Question: A team has a dataset that contains 1000 samples for training a feed-forward neural network. Suppose they decided to use the stochastic gradient descent algorithm to update the weights. How many times do the weights get updated after training the network for 5 epochs?

- 1000
- 5000
- 100
- 5

Answer:
5000

Explanation:

In stochastic gradient descent (SGD), the weights are updated after each sample (or data point) is processed. Given:

- **1000 samples** in the dataset
- **5 epochs** of training

The total number of weight updates is calculated as:

$$1000 \text{ samples} \times 5 \text{ epochs} = 5000 \text{ updates}$$

1. Mini-Batch Gradient Descent

In mini-batch gradient descent, the dataset is divided into smaller batches (mini-batches). Each batch is used to update the weights once per epoch. Suppose the mini-batch size is M.

Formula:

$$\text{Total Updates} = \left(\frac{\text{Number of Samples}}{\text{Mini-Batch Size}} \right) \times \text{Number of Epochs}$$

Example:

Assume:

- **1000 samples** in the dataset
- **Mini-batch size** of 100
- **5 epochs**

Calculation:

1. Calculate the number of mini-batches per epoch:
$$\frac{1000}{100} = 10 \text{ mini-batches}$$
2. Multiply by the number of epochs:
$$10 \text{ mini-batches per epoch} \times 5 \text{ epochs} = 50 \text{ updates}$$

Total Updates for Mini-Batch Gradient Descent: 50

2. Batch Gradient Descent

In batch gradient descent, the entire dataset is used to compute the gradients and update the weights only once per epoch.

Formula:

$$\text{Total Updates} = \text{Number of Epochs}$$

Example:

Assume:

- **1000 samples** in the dataset
- **5 epochs**

Calculation:

Since batch gradient descent processes the entire dataset once per epoch, we only update the weights once per epoch.

1. **5 epochs** results in **5 weight updates**.

Total Updates for Batch Gradient Descent: 5

1. Momentum-Based Gradient Descent

Momentum-based gradient descent builds upon regular gradient descent by adding a "momentum" term, which helps smooth out the updates and accelerates convergence by carrying forward some of the previous update direction.

Formula:

$$\text{Total Updates} = \text{Number of Samples} \times \text{Number of Epochs}$$

- Momentum doesn't change the number of updates but modifies how updates are computed.

Example:

For **1000 samples** and **5 epochs**, the calculation is:

$$1000 \times 5 = 5000 \text{ updates}$$

2. Nesterov Accelerated Gradient (NAG)

Nesterov Accelerated Gradient (NAG) is similar to momentum but performs a "look-ahead" step before calculating the gradient. This look-ahead reduces the oscillations and helps converge faster.

Formula:

$$\text{Total Updates} = \text{Number of Samples} \times \text{Number of Epochs}$$

- Similar to momentum-based GD, NAG does not alter the update count, only how updates are calculated.

Example:

For **1000 samples** and **5 epochs**:

$$1000 \times 5 = 5000 \text{ updates}$$

3. Adagrad (Adaptive Gradient)

Adagrad adapts the learning rate for each parameter based on historical gradient information, using a smaller learning rate for parameters with large gradients. It does not change the update count, just the magnitude of each update.

Formula:

$$\text{Total Updates} = \text{Number of Samples} \times \text{Number of Epochs}$$

Example:

For **1000 samples** and **5 epochs**:

$$1000 \times 5 = 5000 \text{ updates}$$

4. Adam (Adaptive Moment Estimation)

Adam combines momentum and adaptive learning rates (like Adagrad) by keeping track of both the average of the gradients (momentum) and the average of the squared gradients. It doesn't alter the count of updates, only the adjustment per update.

Formula:

$$\text{Total Updates} = \text{Number of Samples} \times \text{Number of Epochs}$$

Example:

For **1000 samples** and **5 epochs**:

$$1000 \times 5 = 5000 \text{ updates}$$

Gradient Descent Type	Number of Updates Calculation	Example Updates (5 Epochs, 1000 Samples)
Stochastic Gradient Descent	$\text{Samples} \times \text{Epochs}$	5000
Mini-Batch Gradient Descent	$\left(\frac{\text{Samples}}{\text{Mini-Batch Size}}\right) \times \text{Epochs}$	50
Batch Gradient Descent	Epochs	5

