Chapter 8

Artificial Neural Network

Contents

8.1 ANN Properties

- ANN is much like a black-box.
- Many neuron-like threshold switching units.
- Many weighted interconnections amount units.
- Highly parallel and distributed process.

8.2 Perceptron

Linear function: $f : X \to Y$

$$\begin{aligned} f(\mathbf{x}) &= w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d \\ &= w_0 + \sum_{j=1}^{d} w_j x_j \end{aligned}$$

$w_0, w_1, \ldots, w_d$ are weights

8.3 Multi-layer Perceptron

- Multi-layer perceptrons (MLPs) were designed to overcome the computational limitation of a single threshold element (perceptron).

- The idea is to stack several layers of threshold elements, each layer using the output of the previous layer as input.
- A feed-forward MLP network defines a mapping:

$$y = f(x; \theta)$$

- Functions are composed in a chain:

$$f(x) = f_3(f_2(f_1(x)))$$

- Input layer: The process starts with the input layer, which receives the input data. Each neuron in this layer represents a feature of input data.
- Weights and Biases: Connections between neurons have associated weights, which are learned during the training process and is crucial to capture patterns in the data.
- Hidden Layers: The neurons in these layers perform computations on the inputs. The output of each neuron is calculated by applying a weighted sum of its inputs (from the previous layer).
- Activation Functions: The activation function is crucial as it introduces non-linearity into the model, allowing it to learn more complex functions. . . .

8.4 Activation Functions

- Non-linearity: This is the most fundamental property; activation functions introduce nonlinearity into the network. This is important because real-world relationships and patterns are rarely linear.
- . . .
- Monotonicity: A monotonic activation function either strictly increases or strictly decreases as input values change. This property ensures that as inputs change, the neuron's output moves in a consistent direction.
- Continuity: A continuous activation function produces smooth and continuous changes in output as inputs change slightly. This property helps in smooth gradient computations for updating weights during the learning process.
- Differentiability: Differentiability is essential for gradient-based optimization algorithms like backpropagation. Activation functions that are differentiable across their domain allow gradients to be computed for weight updates during training.
- Sparsity: Some activation functions promote sparsity by having their outputs be zero for a large portion of input space. This can be beneficial in reducing the complexity of neural networks.

Rectified Linear Units (ReLU):

$$g(x) = \max\{0, x\}$$
$$g'(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{8.1}$$

Sigmoid:

$$g(x) = \frac{1}{1 + e^{-x}}$$
$$g'(x) = g(x)\,(1 - g(x)) \tag{8.2}$$

Hyperbolic tangent tanh

$$g(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$
$$g'(x) = 1 - g(x)^2 \tag{8.3}$$

Table 8.1: Activation Functions

Activation Functions	**Sigmoid**	**tanh**	**ReLU**
Range	$(0, 1)$	$(-1, 1)$	$[0, \infty)$
Vanishing Gradient Problem	Yes	Yes	No
Nature	Non-Linear	Non-Linear	Linear
Zero Centered Activation Function	No	Yes	No
Symmetric Function	No	Yes	No

8.5 Feed-forward MLP

- The weights of the neural network connections are repeatedly adjusted to minimize the difference between the actual output and desired output.
- Aims to minimize the loss function by adjusting the network's weights and biases. The loss function gradients determine the level of adjustment with respect to parameters like activation function, weights, bias, etc.
- The forward step, given the input, computers the output layer-by-layer, starting with the input layer.
- The backward step calculates the error in the output and propagates it backwards; then update the weights layer-by-layer, starting from the output layer.

8.6 Forward Pass

- In this example, we will be using a three-layer neural network with the layers being the input, hidden, and output layers.
- The activation ...

8.7 Gradient Descent

- Goal of NN: Given n training samples (x_i, y_i), find w to minimize:

$$E[w] = \frac{1}{2n} \sum_{i=1}^{n} (y_i - o_i)^2 \tag{8.4}$$

- ...

8.8 Chain Rule

- $\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$
- $\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2} \frac{\partial y_2}{\partial x}$