Assignment 3 - Perceptron, MLP, SVM

by Kunal Sharma

Machine Learning (CSE343/ECE343)

Prof. Jainendra Shukla

Indraprastha Institute of Information Technology

Delhi

Nov 12, 2024

Section A (Theoretical)

A)

Computation Updates for One Training Iteration:

Let's compute the updates for one training iteration using the new initial weights:

Initial Parameters: $w_1 = 0.5,\ b_1 = 0.5,\ w_2 = 0.5,\ b_2 = 0.5$

Learning rate η: 0.01

Dataset: Inputs x = [1, 2, 3], Targets y = [3, 4, 5]

Forward Pass:

Step 1: Compute h for each x

$$h_i = ReLU(w_1 \cdot x_i + b_1)$$

For x = 1:

$$h_1 = ReLU(0.5 \cdot 1 + 0.5) = ReLU(1.0) = 1.0$$

For x = 2:

$$h_2 = ReLU(0.5 \cdot 2 + 0.5) = ReLU(1.5) = 1.5$$

For x = 3:

$$h_3 = ReLU(0.5 \cdot 3 + 0.5) = ReLU(2.0) = 2.0$$

Step 2: Compute $\hat{y}$ for each h

$$\hat{y}_i = w_2 \cdot h_i + b_2$$

For $h_1 = 1.0$:

$$\hat{y}_1 = 0.5 \cdot 1.0 + 0.5 = 1.0$$

For $h_2 = 1.5$:

$$\hat{y}_2 = 0.5 \cdot 1.5 + 0.5 = 1.25$$

For $h_3 = 2.0$:

$$\hat{y}_3 = 0.5 \cdot 2.0 + 0.5 = 1.5$$

Step 3: Compute Loss

The mean squared error (MSE) loss is:

$$L = \left(\frac{1}{3}\right) \sum_{i=1}^{3} (\hat{y}_i - y_i)^2$$

For y = [3, 4, 5] and $\hat{y}$ = [1.0, 1.25, 1.5]:

$$L = \left(\frac{1}{3}\right) [\,(1.0 - 3)^2 + (1.25 - 4)^2 + (1.5 - 5)^2\,] = 7.9375$$

Backward Pass:

Step 1: Gradients of *L* w.r.t $\hat{y}_i$

$$\frac{\partial L}{\partial \hat{y}_i} = 2 \cdot (\hat{y}_i - y_i)$$

For $\hat{y}_1 = 1.0,\ y_1 = 3$:

$$\frac{\partial L}{\partial \hat{y}_i} = 2 \cdot (1.0 - 3) = -4.0$$

For $\hat{y}_2 = 1.25,\ y_2 = 4$:

$$\frac{\partial L}{\partial \hat{y}_i} = 2 \cdot (1.25 - 4) = -5.5$$

For $\hat{y}_3 = 1.5,\ y_3 = 5$:

$$\frac{\partial L}{\partial \hat{y}_i} = 2 \cdot (1.5 - 5) = -7.0$$

Step 2: Gradients w.r.t w_2 and b_2

$$\frac{\partial L}{\partial w_2} = \sum_{i=1}^{3} \frac{\partial L}{\partial \hat{y}_i} \cdot h_i$$

$$\frac{\partial L}{\partial b_2} = \sum_{i=1}^{3} \frac{\partial L}{\partial \hat{y}_i}$$

For w_2:

$$\frac{\partial L}{\partial w_2} = (-4.0 \cdot 1.0) + (-5.5 \cdot 1.5) + (-7.0 \cdot 2.0) = -26.25$$

For b_2:

$$\frac{\partial L}{\partial b_2} = -4.0 + (-5.5) + (-7.0) = -16.5$$

Step 3: Gradients w.r.t w_1 and b_1

$$\frac{\partial L}{\partial h_i} = \frac{\partial L}{\partial \hat{y}_i} . w_2$$

$$\frac{\partial L}{\partial w_1} = \sum_{i=1}^{3} \frac{\partial L}{\partial h_i} . x_i . 1(h_i > 0)$$

$$\frac{\partial L}{\partial b_1} = \sum_{i=1}^{3} \frac{\partial L}{\partial h_i} . 1(h_i > 0)$$

