Tutorial 4

Supervised Learning using Neural Network

Luke Chang

The University of Auckland

Mar. 2021

Objectives

1. Overview on Neural Network
2. Activation Functions
3. Loss Functions
4. Optimization
5. Regularization
6. Different Types of Layers
7. Tutorial Questions

Overview on Neural Network

Where does neural network (NN) shine?

- Commonly used in supervised learning where data has a lot of instances and feature space is large.
- It scales well when data size increases.

An (artificial) neural network is a direct graph.

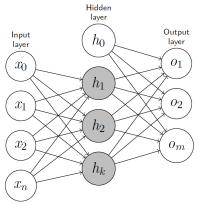
- **Feed-forward neural network (FFNN):** A directed acyclic graph, where nodes are arranged in layers from inputs to outputs.
- **Recurrent neural network (RNN):** A directed graph with cycles; nodes have additionally feedback into themselves or previous nodes.

Motivation

To combat **gradient vanishing** problem, *feedback nodes* in the hidden layers are commonly used in large-scale deep neural networks.

A basic NN with 1 hidden layer

- The data has n features, and the outputs have m classes.
- k hidden neurons in the hidden layer.
- Let x be the input vector where $x \in^{\mathrm{n}}$, hidden layer h is a vector where $h \in^{\mathrm{k}}$ and output o is a vector where $o \in^{\mathrm{m}}$.
- Nodes are connected by weights. These weights are learnt during the training. The weight matrices for the hidden layer is $W_0 \in^{\mathrm{k}\times\mathrm{n}}$, and for the output layer is $W_1 \in^{\mathrm{m}\times\mathrm{k}}$.
- Biases are $b_0 \in^{\mathrm{k}}$ and $b_1 \in^{\mathrm{m}}$, where $x_0 = h_0 = 1$.



$o = f(W_1 \cdot f(W_0 \cdot x + b_0) + b_1)$, where f is the activation function, applied element-wise.

Note: The last activation function should based on the desired outputs. (Eg: One-hot encoding, regression)

Activation Functions

The activation function must be **non-linear**.

For a given non-input node h:

- Suppose there are n nodes connect to h.
- Let x be the column vector input to the node, that $x = (x_0, x_1, \ldots, x_n)^T$, where $x_0 = 1$.
- Let w be the weights on the incident edges, that $w = (w_0, w_1, \ldots, w_n)^T$, where $w_0 = b$ is the bias.

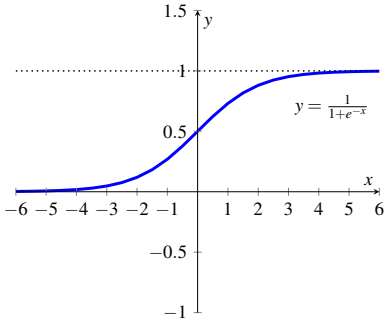
The output from this node is given by:

$$f(w^T x) = f(\sum_{i=0}^{n} w_i x_i) = f(\sum_{i=1}^{n} w_i x_i + b)$$

Where f is the activation function for this node.

If the activation function is linear, we can merge multiple hidden layers into one layer. The structure is equivalent to linear regression.

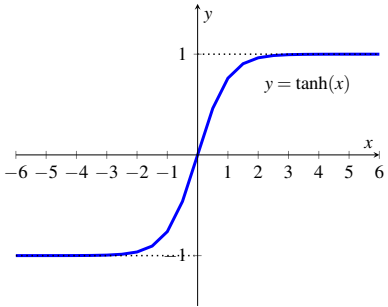
Sigmoid Function



$$f(x) = \text{Sigmoid}(x) = \sigma(x) = \frac{1}{1+e^{-x}}$$

Where $f(x) \in (0,1)$, and the derivative is $f'(x) = f(x)(1-f(x))$.
Rarely used in hidden layers for state-of-the-art models.
Often used in the last hidden layer to produce logistic outputs.

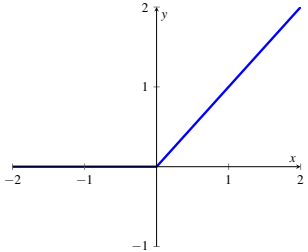
Hyperbolic Tangent (tanh) Function



$$f(x) = \tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Where $f(x) \in (-1, 1)$, and the derivative is $f'(x) = 1 - f(x)^2$
Similar to Sigmoid function, but faster to converge.

Rectified Linear Unit (ReLU) Function



$$f(x) = \text{ReLU}(x) = (x)^+ = \max(0, x)$$

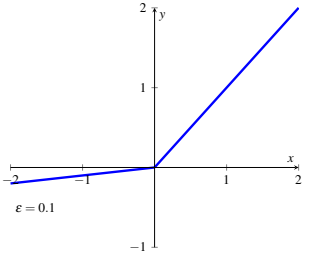
Where $f(x) \in [0, +\infty)$. The derivative is equal to:

$$f'(x) = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x > 0 \end{cases}$$

Where $x = 0$ is not differentiable.

ReLU is the most common activation function for hidden layers in recent deep learning.

Leaky ReLU



Let ε be the negative slope:

$$f(x) = \text{LeakyReLU}(x) = \max(0,x) + \varepsilon \times \min(0,x) = \begin{cases} x, & \text{if } x \geq 0 \\ \varepsilon \times x, & \text{otherwise} \end{cases}$$

Where $f(x) \in$, and $\varepsilon \in [0,1)$ The derivative is equal to ($x = 0$ is not differentiable):

$$f'(x) = \begin{cases} \varepsilon, & \text{if } x < 0 \\ 1, & \text{if } x > 0 \end{cases}$$

Similar to ReLU, but overcomes the "dead neuron" problem.

Loss Functions / Cost Function

Optimization

Dropout Layers

Convolution Layers

Pooling Layers

Recurrent Layers

Transformer Layers

Tutorial Questions