Explanation of Backpropagation Implementation

Selim Tarık ARI

1 Introduction

This document provides an explanation of the backpropagation implementations in two multilayer perceptron (MLP) models: one for classification and one for regression. Each model includes two layers, a hidden layer and an output layer, and utilizes different activation functions and loss functions tailored to each task.

2 MLPClassifier: Classification with Cross-Entropy Loss

The MLP classifier is implemented in `mlp_classification_backpropagation.py`. This network is structured with an input layer (4 nodes), a hidden layer (3 nodes), and an output layer (3 nodes). The hidden layer uses a sigmoid activation function, and the output layer uses a softmax activation function to produce probabilities for each class. Training is performed using the cross-entropy loss function.

2.1 Forward Pass

In the forward pass:

- The hidden layer input is computed as:

  $$\text{hidden\_input} = x \cdot W + W\_bias$$

  where x is the input vector, W is the weight matrix connecting the input to the hidden layer, and W_bias is the bias for the hidden layer.

- The hidden layer output applies the sigmoid activation:

  $$\text{hidden\_layer\_output} = \sigma(\text{hidden\_input}) = \frac{1}{1 + e^{-\text{hidden\_input}}}$$

- The output layer input is computed as:

  $$\text{output\_input} = \text{hidden\_layer\_output} \cdot \Gamma + \Gamma\_bias$$

  where Γ and Γ_bias are the weights and biases connecting the hidden layer to the output layer.

- The softmax function is applied to obtain the final output probabilities:

$$\text{output\_layer\_output} = \text{softmax}(\text{output\_input}) = \frac{e^{\text{output\_input}}}{\sum e^{\text{output\_input}}}$$

2.2 Backpropagation

The goal is to minimize the cross-entropy loss:

$$\text{CrossEntropy} = -\sum_i y_i \log(\hat{y}_i)$$

where y_i is the true label, and $\hat{y}_i$ is the predicted probability.

- The output layer error is calculated as:

$$\text{output\_error} = \text{output\_layer\_output} - \text{label}$$

- The delta for the output layer (gradient of the error) is:

$$\text{output\_delta} = \text{output\_error}$$

- The hidden layer error is obtained by backpropagating the output error through Γ:

$$\text{hidden\_error} = \text{output\_delta} \cdot \Gamma^T$$

- The delta for the hidden layer is calculated using the sigmoid derivative:

$$\text{hidden\_delta} = \text{hidden\_error} \cdot \sigma'(\text{hidden\_layer\_output})$$

The weight updates are calculated as:

$$\Gamma\text{\_update} = \text{hidden\_layer\_output}^T \cdot \text{output\_delta}$$

$$W\text{\_update} = x^T \cdot \text{hidden\_delta}$$

Bias updates:

$$\Gamma\text{\_bias\_update} = \text{output\_delta}$$

$$W\text{\_bias\_update} = \text{hidden\_delta}$$

Finally, the weights are updated by applying the learning rate:

$$W = W - \text{learning\_rate} \times W\text{\_update}$$

$$\Gamma = \Gamma - \text{learning\_rate} \times \Gamma\text{\_update}$$

and similarly for biases.

3 MLPRegressor: Regression with Mean Squared Error Loss

The MLP regressor in `mlp_regression_backpropagation.py` is structured similarly to the classifier but uses a linear output layer. The network aims to minimize the mean squared error (MSE) loss.

3.1 Forward Pass

In the forward pass:

- The hidden layer input and output are computed as:

$$\text{hidden\_layer\_input} = x \cdot W + W\_bias$$

$$\text{hidden\_layer\_output} = \sigma(\text{hidden\_layer\_input})$$

- The output layer applies an identity activation (linear output):

$$\text{output\_layer\_output} = \text{hidden\_layer\_output} \cdot \Gamma + \Gamma\_bias$$

3.2 Backpropagation

The regression model minimizes the mean squared error (MSE):

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

- The output error is calculated as:

$$\text{output\_error} = 2 \times (\text{output\_layer\_output} - \text{label})$$

- The gradient for Γ and Γ_bias is computed as:

$$\Gamma\text{\_update} = \text{hidden\_layer\_output}^T \cdot \text{output\_error}$$

$$\Gamma\text{\_bias\_update} = \text{output\_error}$$

- The hidden layer error is calculated as:

$$\text{hidden\_error} = \text{output\_error} \cdot \Gamma^T \cdot \sigma'(\text{hidden\_layer\_output})$$

- The gradient for W and W_bias is computed as:

$$W\text{\_update} = x^T \cdot \text{hidden\_error}$$

$$W\text{\_bias\_update} = \text{hidden\_error}$$

Weight updates are applied as in the classifier:

$$W = W - \text{learning\_rate} \times W\text{\_update}$$

$$\Gamma = \Gamma - \text{learning\_rate} \times \Gamma\text{\_update}$$

with corresponding bias updates.

4 Conclusion

This document explains the backpropagation process for both the MLP classifier and regressor. The classifier utilizes cross-entropy loss for multi-class prediction, while the regressor uses MSE for single-output regression. Both models update weights and biases iteratively based on the gradients calculated during each epoch.