Gradient Descent Method	Mechanism	Number of Updates Calculation	Example Updates (5 Epochs, 1000 Samples)
Momentum-Based GD	Adds momentum to updates	$\text{Samples} \times \text{Epochs}$	5000
NAG	Adds "look-ahead" to momentum updates	$\text{Samples} \times \text{Epochs}$	5000
Adagrad	Adapts learning rate based on gradients	$\text{Samples} \times \text{Epochs}$	5000
Adam	Combines momentum + adaptive learning	$\text{Samples} \times \text{Epochs}$	5000

A team has a dataset that contains 100 samples for training a feed-forward neural network. They decided to use the gradient descent algorithm to update the weights. Additionally, they use a line search algorithm for the learning rate, with options $\eta = [0.01, 0.1, 1, 2, 10]$. How many times do the weights get updated after training the network for 10 epochs? (Note, for each weight update, the loss has to decrease).

- 100
- 5
- 500
- 10
- 50

2. **Weight Updates in Gradient Descent:**
   - In standard gradient descent, weights are updated **once per epoch** based on the gradient computed from all samples.
   - Therefore, for your process that involves evaluating different learning rates, you still only have **1 update per epoch**.
3. **Learning Rates Trials:**
   - You are trying **5 different learning rates** for each epoch to find one that results in a decrease in loss.
   - Although you evaluate 5 learning rates, only one update is accepted per epoch if the loss decreases.
4. **Understanding Total Updates:**
   - Since you only apply one weight update per epoch, across **10 epochs**, the total number of weight updates remains straightforward:
     - **Total weight updates = 10 epochs × 1 update per epoch = 10 updates.**

3) Select the true statements about the factor β used in the momentum based gradient descent algorithm.

- [x] Setting $\beta = 0.1$ allows the algorithm to move faster than the vanilla gradient descent algorithm
- [x] Setting $\beta = 0$ makes it equivalent to the vanilla gradient descent algorithm
- [ ] Setting $\beta = 1$ makes it equivalent to the vanilla gradient descent algorithm
- [x] Oscillation around the minimum will be less if we set $\beta = 0.1$ than setting $\beta = 0.99$

Yes, the answer is correct.
Score: 1

Accepted Answers:
Setting $\beta = 0.1$ allows the algorithm to move faster than the vanilla gradient descent algorithm
Setting $\beta = 0$ makes it equivalent to the vanilla gradient descent algorithm
Oscillation around the minimum will be less if we set $\beta = 0.1$ than setting $\beta = 0.99$

4) What is the advantage of using mini-batch gradient descent over batch gradient descent?

- [x] Mini-batch gradient descent is more computationally efficient than batch gradient descent.
- [ ] Mini-batch gradient descent leads to a more accurate estimate of the gradient than batch gradient descent.
- [ ] Mini batch gradient descent gives us a better solution.
- [x] Mini-batch gradient descent can converge faster than batch gradient descent.

Yes, the answer is correct.
Score: 1

Accepted Answers:
Mini-batch gradient descent is more computationally efficient than batch gradient descent.
Mini-batch gradient descent can converge faster than batch gradient descent.

5) Given data where one column predominantly contains zero values, which algorithm should be used to achieve faster convergence and optimize the loss function?

- (•) Adam
- ( ) NAG
- ( ) Momentum-based gradient descent
- ( ) Stochastic gradient descent

Yes, the answer is correct.
Score: 1

Accepted Answers:
Adam

When dealing with data where one column predominantly contains zero values, the choice of an optimization algorithm can significantly affect the convergence speed and overall performance of the training process. Let's evaluate the options provided:

1. **Adam (Adaptive Moment Estimation)**:
   - **Advantage**: Adam is designed to handle sparse gradients well. It adapts the learning rate for each parameter based on the estimates of first and second moments of the gradients, which can be beneficial in cases like this where some features do not contribute much compared to others (like the predominantly zero feature).
   - **Conclusion**: Likely to achieve faster convergence.
2. **NAG (Nesterov Accelerated Gradient)**:
   - **Advantage**: NAG is an improved version of momentum-based gradient descent that allows the algorithm to look ahead in the direction of the momentum. It can help reduce oscillations and can converge faster than standard momentum.
   - **Conclusion**: Good option, but not specifically optimized for sparsity in data.
3. **Momentum-based gradient descent**:
   - **Advantage**: Momentum helps accelerate gradient descent in the relevant direction and can help escape local minima. However, it does not inherently adapt to the sparsity of the data.
   - **Conclusion**: Useful but may not be the most efficient for sparse data.
4. **Stochastic Gradient Descent (SGD)**:
   - **Advantage**: SGD updates the model more frequently (every training example) and can lead to faster convergence in terms of reaching a good solution. However, the randomness can lead to noise in the updates, and it may not handle the sparsity as effectively as adaptive methods.
   - **Conclusion**: Can converge faster but not necessarily the best for sparse features without additional adaptations.

