Numerical Problems - Deep Learning Techniques

Dr. Rajesh Kumar Maurya

August 12, 2025

1 Activation Functions and Feed-Forward Neural Networks

1.1 Compute the Sigmoid Activation Output

Problem Statement: A neuron has an input $x = [2, -3, 1]$, weight vector $w = [0.5, -0.6, 0.8]$, and bias $b = 0.2$. Compute the output using the Sigmoid activation function.

Solution:

$$z = w^T x + b$$

$$= (0.5 \times 2) + (-0.6 \times -3) + (0.8 \times 1) + 0.2$$

$$= 3.8$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-3.8}} = 0.978$$

Final Answer: 0.978

—

1.2 Compute ReLU Activation Output

Problem Statement: Calculate the output for $x = [-2, 0, 3, -5, 4]$ using ReLU.

Solution:

$$\text{ReLU}(x) = \max(0, x)$$

$$\text{ReLU}([-2, 0, 3, -5, 4]) = [0, 0, 3, 0, 4]$$

Final Answer: $[0, 0, 3, 0, 4]$

—

1.3 Compute the Tanh Activation Output

Problem Statement: Given input = 4, weight = -0.7, and bias = 1.5, compute the Tanh activation.

Solution:

$$z = (4 \times -0.7) + 1.5 = -1.3$$

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = -0.861$$

Final Answer: -0.861

1.4 Compute Softmax Probabilities

Problem Statement: A neural network uses the Softmax activation function for classification. Given the input vector:

$$x = [2.0, 1.0, 0.1]$$

Compute the Softmax probabilities.

Solution:

Step 1: Compute exponentials of input values

$$e^{x_1} = e^2 = 7.389, \quad e^{x_2} = e^1 = 2.718, \quad e^{x_3} = e^{0.1} = 1.105$$

Step 2: Compute the denominator (sum of exponentials)

$$\sum e^{x_j} = 7.389 + 2.718 + 1.105 = 11.212$$

Step 3: Compute Softmax probabilities

$$P_i = \frac{e^{x_i}}{\sum e^{x_j}}$$

$$P_1 = \frac{7.389}{11.212} = 0.659$$

$$P_2 = \frac{2.718}{11.212} = 0.242$$

$$P_3 = \frac{1.105}{11.212} = 0.099$$

Final Answer:

$$P_1 \approx 0.659, \quad P_2 \approx 0.242, \quad P_3 \approx 0.099$$

—

1.5 Compute Cross-Entropy Loss

Problem Statement: For a given dataset, a neural network outputs a probability of 0.85 for the positive class and 0.15 for the negative class. Compute the cross-entropy loss for the true label $y = 1$.

Solution:

Cross-Entropy Loss Formula:

$$L = -[y \log p + (1 - y) \log(1 - p)]$$

Step 1: Substitute given values

$$L = -[1 \times \log 0.85 + (1 - 1) \times \log(1 - 0.85)]$$

$$= -[\log 0.85 + 0]$$

Step 2: Compute logarithm value

$$\log 0.85 \approx -0.1625$$

Step 3: Compute the final value

$$L = -(-0.1625) = 0.1625$$

Final Answer:

$$L \approx 0.1625$$

—

2 Unit 2: Beyond Gradient Descent

2.1 Compute Weight Update using Gradient Descent

Problem Statement: Update the weight after one iteration using gradient descent with:

- Learning rate: $\alpha = 0.1$
- Initial weight: $w = 0.5$
- Gradient: $\frac{\partial L}{\partial w} = -0.8$

Solution:

$$w_{\text{new}} = w - \alpha \cdot \frac{\partial L}{\partial w}$$

$$= 0.5 - (0.1 \times -0.8) = 0.58$$

Final Answer: $w_{\text{new}} = 0.58$

—

2.2 Compute Weight Update using AdaGrad

Problem Statement: Compute the updated weight after one step for:

