients allows the optimizer to escape sharp local minima and find flatter minima, which are known to generalize better.
- **Regularization Effect:** Acts like an implicit form of regularization (similar to dropout or weight decay), improving the model's ability to adapt to new data.

Best for: Complex datasets where generalization is more important than speed.

1.2.2 Medium Batches (e.g., 128 - 512)

Pros: Balanced Generalization & Speed
Cons: Still Some Noise, But More Stability

- **Compromise Between Stability & Exploration:** Reduces noise compared to small batches, but still retains some randomness for better generalization.
- **Faster Convergence:** More accurate gradient estimates lead to quicker learning without excessive computational cost.
- **Good for Many Applications:** Works well for most deep learning tasks, including computer vision and NLP.

Best for: Many practical applications where training time and generalization must be balanced.

1.2.3 Large Batches (e.g., 1024 - 4096+)

Pros: Faster Training & More Stable Gradients
Cons: Poor Generalization Without Adjustments

- **Less Noise, More Precise Updates:** Large batches have smoother gradient estimates, making convergence more stable.
- **Risk of Overfitting:** Can lead to sharp local minima that do not generalize well to unseen data.
- **Difficulty Escaping Bad Minima:** The optimizer may get trapped in sharp, high-curvature regions of the loss landscape.
- **Requires Learning Rate Scaling:** To maintain effective optimization, the learning rate must often be increased.

Best for: When training speed is critical, but requires additional techniques to improve generalization.

1.2.4 Key Takeaways

Batch Size	Noise in Gradients	Convergence Speed	Generalization
Small (8-64)	High (noisy)	Slow	Best (flat minima)
Medium (128-512)	Moderate	Balanced	Good
Large (1024-4096+)	Low (smooth)	Fast	Poor (sharp minima)

1.2.5 Improving Generalization for Large Batches

- **Learning Rate Warmup** – Start with a small learning rate and gradually increase it.
- **Batch Normalization** – Helps mitigate issues related to sharp minima.
- **Stochastic Weight Averaging (SWA)** – Averages multiple models trained with different batch updates to improve generalization.
- **Data Augmentation & Regularization** – Techniques like dropout, weight decay, and mixup help prevent overfitting.
- **Sharpness-Aware Minimization (SAM)** – A technique that encourages flatter minima even with large batch sizes.

1.3 Q3. Does Epochs and Learning Rate share any relation. Justify

1.3.1 Inverse Relationship

- The number of epochs required for convergence decreases as the learning rate increases.
- A higher learning rate updates weights more aggressively, leading to faster convergence.

1.3.2 Trade-Off Between Stability and Speed

- If the learning rate is too small, the model converges slowly, requiring more epochs.
- If the learning rate is too high, the model may overshoot the optimal value or diverge.

1.3.3 Optimal Learning Rate Selection

- A moderate learning rate ensures efficient convergence in fewer epochs while maintaining stability.
- Adaptive learning rate strategies (e.g., decay or scheduling) help balance convergence speed and stability.

1.3.4 Empirical Justification

- In gradient descent plots, a low learning rate shows a gradual and smooth decline in loss.
- A high learning rate can show oscillations or even divergence instead of convergence.

2 Question 2

2.1 a

Given Functions:

$$f_1(x) = \sin(x_1)\cos(x_2), \quad x \in \mathbb{R}^2$$
$$f_2(x, y) = x^\top y, \quad x, y \in \mathbb{R}^n$$
$$f_3(x) = xx^\top, \quad x \in \mathbb{R}^n$$

(a) Dimensions of $\frac{\partial f}{\partial x}$:

- For $f_1(x)$:
$$\frac{\partial f_1}{\partial x} \in \mathbb{R}^{1\times 2}$$
because $f_1(x)$ is scalar and $x \in \mathbb{R}^2$.
- For $f_2(x, y)$:
$$\frac{\partial f_2}{\partial x} \in \mathbb{R}^{1\times n}, \quad \frac{\partial f_2}{\partial y} \in \mathbb{R}^{1\times n}$$
because $f_2(x, y)$ is scalar and $x, y \in \mathbb{R}^n$.
- For $f_3(x)$:
$$\frac{\partial f_3}{\partial x} \in \mathbb{R}^{n\times n\times n}$$
because $f_3(x) \in \mathbb{R}^{n\times n}$ and $x \in \mathbb{R}^n$.