6) Which parameter in vanilla gradient descent determines the step size taken in the direction of the gradient?

- (●) Learning rate
- ( ) Momentum
- ( ) Gamma
- ( ) None of the above

Yes, the answer is correct.
Score: 1
Accepted Answers:
Learning rate

The parameter in vanilla gradient descent that determines the step size taken in the direction of the gradient is the **Learning Rate**.

7) Which of the following algorithms will result in more oscillations of the parameter during the training process of the neural network?

- (•) Stochastic gradient descent
- ( ) Mini batch gradient descent
- ( ) Batch gradient descent
- ( ) Batch NAG

Yes, the answer is correct.
Score: 1
Accepted Answers:
Stochastic gradient descent

1. **Stochastic Gradient Descent (SGD)**:
   - **Description**: Updates parameters using one training sample at a time.
   - **Oscillation**: **Most oscillations** due to high noise in updates.
2. **Mini-batch Gradient Descent**:
   - **Description**: Uses a small batch of samples for updates.
   - **Oscillation**: **Moderate oscillations**, less than SGD.
3. **Batch Gradient Descent**:
   - **Description**: Updates using the entire dataset.
   - **Oscillation**: **Minimal or no oscillations**, smooth convergence.
4. **Batch NAG (Nesterov Accelerated Gradient)**:
   - **Description**: Enhances momentum by anticipating future gradients.
   - **Oscillation**: **Less oscillation** compared to SGD and mini-batch.

Summary:

Stochastic Gradient Descent results in the **most oscillations** during training.

8) Which of the following are among the disadvantages of Adagrad?

- ( ) It doesn't work well for the Sparse matrix.
- ( ) It usually goes past the minima.
- ( ) It gets stuck before reaching the minima.
- (•) Weight updates are very small at the initial stages of the algorithm.

No, the answer is incorrect.
Score: 0
Accepted Answers:
It gets stuck before reaching the minima.

Adagrad (Adaptive Gradient Algorithm) has specific disadvantages that can affect its performance. Here's a brief evaluation of each option provided:

1. **It doesn't work well for the Sparse matrix.**
   - **False**: Adagrad is generally designed to perform well on sparse data because it adapts the learning rates based on past gradients, making it effective for features that may not update frequently.
2. **It usually goes past the minima.**
   - **False**: Adagrad does not typically overshoot the minima in the same way that momentum-based methods might, but it can become inefficient as the learning rate shrinks too much.
3. **It gets stuck before reaching the minima.**
   - **True**: As Adagrad accumulates squared gradients, the effective learning rate can become very small, potentially leading to getting stuck and making it hard for the algorithm to reach the true minimum.
4. **Weight updates are very small at the initial stages of the algorithm.**
   - **False**: Initially, weight updates can be relatively larger since the parameters begin with a uniform learning rate, but they may decrease over time as the sum of the squared past gradients grows.

9) Consider a gradient profile $\nabla W = [1, 0.9, 0.6, 0.01, 0.1, 0.2, 0.5, 0.55, 0.56]$. Assume $v_{-1} = 0, \epsilon = 0, \beta = 0.9$ and the learning rate is $\eta_{-1} = 0.1$. Suppose that we use the Adagrad algorithm then what is the value of $\eta_6 = \eta/sqrt(v_t + \epsilon)$?

- 0.03
- 0.06
- 0.08
- 0.006

Yes, the answer is correct.
Score: 1

Accepted Answers:
0.06

1. **Understanding Adagrad Update**:
   The learning rate in Adagrad is adjusted based on the accumulated squared gradients. The effective learning rate at time step t is given by:

$$\eta_t = \frac{\eta}{\sqrt{v_t + \epsilon}}$$

where:

- η is the initial learning rate,
- v_t is the accumulated squared gradients up to time step t,
- ϵ is a small value (usually for numerical stability).

