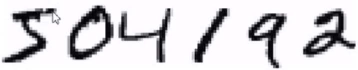
Neural Networks

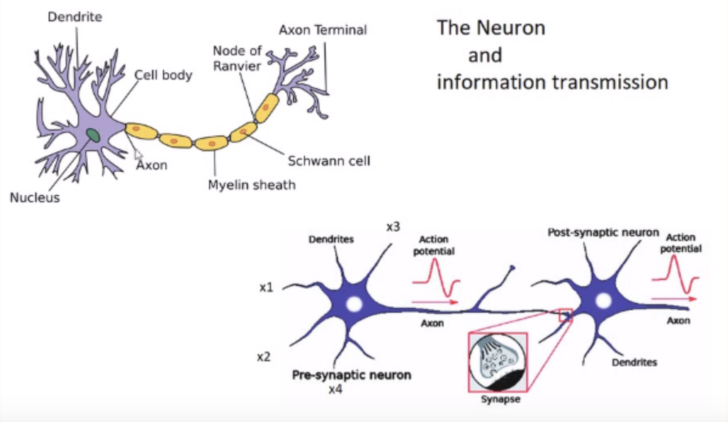
Steve Cygu

January 23, 2019

OUTLINE

Can we write algorithm to correctly identify each of the objects?




The Neuron
and
information transmission
Dendrite
Axon Terminal
Node of Ranvier
Cell body
Schwann cell
Axon
Myelin sheath
Nucleus
Dendrites
x3
Action potential
Post-synaptic neuron
Action potential
x1
Axon
Axon
x2
Dendrites
Pre-synaptic neuron
x4
Synapse

Layers

- **Input nodes**
  No computation
- **Hidden nodes (Neurons)**
  Intermediate processing and computation and transfers (another hidden layer or output).
- **Output nodes**
  Uses a function (not necessarily activation function) to map the input from other layers to desired output format.
  - Sigmoid
  - Softmax

Synapse/Connections

- Transfers the output of neuron i to the input of neuron j.
- Each connection is assigned weight, W_{ij}

Activation function

Introduces nonlinearity into the neuron output.

- Sigmoid (Logistic Activation Function)

$$a(z) = \frac{1}{1 + exp(-z)}$$

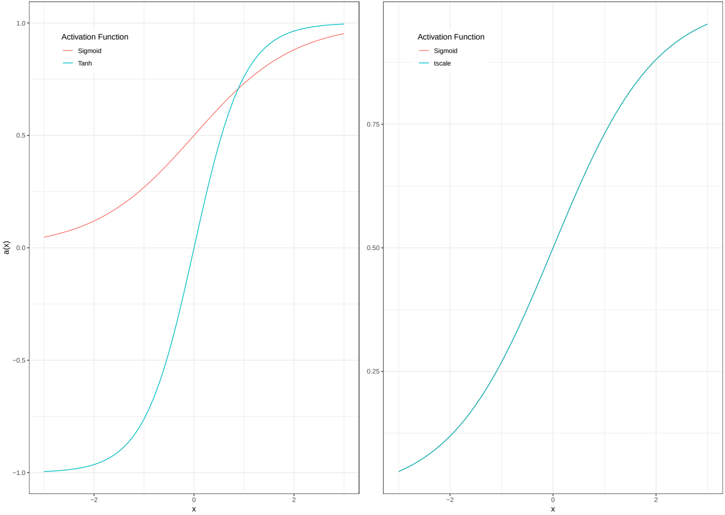
- Tanh (hyperbolic tangent Activation Function)

$$a(z) = tanh(z) = \frac{2}{1 + exp(-2z)} - 1 = 2sigmoid(2z) - 1$$

- ReLU (Rectified Linear Unit Activation Function)