For $h_1, h_2, h_3 > 0$:

$$\frac{\partial L}{\partial h_1} = -4.0 \cdot 0.5 = -2.0, \frac{\partial L}{\partial h_2} = -5.5 \cdot 0.5 = -2.75,$$

$$\frac{\partial L}{\partial h_3} = -7.0 \cdot 0.5 = -3.5$$

For w_1:

$$\frac{\partial L}{\partial w_1} = (-2.0 \cdot 1) + (-2.75 \cdot 2) + (-3.5 \cdot 3) = -18.0$$

For b_1:

$$\frac{\partial L}{\partial b_1} = -2.0 + (-2.75) + (-3.5) = -8.25$$

Weight and Bias Updates:

Update weights and biases using:

$$w_1 \leftarrow w_1 - \eta \cdot \frac{\partial L}{\partial w_1}_1,\ b_1 \leftarrow b_1 - \eta \cdot \frac{\partial L}{\partial b_1}$$

$$w_2 \leftarrow w_2 - \eta \cdot \frac{\partial L}{\partial w_2},\ b_2 \leftarrow b_2 - \eta \cdot \frac{\partial L}{\partial b_2}$$

For $w_1\ and\ b_1$:

$$w_1 = 0.5 - 0.01 \cdot (-18.0) = 0.5 + 0.18 = 0.68$$

$$b_1 = 0.5 - 0.01 \cdot (-8.25) = 0.5 + 0.0825 = 0.5825$$

For $w_2\ and\ b_2$:

$$w_2 = 0.5 - 0.01 \cdot (-26.25) = 0.5 + 0.2625 = 0.7625$$

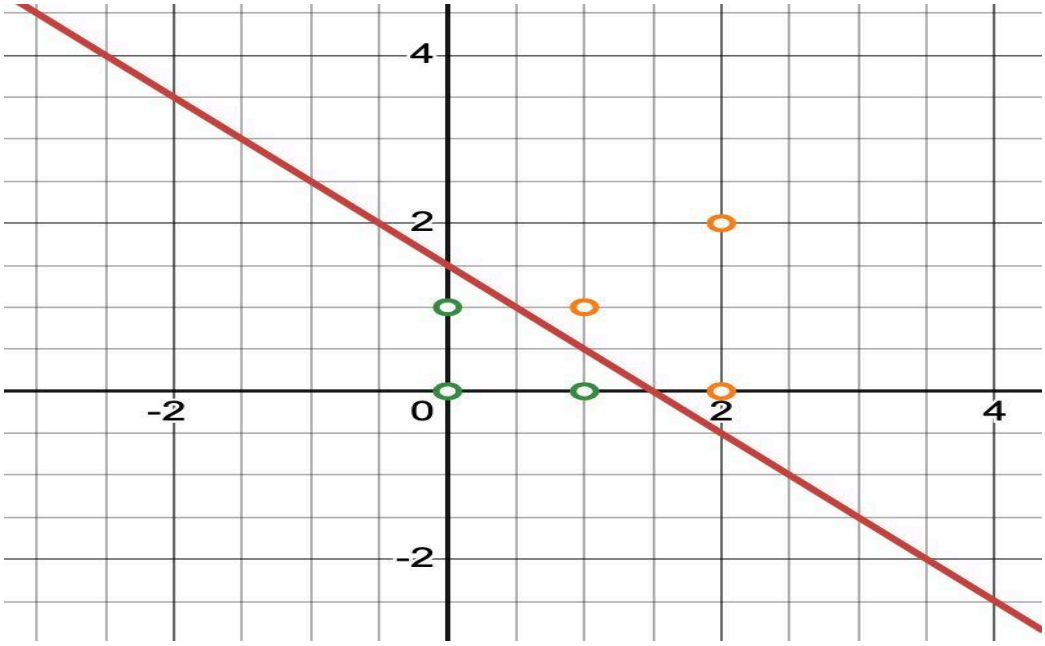
$$b_2 = 0.5 - 0.01 \cdot (-16.5) = 0.5 + 0.165 = 0.665$$

Hence, Updated Parameters:

$$w_1 = 0.68,\ b_1 = 0.5825,\ w_2 = 0.7625,\ b_2 = 0.665$$

B)

a) Yes, the points are linearly separable as shown below:



b) Weight Vector and Support Vectors:

Step 1: The Equation of a Hyperplane

The equation of a hyperplane in 2D is given by:

$$x_1 w_1 + x_2 w_2 + b = 0$$

where: w = [w_1, w_2] is the weight vector, and b is the bias term.