2. **Initialization**:
   Given:

   - Initial learning rate $\eta = 0.1$
   - $\epsilon = 0$ (but we will add a small value to avoid division by zero in practice)
   - $\beta = 0.9$

   For this calculation, we will assume $\epsilon = 1 \times 10^{-8}$ for stability.

3. **Calculate the Accumulated Squared Gradients**:
   We need to compute v_t up to $t = 6$ using the provided gradient profile:

$$\nabla W = [1, 0.9, 0.6, 0.01, 0.1, 0.2, 0.5, 0.55, 0.56]$$

The accumulated squared gradient can be computed as follows:

The accumulated squared gradient can be computed as follows:

$$v_0 = 0$$

$$v_1 = v_0 + (1^2) = 0 + 1 = 1$$

$$v_2 = v_1 + (0.9^2) = 1 + 0.81 = 1.81$$

$$v_3 = v_2 + (0.6^2) = 1.81 + 0.36 = 2.17$$

$$v_4 = v_3 + (0.01^2) = 2.17 + 0.0001 = 2.1701$$

$$v_5 = v_4 + (0.1^2) = 2.1701 + 0.01 = 2.1801$$

$$v_6 = v_5 + (0.2^2) = 2.1801 + 0.04 = 2.2201$$

4. **Calculate the Effective Learning Rate η_6:**
   Now we can compute η_6:

$$\eta_6 = \frac{0.1}{\sqrt{v_6 + \epsilon}} = \frac{0.1}{\sqrt{2.2201 + 1 \times 10^{-8}}}$$

Approximating $\sqrt{2.2201}$:

$\sqrt{2.2201} \approx 1.49$ (You could compute this more precisely, but this is sufficient for our

$$\eta_6 \approx \frac{0.1}{1.49} \approx 0.06745$$

5. **Choose Closest Option:**
   Among the given options (0.03, 0.06, 0.08, 0.006), the closest value to our result (0.06745) is **0.06**.

10) What are the two main components of the ADAM optimizer?

- ○ Momentum and learning rate.
- ○ Gradient magnitude and previous gradient.
- ◉ Exponential weighted moving average and gradient variance.
- ○ Learning rate and a regularization term.

Yes, the answer is correct.
Score: 1

Accepted Answers:
Exponential weighted moving average and gradient variance.

Question 6:

In Nesterov accelerated gradient descent, what step is performed before determining the update size?

- A. Increase the momentum
- B. Adjust the learning rate
- C. Decrease the step size
- D. Estimate the next position of the parameters

Answer: D. Estimate the next position of the parameters

Explanation: In Nesterov accelerated gradient descent, the algorithm first estimates the next position of the parameters using the momentum before calculating the gradient at that estimated position to determine the update size.

Question 7:

We have the following functions: z^3, ln(z), e^z, z, and 4. Which of the following functions has the steepest slope at x=1?

- A. z^3
- B. ln(z)
- C. e^z
- D. 4

Answer: A. z^3

Explanation: The slope of a function is given by its derivative. At x = 1:

- The derivative of z^3 is $3z^2 \rightarrow 3(1)^2 = 3$
- The derivative of $\ln(z)$ is $\frac{1}{z} \rightarrow \frac{1}{1} = 1$
- The derivative of e^z is $e^z \rightarrow e^1 \approx 2.718$
- The derivative of z is 1
- The derivative of 4 is 0

The steepest slope at x = 1 is from the function z^3 with a slope of 3.

1. For $f(z) = z^3$:

$$f'(z) = 3z^2 \quad \Rightarrow \quad f'(1) = 3(1)^2 = 3$$

2. For $f(z) = \ln(z)$:

$$f'(z) = \frac{1}{z} \quad \Rightarrow \quad f'(1) = \frac{1}{1} = 1$$

3. For $f(z) = e^z$:

$$f'(z) = e^z \quad \Rightarrow \quad f'(1) = e^1 \approx 2.718$$

4. For $f(z) = z$:

$$f'(z) = 1 \quad \Rightarrow \quad f'(1) = 1$$

5. For $f(z) = 4$:

$$f'(z) = 0 \quad \Rightarrow \quad f'(1) = 0$$

Summary of slopes at $x = 1$:

- z^3: 3
- $\ln(z)$: 1
- e^z: $e \approx 2.718$
- z: 1
- 4: 0

The function with the steepest slope at $x = 1$ is z^3.

- A **steeper slope** means that the function rises or falls more sharply, indicating a larger rate of change at that point.
- Conversely, a **gentler slope** indicates a smaller rate of change.