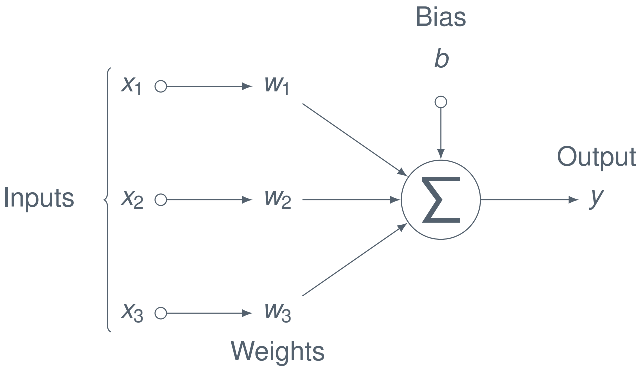
Neural Network

XuJingda
xjd15076083141@gmail.com

July 27, 2016

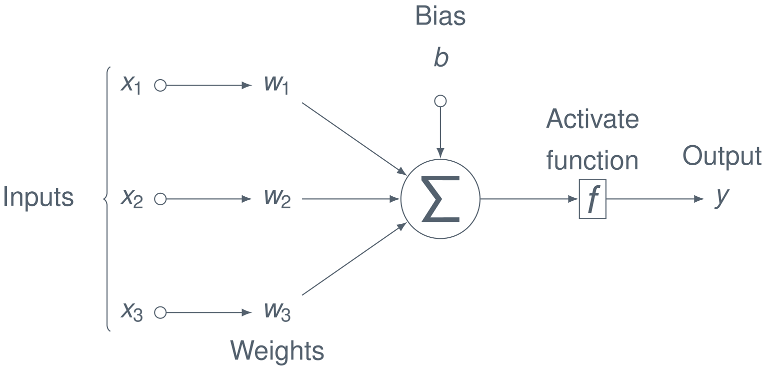
Content

Neuron



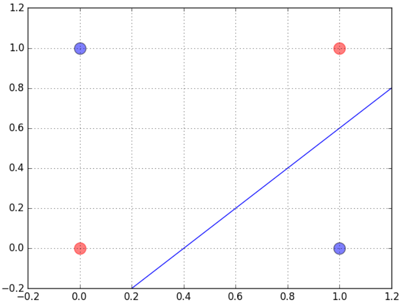
Perceptron

- Perceptron is a type of linear classifier.

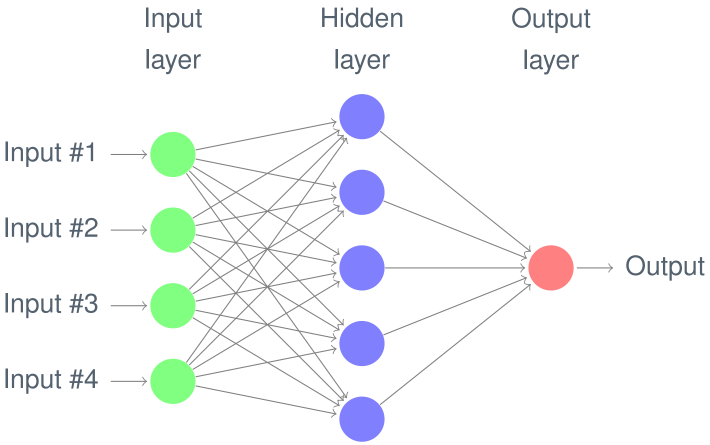


XOR Problem

- perceptron can not solve XOR problem



The structure of Neural Network



Loss function

- Regression problem

Mean Square Error Function

$J(\theta) = \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\|h_{W,b}(x^i) - y^i\|^2\right)\right]$

- Classification problem

Cross Entropy Function

$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=0}^{1}\left(I\{y^i = j\}\log P(y^i = j|x^i;\theta)\right)$

Optimization algorithm

- Gradient Descent (GD):compute the cost gradient based on the complete training set
- Stochastic Gradient Descent (SGD):
  - update the weights after each training sample
  - the path towards the global cost minimum is not 'direct' as in GD, but may go 'zig-zag'
- Mini-Batch Gradient Descent:update the model based on smaller groups of training samples

Backpropagation Algorithm

- BP is a method whitch calculates the gradient of a loss function with respect to all the weights in the network
- BP uses the chain rule to iteratively compute gradients for each layer

Formula Derivation

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m} J(W,b,x^i,y^i)\right] = \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\|h_{W,b}(x^i) - y^i\|^2\right)\right]$$

$$\frac{\partial}{\partial W_{i,j}^l} J(W,b) = \frac{\partial J(W,b)}{\partial a_i^{l+1}} \frac{\partial a_i^{l+1}}{\partial z_i^{l+1}} \frac{\partial z_i^{l+1}}{\partial w_{i,j}^l}$$

- Our goal is changed to compute: $\frac{\partial J(W,b)}{\partial a_i^{l+1}}$, $\frac{\partial a_i^{l+1}}{\partial z_i^{l+1}}$ and $\frac{\partial z_i^{l+1}}{\partial w_{i,j}^l}$