- Initial weight: $w = 0.9$
- Learning rate: $\alpha = 0.01$
- Gradient: $\nabla w_t = -0.5$
- Accumulated squared gradient: $G_t = 0.2$
- Small constant: $\epsilon = 10^{-8}$

Solution:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{G_t + \epsilon}} \nabla w_t$$

$$= 0.9 - \frac{0.01}{\sqrt{0.2 + 10^{-8}}} \times (-0.5) = 0.9112$$

Final Answer: $w_{t+1} = 0.9112$

2.3 Compute Weight Update using Adam Optimizer

Problem Statement: A model using the Adam optimizer has:

- Learning rate: $\alpha = 0.001$
- Initial parameter: $w = 1.0$
- First moment estimate: $m_t = 0.7$
- Second moment estimate: $v_t = 0.2$
- $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$

Compute the updated weight using the Adam rule:

$$w_{t+1} = w_t - \frac{\alpha m_t}{\sqrt{v_t} + \epsilon}$$

Solution:
Substituting values:

$$w_{t+1} = 1.0 - \frac{0.001 \times 0.7}{\sqrt{0.2} + 10^{-8}}$$

$$= 1.0 - \frac{0.0007}{0.4472 + 10^{-8}}$$

$$= 1.0 - \frac{0.0007}{0.4472}$$

$$= 1.0 - 0.00156$$

$$= 0.99844$$

Final Answer: $w_{t+1} = 0.99844$

—

2.4 Compute Weight Update using RMSProp Optimizer

Problem Statement: Consider a RMSProp optimizer with:

- Learning rate: $\alpha = 0.01$
- Initial weight: $w = 2.5$
- Gradient: $\nabla w = -0.7$
- Second moment estimate: $v_t = 0.3$
- Decay rate: $\beta = 0.9$
- Small constant: $\epsilon = 10^{-8}$

Compute the new weight update after one iteration.

Solution:
First, update the second moment estimate:

$$v_t' = \beta v_t + (1 - \beta)(\nabla w)^2$$

$$= 0.9 \times 0.3 + (1 - 0.9) \times (-0.7)^2$$

$$= 0.27 + 0.049 = 0.319$$

Now, update the weight:

$$w_{\text{new}} = w - \frac{\alpha}{\sqrt{v_t'} + \epsilon} \nabla w$$

$$= 2.5 - \frac{0.01}{\sqrt{0.319} + 10^{-8}} \times (-0.7)$$

$$= 2.5 - \frac{0.01}{0.5649} \times (-0.7)$$

$$= 2.5 + 0.0124$$

$$= 2.5124$$

Final Answer: $w_{\text{new}} = 2.5124$

—

2.5 Compute Total Trainable Parameters in a Neural Network

Problem Statement: Suppose a neural network with two hidden layers has:

- First layer (L1): 50 neurons, each with 40 inputs.
- Second layer (L2): 30 neurons, each connected to all L1 neurons.

Compute the total number of trainable parameters (weights and biases). (Hint: Parameters = weights + biases).

Solution:

Step 1: Compute trainable parameters for Layer 1

$$\text{L1 parameters} = (\text{Inputs} \times \text{Neurons}) + \text{Biases}$$

$$= (40 \times 50) + 50$$

$$= 2000 + 50 = 2050$$

Step 2: Compute trainable parameters for Layer 2

$$\text{L2 parameters} = (\text{L1 Neurons} \times \text{L2 Neurons}) + \text{Biases}$$

$$= (50 \times 30) + 30$$

$$= 1500 + 30 = 1530$$

Step 3: Compute total trainable parameters

$$\text{Total parameters} = \text{L1 parameters} + \text{L2 parameters}$$

$$= 2050 + 1530 = 3580$$

Final Answer: 3580 trainable parameters.

—

3 Unit 3: Convolutional Neural Networks (CNNs)

3.1 Compute CNN Output Feature Map Size

Problem Statement: A CNN applies a 3×3 filter on a 28×28 image with stride = 1 and no padding. Compute the output feature map size.