$$a(z) = max(0, z)$$

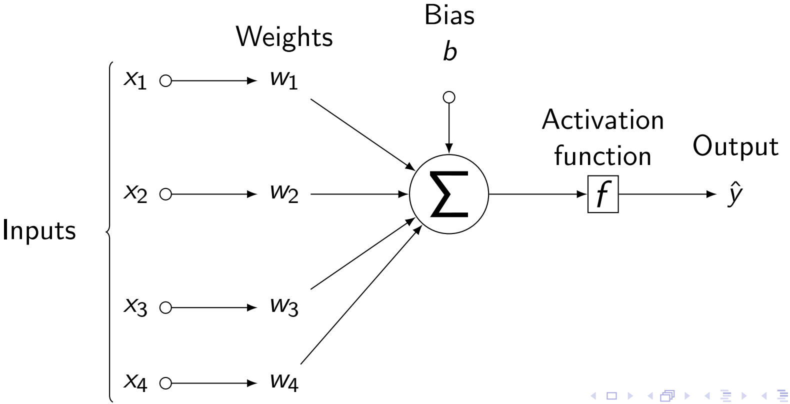
Sigmoid and Tanh Activation Functions



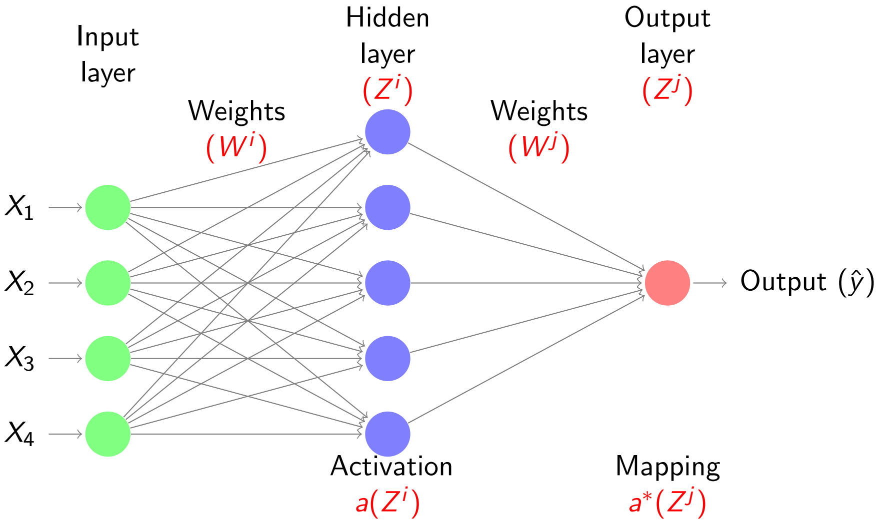
Single-layer Perceptron

- No hidden layer, a single neuron.

$$\hat{y} = b + \sum_{i=1}^{n} x_i w_i$$



Multi-layer perceptron (MLP)



Other types

- Convolutional Neural Network (CNN)
- Recurrent neural networks

Weights are the parameters. The generic approach is by **gradient descent**.

- Forward-propagation (feed-forward)
- Backward-propagation

Consider,

$$Z_1^i = X_1 W_{1,1}^i + X_2 W_{2,1}^i + X_3 W_{3,1}^i + X_4 W_{4,1}^i$$

Consider,

$$Z_1^i = X_1 W_{1,1}^i + X_2 W_{2,1}^i + X_3 W_{3,1}^i + X_4 W_{4,1}^i$$
$$Z_2^i = X_1 W_{1,2}^i + X_2 W_{2,2}^i + X_3 W_{3,2}^i + X_4 W_{4,2}^i$$

Consider,

$$Z_1^i = X_1 W_{1,1}^i + X_2 W_{2,1}^i + X_3 W_{3,1}^i + X_4 W_{4,1}^i$$

$$Z_2^i = X_1 W_{1,2}^i + X_2 W_{2,2}^i + X_3 W_{3,2}^i + X_4 W_{4,2}^i$$

$$\vdots$$

Consider,

$$
\begin{aligned}
Z_1^i &= X_1 W_{1,1}^i + X_2 W_{2,1}^i + X_3 W_{3,1}^i + X_4 W_{4,1}^i \\
Z_2^i &= X_1 W_{1,2}^i + X_2 W_{2,2}^i + X_3 W_{3,2}^i + X_4 W_{4,2}^i \\
&\vdots
\end{aligned}
$$

Z^i 'component' is the sum of weighted inputs to each neuron.

$$Z^i = XW^i \tag{1}$$

Apply activation function to 1

$$a^i = a(Z^i) \tag{2}$$

Propagate 2 to the output layer

$$Z^j = a^i W^j \tag{3}$$

$$\Longrightarrow \hat{y} = a^j = a^*(Z^j) \tag{4}$$

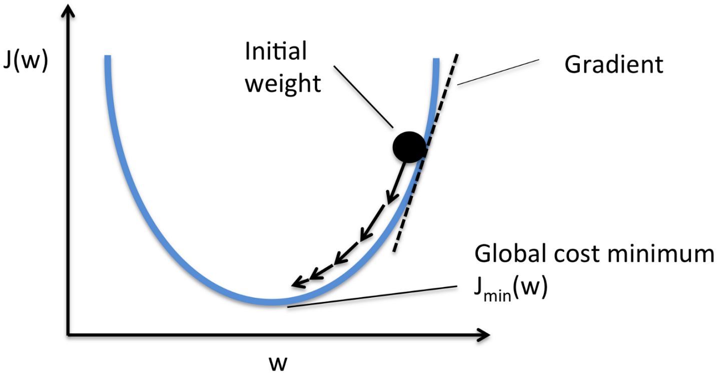
- Estimate weights that ensures the model fits the training data well.