Formula Derivation

- we begin by calculating the gradient of output layer

$\frac{\partial}{\partial W_{i,j}^{n_l-1}} J(W, b)$

$$\frac{\partial J(W,b)}{\partial a_i^{n_l}} = \frac{\partial \left(\frac{1}{2} \sum_{j=1}^{s_{n_l}} (y_j - a_j^{n_l})^2 \right)}{\partial a_i^{n_l}} = a_i^{n_l} - y_i$$

$$\frac{\partial a_i^{n_l}}{\partial z_i^{n_l}} = f^{'}(z_i^{n_l}) = a_i^{n_l}(1 - a_i^{n_l})$$

$$\frac{\partial z_i^{n_l}}{\partial w_{ij}^{n_l-1}} = a_j^{n_l-1}$$

- $\frac{\partial J(W,b)}{\partial a_i^{l+1}} \frac{\partial a_i^{l+1}}{\partial z_i^{l+1}} \Rightarrow \delta^{l+1}$

Formula Derivation

- calculate the gradient of second last layer

$$\frac{\partial}{\partial W_{i,j}^{n_l-2}} J(W,b) = \frac{\partial J(W,b)}{\partial a_i^{n_l-1}} \frac{\partial a_i^{n_l-1}}{\partial z_i^{n_l-1}} \frac{\partial z_i^{n_l-1}}{\partial w_{i,j}^{n_l-2}}$$

$$\frac{\partial J(W,b)}{\partial a_i^{n_l-1}} = \sum_{j=1}^{s_{n_l}} (a_j^{n_l} - y_j) a_j^{n_l} (1 - a_j^{n_l}) W_{ji}^{n_l-1}$$

$$\frac{\partial a_i^{n_l-1}}{\partial z_i^{n_l-1}} = f'(z_i^{n_l-1}) = a_i^{n_l-1}(1 - a_i^{n_l-1})$$

$$\frac{\partial z_i^{n_l}}{\partial w_{ij}^{n_l-1}} = a_j^{n_l-2}$$

Formula Derivation

- we can find :

$$\delta_i^{n_l-1} = \left(\sum_{j=1}^{s_{n_l}} \delta_j^{n_l} W_{ji}^{n_l-1} \right) a_i^{n_l-1}(1 - a_i^{n_l-1})$$

- update weights

$$\frac{\partial}{\partial W_{i,j}^{l}} J(W, b) = \left(\sum_{j=1}^{s_{l+2}} \delta_j^{l+2} W_{ji}^{l+1} \right) a_i^{l+1}(1 - a_i^{l+1}) a_j^{l} = \delta_i^{l+1} a_j^{l}$$

Activation Function

sigmoid

$$f'(x) = f(x)(1 - f(x)) \in (0, 0.25)$$

tanh

$$f'(x) = (1 - f^2(x)) \in (0, 1)$$

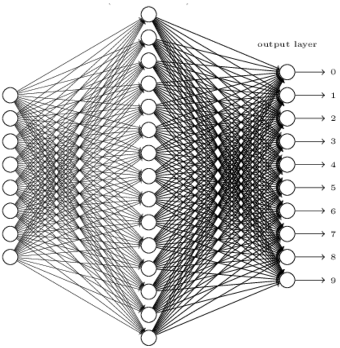
Relu

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

Neural Network

Talk is cheap ,show me your code.

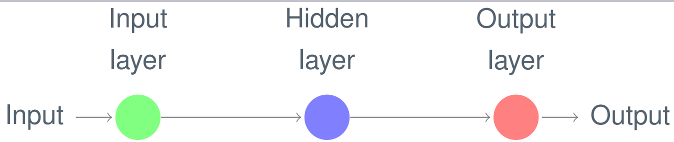
- Using neural nets to recognize handwritten digits(mnist).

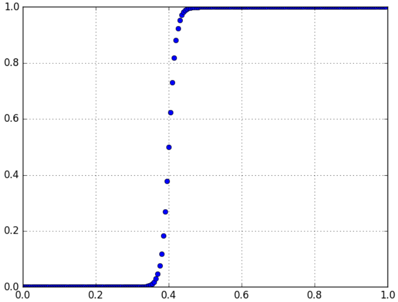


- sum 10000 right 9436 wrong 564

Neural Network

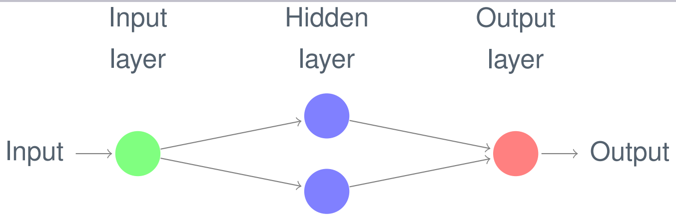
Talk is cheap ,show me your code.

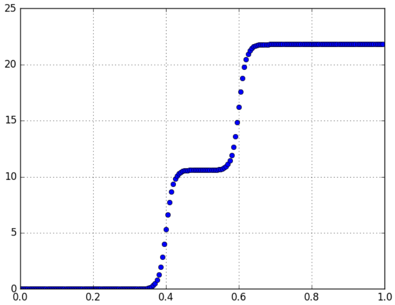




Neural Network

Talk is cheap ,show me your code.





- Overfit
- $y = y/(max(y) - min(y))$