(b) Jacobians:

1. For $f_1(x)$:

$$\frac{\partial f_1}{\partial x} = \begin{bmatrix} \cos(x_1)\cos(x_2) & -\sin(x_1)\sin(x_2) \end{bmatrix}$$

2. For $f_2(x, y)$:

$$\frac{\partial f_2}{\partial x} = y^\top, \quad \frac{\partial f_2}{\partial y} = x^\top$$

3. For $f_3(x)$:

$$\frac{\partial f_3}{\partial x} = \frac{\partial}{\partial x}(xx^\top) = \begin{bmatrix} \frac{\partial}{\partial x_1}(xx^\top) & \cdots & \frac{\partial}{\partial x_n}(xx^\top) \end{bmatrix}$$

Each term is given as:

$$\frac{\partial}{\partial x_i}(xx^\top) = \begin{bmatrix} \delta_{1i}x_1 + \delta_{i1}x_1 & \cdots & \delta_{1i}x_n + \delta_{i1}x_n \\ \vdots & \ddots & \vdots \\ \delta_{ni}x_1 + \delta_{in}x_1 & \cdots & \delta_{ni}x_n + \delta_{in}x_n \end{bmatrix}$$

where δ_{ij} is the Kronecker delta.

2.2 b

(a) Use the chain rule and provide the dimensions:

Given:

$$f(z) = \exp\left(-\frac{1}{2}z\right), \quad z = g(y) = y^\top S^{-1} y, \quad y = h(x) = x - \mu,$$

where $x, \mu \in \mathbb{R}^D$ and $S \in \mathbb{R}^{D \times D}$.

Step 1: Dimensions of $f(z)$:

$$f(z) : \mathbb{R} \to \mathbb{R}, \quad \frac{\partial f}{\partial z} \in \mathbb{R}.$$

Step 2: Dimensions of $z = y^\top S^{-1} y$:

$$y \in \mathbb{R}^D, \quad S^{-1} \in \mathbb{R}^{D \times D}, \quad z = y^\top S^{-1} y \in \mathbb{R}.$$

$$\frac{\partial z}{\partial y} = 2S^{-1}y, \quad \frac{\partial z}{\partial y} \in \mathbb{R}^D.$$

Step 3: Dimensions of $y = x - \mu$:

$$x, \mu \in \mathbb{R}^D, \quad y = x - \mu \in \mathbb{R}^D.$$

$$\frac{\partial y}{\partial x} = I, \quad \frac{\partial y}{\partial x} \in \mathbb{R}^{D \times D}.$$

Chain Rule: Using the chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial z} \cdot \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}.$$

Dimensions:

$$\frac{\partial f}{\partial x} \in \mathbb{R}^D.$$

—

(b) Compute the derivatives of $f(x) = \mathbf{tr}(xx^\top + \sigma^2 I)$:

Given:

$$f(x) = \mathrm{tr}(xx^\top + \sigma^2 I),$$

where $x \in \mathbb{R}^D$ and $I \in \mathbb{R}^{D \times D}$ is the identity matrix.

Step 1: Simplify $f(x)$:

$$f(x) = \mathrm{tr}(xx^\top) + \mathrm{tr}(\sigma^2 I).$$

$$\mathrm{tr}(xx^\top) = \sum_{i=1}^{D} x_i^2, \quad \mathrm{tr}(\sigma^2 I) = D \cdot \sigma^2.$$

$$f(x) = \sum_{i=1}^{D} x_i^2 + D \cdot \sigma^2.$$

Step 2: Derivative of $f(x)$: The derivative of $\mathrm{tr}(xx^\top)$ with respect to x is:

$$\frac{\partial \mathrm{tr}(xx^\top)}{\partial x} = 2x.$$

$$\frac{\partial f}{\partial x} = 2x.$$

—

(c) Use the chain rule for $f = \tanh(z)$:

Given:

$$f = \tanh(z), \quad z = Ax + b, \quad x \in \mathbb{R}^N,\ A \in \mathbb{R}^{M\times N},\ b \in \mathbb{R}^M.$$

Step 1: Dimensions:

$$f \in \mathbb{R}^M, \quad z \in \mathbb{R}^M, \quad x \in \mathbb{R}^N, \quad A \in \mathbb{R}^{M\times N}, \quad b \in \mathbb{R}^M.$$

Step 2: Derivatives:

- $\frac{\partial f}{\partial z} = \mathrm{diag}(1 - \tanh^2(z)) \in \mathbb{R}^{M\times M}$.
- $\frac{\partial z}{\partial x} = A \in \mathbb{R}^{M\times N}$.

Chain Rule: Using the chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial z} \cdot \frac{\partial z}{\partial x}.$$

Dimensions:

$$\frac{\partial f}{\partial x} \in \mathbb{R}^{M\times N}.$$

3 Question 3

The colab notebook for the solution of the question is attached below: Colab Notebook