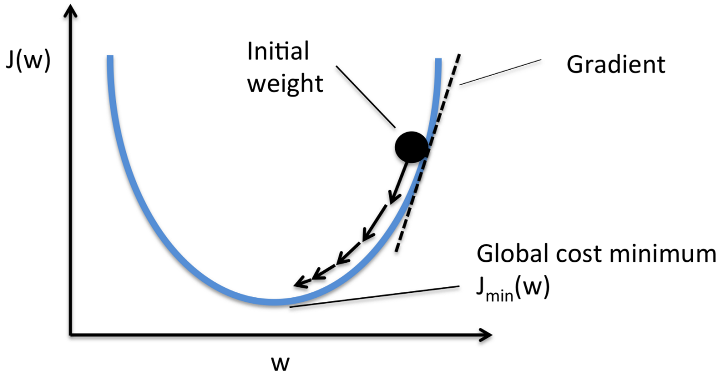
$$J = \sum \frac{1}{2}(y - \hat{y})^2 \tag{5}$$

- Estimate weights that ensures the model fits the training data well.

$$J = \sum \frac{1}{2}(y - \hat{y})^2 \tag{5}$$

$$J(W) = \frac{1}{2}\sum \left(y - a(a(XW^i)W^j)\right)^2 \tag{6}$$


J(w)
Initial weight
Gradient
Global cost minimum
$J_{min}(w)$
w



$$W_{t+1} = W_t - \gamma \Delta J(W_t)$$

Compute $\frac{\partial J}{\partial W^i}$ and $\frac{\partial J}{\partial W^j}$

$$\begin{aligned} \frac{\partial J}{\partial W^j} &\approx -(y-\hat{y})\frac{\partial \hat{y}}{\partial W^j} \\ \implies \frac{\partial J}{\partial W^j} &= -a^{iT}(y-\hat{y})a'(Z^j) \end{aligned}$$

$$\frac{\partial J}{\partial W^i} \approx -(y-\hat{y})\frac{\partial \hat{y}}{\partial W^i}$$
$$\implies \frac{\partial J}{\partial W^i} = -X^T(y-\hat{y})a'(Z^j)W^{jT}a'(Z^i)$$

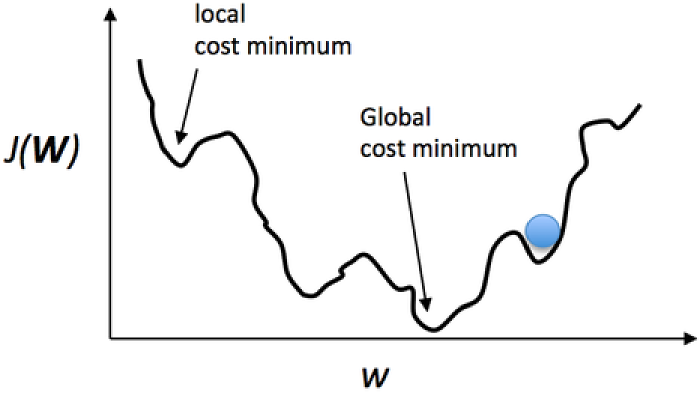
1. Starting values
    - Starting weights are random numbers near zero.
    - However, near weights collapses NN into approximately linear model.
    - Exactly zero weights leads to zero derivatives and perfect symmetry.
    - Large weights lead to poor results.

1. Starting values
    - Starting weights are random numbers near zero.
    - However, near weights collapses NN into approximately linear model.
    - Exactly zero weights leads to zero derivatives and perfect symmetry.
    - Large weights lead to poor results.
2. Overfitting and Stopping Criterion
    - Reduce training error to some predetermined threshold $\longrightarrow$ overffiting.
    - Regularization by *weight decay* (analogous to ridge regression)

$$J(W) = \sum \frac{1}{2}(y - \hat{y})^2/n + \frac{1}{2}\lambda\left(\sum W_1^2 + \sum W_2^2\right)$$

      $\lambda \geq 0$ - Is tuning parameter. Larger values of λ shrinks weights toward zero.
    - Cross-validation is used to estimate λ.

3. Convergence at the Local Minima



[1] Trevor, H., Robert, T., & JH, F. (2009). The elements of statistical learning: data mining, inference, and prediction. *Springer series in statistics*. Second Edition

[2] Tom M. Mitchell. (1997). Machine Learning *McGraw-Hill International Editions*.

[3] Internet sources (2019).