Neural Networks Formula Sheet

Motivation

- Classification: Approximate membership function $y = f(x, w)$, where y is discrete.
- Regression: Approximate generating function $y = f(x, w)$, where y is continuous.

Artificial Neuron

$$\text{Weighted Sum: } s = \sum_i w_i x_i$$

$$\text{Activation Function: } g(s) = \begin{cases} 1, & \text{if } s \geq \theta \\ 0, & \text{if } s < \theta \end{cases}$$

Learning: Adjust weights w_i to obtain intended outputs.

Single-layer Perceptron

- Classification Function: $g(u) = \mathbf{w} \cdot \mathbf{u}$.
- Hyperplane: $g(u) = 0$, divides classes.
- Perceptron Convergence Theorem: Finds a hyperplane for any linearly separable dataset in finite steps.
- Limitation: Cannot classify non-linearly separable data (e.g., XOR).

Multi-layer Perceptron (MLP)

- Layered Structure: Inputs pass sequentially through hidden layers to outputs.

- Feedforward Propagation:

$$a_j = g\left(\sum_i w_{i,j} a_i + b_j\right)$$

- Activation Functions:
  - Linear Function: $g(s) = s$
  - Step Function:

$$g(s) = \begin{cases} 1, & \text{if } s \geq 0 \\ 0, & \text{if } s < 0 \end{cases}$$

  - Sigmoid: $g(s) = \frac{1}{1+e^{-s}}$
  - Hyperbolic Tangent (Tanh): $g(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$
  - ReLU: $g(s) = \max(0, s)$
  - Leaky ReLU:

$$g(s) = \begin{cases} s, & \text{if } s > 0 \\ \alpha s, & \text{if } s \leq 0 \end{cases}$$

  where α is a small positive constant.

Linear Discriminant Function

$$g(u) = \mathbf{w} \cdot \mathbf{u} + b$$

- $\mathbf{w}$: Weight vector.
- $\mathbf{u}$: Input vector.
- b: Bias.

Learning Rule

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta(y - \hat{y}) x_i$$

- η: Learning rate.

- y: Desired output.
- $\hat{y}$: Predicted output.

Backpropagation Algorithm

1. **Forward Pass:** Compute outputs for all layers.
2. **Error Calculation:**
$$E = \frac{1}{2} \sum (y_{\text{desired}} - y_{\text{actual}})^2$$
3. **Backward Pass:** Update weights to minimize error:
$$\Delta w_{i,j} = -\eta \frac{\partial E}{\partial w_{i,j}}$$
where η is the learning rate.

Hebbian Learning Rule

- Update Rule: $\Delta w_{ij} = \eta a_i a_j$
- Principle: "Neurons that fire together, wire together."

Activation Functions in Neural Networks

1. Linear Function

$$g(s) = s$$

Used for regression tasks where the output is continuous.

2. Step Function

$$g(s) = \begin{cases} 1, & \text{if } s \geq 0 \\ 0, & \text{if } s < 0 \end{cases}$$

Commonly used in early perceptron models for binary classification.

3. Sigmoid Function

$$g(s) = \frac{1}{1 + e^{-s}}$$

Maps inputs to the range $(0, 1)$, often used in the output layer for binary classification.

4. Hyperbolic Tangent (Tanh) Function

$$g(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$$

Maps inputs to the range $(-1, 1)$. It is zero-centered and often preferred over sigmoid for hidden layers.

5. Rectified Linear Unit (ReLU) Function

$$g(s) = \max(0, s)$$

Efficient and widely used activation function for hidden layers in deep learning.

6. Leaky ReLU Function

$$g(s) = \begin{cases} s, & \text{if } s > 0 \\ \alpha s, & \text{if } s \leq 0 \end{cases}$$

where α is a small positive constant, typically $\alpha = 0.01$. This prevents the dying ReLU problem.

7. Parametric ReLU (PReLU) Function

$$g(s) = \begin{cases} s, & \text{if } s > 0 \\ \alpha s, & \text{if } s \leq 0 \end{cases}$$

where α is learned during training.

8. Exponential Linear Unit (ELU) Function

$$g(s) = \begin{cases} s, & \text{if } s > 0 \\ \alpha(e^s - 1), & \text{if } s \leq 0 \end{cases}$$

where α is a hyperparameter. ELU is smooth and avoids the dead ReLU problem.

9. Softmax Function

$$g_i(s) = \frac{e^{s_i}}{\sum_j e^{s_j}}$$

Used in the output layer for multi-class classification. It converts logits into probabilities.

10. Swish Function

$$g(s) = s \cdot \sigma(s)$$

where $\sigma(s)$ is the sigmoid function. Swish is a smooth, self-gated activation function.