The hyperplane separates the points such that:

$$x_1 w_1 + x_2 w_2 + b > 0 \; for \; y = +1,$$

$$x_1 w_1 + x_2 w_2 + b < 0 \; for \; y = -1.$$

Step 2: Maximum Margin Hyperplane

The maximum margin hyperplane is equidistant from the closest points of both classes, known as the support vectors.

Let the margin distance be γ. The hyperplane satisfies:

$$x_1 w_1 + x_2 w_2 + b \geq +1 \; for\; positive\; class\; support\; vectors,$$

$$x_1 w_1 + x_2 w_2 + b \leq -1 \; for\; negative\; class\; support\; vectors.$$

The margin is given by:

$$\gamma = \frac{2}{||w||},\; where\; ||w|| = \sqrt{w_1^{\,2} + w_2^{\,2}}$$

Positive Points (+1):

• For (0, 0): $w_1 \cdot 0 + w_2 \cdot 0 + b \geq 1$

$$\Rightarrow b \geq 1 \quad --- \; (1)$$

• For (1, 0): $w_1 \cdot 1 + w_2 \cdot 0 + b \geq 1$

$$\Rightarrow w_1 + b \geq 1 \quad --- \; (2)$$

• For (0, 1): $w_1 \cdot 0 + w_2 \cdot 1 + b \geq 1$

$$\Rightarrow w_2 + b \geq 1 \quad --- \; (3)$$

Negative Points (−1):

• For (1, 1): $w_1 \cdot 1 + w_2 \cdot 1 + b \leq -1$

$$\Rightarrow w_1 + w_2 + b \leq -1 \quad --- \; (4)$$

• For (2, 2): $w_1 \cdot 2 + w_2 \cdot 2 + b \leq -1$

$$\Rightarrow 2w_1 + 2w_2 + b \leq -1 \quad --- \; (5)$$

• For (2, 0): $w_1 \cdot 2 + w_2 \cdot 0 + b \leq -1$

$$\Rightarrow 2w_1 + b \leq -1 \quad --- \; (6)$$

Making the constraints tight as Support vectors lie on boundary:

Step 1: Solve for b in terms of w_1 and w_2:

From Equation (2): $b = 1 - w_1 \quad --- (7)$

From Equation (3): $b = 1 - w_2 \quad --- (8)$

Setting the two expressions for b equal:

$$1 - w_1 = 1 - w_2 \Rightarrow w_1 = w_2 \quad --- (9)$$

Step 2: Substitute $w_1 = w_2$ into Equation (4)

From Equation (4): $w_1 + w_2 + b = -1$

Substitute $w_1 = w_2$ and $b = 1 - w_1$ (from Equation 7):

$$w_1 + w_1 + (1 - w_1) = -1$$

$$2w_1 + 1 - w_1 = -1$$

$$w_1 = -2$$

Step 3: Solve for b and w_2

From Equation 9 ($w_1 = w_2$):

$$w_2 = -2$$

Substitute $w_1 = -2$ into Equation 7 ($b = 1 - w_1$):

$$b = 1 - (-2)$$

$$b = 3$$

Final Solution:

$$w_1 = -2, w_2 = -2, b = 3$$

Support vectors are the points that lie on the boundary:

- Of positive (+) class: (0, 1), (1, 0)
- Of negative (−) class: (1, 1), (2, 0)

SECTION C

2)

For each activation function, I plotted the training loss vs. epochs and validation loss vs. epochs curves to monitor convergence and generalization. The test accuracy was evaluated for each activation function, and the results are summarized below:

Activation Function	**Final Training Loss**	**Final Validation Loss**	**Test Accuracy**
Logistic	1.6676	1.6753	0.4510
Tanh	0.4246	0.4552	0.8315
Relu	0.4110	0.4439	0.8360
Identity	0.4143	0.4533	0.8275

Observations :

1. The logistic activation function yielded the lowest test accuracy (45.10%) and demonstrated poor convergence.
2. The tanh, relu, and identity activation functions exhibited significantly better convergence and higher test accuracies.
3. Among these, the relu activation function performed the best, achieving the highest test accuracy of **83.60%**, indicating its superior capability for this task.