Solution:

$$O = \frac{(I - K)}{S} + 1$$

$$= \frac{(28-3)}{1} + 1 = 26$$

Final Answer: 26×26

—

3.2 Compute CNN Output Feature Map Size with Padding

Problem Statement: A 32×32 grayscale image is passed through a 5×5 filter with stride $= 1$ and padding $= 2$. Compute the output feature map size.

Solution:

$$O = \frac{(I + 2P - K)}{S} + 1$$

$$= \frac{(32 + 4 - 5)}{1} + 1 = 32$$

Final Answer: 32×32

—

3.3 Compute Max Pooling Output Size

Problem Statement: Given a max pooling layer with a 2×2 filter and stride $= 2$, compute the output size of a 64×64 image.

Solution:

$$O = \frac{(I - K)}{S} + 1$$

$$= \frac{(64 - 2)}{2} + 1 = 32$$

Final Answer: 32×32

—

3.4 Compute CNN Trainable Parameters

Problem Statement: A CNN has:

- Input size: 64×64
- Filter size: 3×3
- Stride $= 1$
- Padding $= 1$
- Number of filters $= 32$

Compute:

1. The output size after the first convolutional layer.
2. The total number of trainable parameters for this layer.

Solution:

$$O = \frac{(I + 2P - K)}{S} + 1$$

$$= \frac{(64 + 2 - 3)}{1} + 1 = 64$$

Total trainable parameters:

$$\text{Params} = (K \times K \times \text{Input channels} \times \text{Filters}) + \text{Biases}$$

$$= (3 \times 3 \times 1 \times 32) + 32 = 320$$

Final Answers:

- Output feature map size: 64×64
- Total trainable parameters: 320

3.5 Compute CNN Output Size and Trainable Parameters

Problem Statement: A CNN has:

- Input image size: 64×64
- Filter size: 3×3
- Stride = 1
- Padding = 1
- Number of filters = 32

Compute:

1. The output size after the first convolutional layer.
2. The total number of trainable parameters for this layer.

Solution:
Step 1: Compute Output Size

$$O = \frac{(I + 2P - K)}{S} + 1$$

$$= \frac{(64 + 2 \times 1 - 3)}{1} + 1$$

$$= \frac{(64 + 2 - 3)}{1} + 1$$

$$= \frac{63}{1} + 1 = 64$$

Step 2: Compute Trainable Parameters

$$\text{Trainable Parameters} = (\text{Filter Height} \times \text{Filter Width} \times \text{Input Channels} \times \text{Number of Filters}) + \text{Biases}$$

$$= (3 \times 3 \times 1 \times 32) + 32$$

$$= (9 \times 32) + 32$$

$$= 288 + 32 = 320$$

Final Answers:

- Output feature map size: 64×64
- Total trainable parameters: 320

—

3.6 Compute Output Feature Map and Trainable Parameters

Problem Statement: A CNN takes a $128 \times 128 \times 3$ image as input and applies:

- 64 filters of size $3 \times 3 \times 3$
- Stride = 1, Padding = 1

Compute:

1. The output feature map dimensions.
2. The total number of trainable parameters in this convolutional layer.

Solution:
Step 1: Compute Output Feature Map Size

$$O = \frac{(I + 2P - K)}{S} + 1$$

$$= \frac{(128 + 2 \times 1 - 3)}{1} + 1$$

$$= \frac{(128 + 2 - 3)}{1} + 1$$

$$= \frac{127}{1} + 1$$

$$= 128$$

Since we have 64 filters, the output feature map has dimensions:

$$128 \times 128 \times 64$$

Step 2: Compute Trainable Parameters

$$\text{Trainable Parameters} = (\text{Filter Height} \times \text{Filter Width} \times \text{Input Channels} \times \text{Number of Filters}) + \text{Biases}$$

$$= (3 \times 3 \times 3 \times 64) + 64$$

$$= (27 \times 64) + 64$$

$$= 1728 + 64 = 1792$$

Final Answers:

- Output feature map dimensions: $128 \times 128 \times 64$
- Total trainable parameters: 1792