ECON 6930 — Problem Set 3

Nicola Pipoli

November 9, 2024

1 Basic Architecture of a Neural Network

Weights and Biases Initialization:

- W_1 represents the weight matrix for the input layer to the hidden layer, with dimensions $(4, 3)$, meaning it has 4 neurons in the hidden layer and 3 input features in the input layer.
- b_1 represents the bias vector for the hidden layer, with dimensions $(4, 1)$.
- W_3 represents the weight matrix for the hidden layer to the output neuron, with dimensions $(1, 4)$.
- b_3 represents the bias for the output neuron, with dimensions $(1, 1)$ (same dimension as the output).

2 Activation Functions

- **Sigmoid:** $\frac{1}{1+\exp(-x)}$ gives an S-shaped curve, making sure the output falls between 0 and 1.
- **ReLU:** Outputs 0 for negative inputs and the input itself if greater than 0.
- **Leaky ReLU:** Allows a small gradient (controlled by α) for negative inputs. In my case I am using $\alpha = 0.1$.

Input Range

Inputs from -10 to 10 are generated to show the behavior of each activation function across both positive and negative values and how non-linearity is introduced in the neural network.

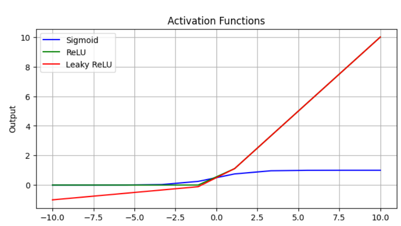

Figure 1: ReLU, Leaky ReLU and Sigmoid Activation Functions Plot.

.

3 Forward Propagation

The code for this taks performs a forward pass.

4 Over-fitting and Regularization

MSE Loss Calculation calculates the Mean Squared Error between the predicted output and the true target value. L2 Regularization Term calculates the L2 penalty for weights W_1 and W_3. The regularization factor λ_{reg} is used to change the strength of regularization. A higher λ_{reg} value will penalize larger weights more heavily.

$$\text{total\_loss} = \text{mse\_loss} + \text{l2\_loss}$$

represents the final loss, which combines the MSE loss and the L2 regularization term.

5 Adaptive Learning Rates

The code for this task implements a simple gradient descent loop to train the neural network.

6 Training a Neural Network

The Adam optimizer is an advanced version of gradient descent that adapts the learning rate for each parameter based on past gradient information:

- **Momentum:** It averages past gradients (1st moment), smoothing out updates and reducing oscillations.
- **Adaptive Rates:** It keeps a separate average of squared gradients (2nd moment) to scale learning rates individually for each parameter, adjusting for the gradient's magnitude.
- **Bias Correction:** Corrects initial biases in these averages for accurate updates early on.

Adam's adaptive and smooth updates make training faster and more stable, especially on complex models.