Conclusion : The relu activation function was determined to be the best for this dataset and task, as it achieved the highest test accuracy and demonstrated efficient convergence during training.

3)

I performed a grid search to identify the best hyperparameters for the MLP Classifier using the relu activation function, which was determined to be the best in the previous step. The

hyperparameters considered in the grid search included batch_size, learning_rate_init, and solver.

The grid search was conducted using cross-validation to ensure reliable evaluation of model performance. The best hyperparameters and their corresponding results are summarized below:

Best Hyperparameters :

1. **Batch Size**: 64
2. **Learning Rate**: 0.001
3. **Solver**: adam

Results :

1. **Best Cross-Validation Accuracy**: 86.19%
2. **Test Accuracy of Best Model**: 84.75%

Conclusion : The optimal hyperparameters for the MLP Classifier were a batch size of 64, a learning rate of 0.001, and the ‘adam’ solver. This configuration achieved a test accuracy of **84.75%**, demonstrating improved performance compared to the initial configuration.

4)

I trained two MLPRegressor models for the regeneration task using a 5-layer neural network with the structure [c, b, a, b, c] (where c>b>a). The task was to regenerate input images using the designed network. Both models were trained using the adam solver, a constant learning rate of 2e-5, and different activation functions (relu and identity). The training and validation losses per epoch were recorded to monitor the model’s convergence.

Model Performance :

1. **Relu Activation :**
    a. Final Training Loss: **0.022850**
    b. Final Validation Loss: **0.023132**
2. **Identity Activation :**

a. Final Training Loss: **0.021113**
b. Final Validation Loss: **0.021404**

Visualizations of Regenerated Images : The visualizations of regenerated images for the 10 test samples are shown in the figure above. The original images are displayed alongside the regenerated outputs for both relu and identity activation functions.

Observations :

1. **Relu Activation :**
    a. The regenerated images show a reasonable level of sharpness and detail.
    b. Although the training and validation losses were slightly higher compared to the identity activation, the generated images retained significant visual features of the original inputs.
2. **Identity Activation :**
    a. The regenerated images exhibit marginally more accurate regeneration in terms of finer details and smoother outputs.
    b. The slightly lower training and validation losses compared to relu indicate that the network with identity activation was able to generalize slightly better.

Conclusion : Based on the training and validation losses, as well as the regenerated images:

1. The **identity activation function** resulted in better performance with slightly lower losses and more refined regenerated outputs.
2. However, the differences between the two activation functions are minimal, with both being effective for the regeneration task.

5)

I extracted bottleneck features of size a from the two previously trained MLPRegressor models using the ‘relu’ and ‘identity’ activation functions. These features were used as the new set of image representations for training two smaller MLP classifiers with 2 layers, each of size a. Both classifiers were trained for 200 iterations using the same adam solver and a learning rate of 2e−5 as used in Part 2.

Results :

1. **Test Accuracy for Classifiers trained on Bottleneck Features :**
    a. relu: **76.40%**
    b. identity: **79.35%**
2. **Test Accuracies from Part 2 for Comparison :**
    a. logistic: **45.10%**
    b. tanh: **83.15%**
    c. relu: **83.60%**
    d. identity: **82.75%**

Observations :

1. The classifiers trained on bottleneck features achieved relatively high test accuracies (76.40% for ‘relu’ and 79.35% for identity), though slightly lower than the original MLPClassifier models trained in Part 2.
2. The bottleneck features effectively reduced the dimensionality of the input data while preserving meaningful image representations, resulting in a robust feature set for classification.

Reasons for Decent Performance :

1. The bottleneck features extracted by the MLPRegressor models encode high-level image representations that simplify the task of the smaller classifiers, making them less prone to overfitting and reducing the complexity of the classification task.
2. The feature extraction process removes noise and irrelevant details, leaving more discriminative features for training the classifiers.
3. While the classifiers trained in Part 2 had direct access to the raw input data, the smaller classifiers trained on bottleneck features benefited from the pre-learned abstract representations, achieving decent performance even with reduced complexity.