

Machine Learning Formulae

Notations

m = Number of training examples.

n = Number of features.

$x^{(i)}$ = The features of the i_{th} training example, which is an $n + 1$ vector.

$y^{(i)}$ = The Target value of the i_{th} training example, which is a number.

$(x^{(i)}, y^{(i)})$ = The i_{th} training example.

$x_j^{(i)}$ = The j_{th} feature of the i_{th} training example, which is a number.

α = Learning rate

Multivariate Linear Regression

Hypothesis

$h_x(\theta) = \sum_{i=0}^n \theta_i x_i$ x_0 is always equal to 1.

Cost Function

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Gradient Descent

Repeat{

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$(j = 0, 1, 2, \dots, n)$, α stands for learning rate.

}

Calculate the $\frac{\partial}{\partial \theta_j} J(\theta)$, which means the partial derivative respect to θ_j :

Repeat{

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

}

Feature Scaling

Replace every x with:

$$\frac{x - \mu}{s}$$

μ stands for the average of x 's, and s stands for the standard deviation of x 's.

Normal Equation

Define the feature vector of the i_{th} training example:

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \dots \\ x_n^{(i)} \end{bmatrix} \quad x^{(i)} \text{ is an } n + 1 \text{ vector.}$$

Define **design matrix** X :

$$X = \begin{bmatrix} (x^{(1)})^T \\ (x^{(2)})^T \\ \dots \\ (x^{(m)})^T \end{bmatrix} \quad X \text{ is an } m \times (n + 1) \text{ matrix.}$$

Define the target value vector:

$$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(m)} \end{bmatrix} \quad y \text{ is an } m \times (n + 1) \text{ vector.}$$

Then, we can calculate θ :

$$\theta = (X^T X)^{-1} X^T y \quad \theta \text{ is an } n + 1 \text{ vector.}$$

Logistic Regression

Hypothesis

Define:

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_n \end{bmatrix} \quad x = \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix}$$

Hypothesis:

$$h_{\theta}(x) = g(\theta^T x)$$

$$g(z) = \frac{1}{1+e^{-z}} \quad (\text{sigmoid/logistic function})$$

which means:

$$h_{\theta}(x) = \frac{1}{1+e^{-\theta^T x}} \in (0, 1)$$

Cost Function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_{\theta}(x^{(i)}), y^{(i)}) = \begin{cases} -\log(1 - h_{\theta}(x)), & y = 0 \\ -\log(h_{\theta}(x)), & y = 1 \end{cases} \iff \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)}) = -y \log(h_{\theta}(x)) - (1 - y) \log(1 - h_{\theta}(x))$$

Gradient Descent

Repeat{

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

}

Multiclass Classification

Train a logistic regression classifier $h_{\theta}^{(i)}(x)$ for each class i to predict the probability that $y = i$. On a new input x , to make a prediction, pick the class i that maximizes $h_{\theta}^{(i)}(x)$.

Regularized Linear Regression

Cost Function

$$J(\theta) = \frac{1}{2m} [\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{i=1}^n \theta_j^2]$$

Gradient Descent

Repeat{

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha [\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j]$$

$$(j = 1, 2, 3, \dots, n)$$

}

Normal Equation

$$\theta = (X^T X + \lambda \begin{bmatrix} 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \dots & & & & & \\ 0 & 0 & 0 & 0 & \dots & 1 \end{bmatrix})^{-1} X^T y$$

Regularized Logistic Regression

Cost Function

$$J(\theta) = -\left[\frac{1}{m} \sum_{i=1}^m y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

Gradient Descent

Repeat{

$$\theta_0 := \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

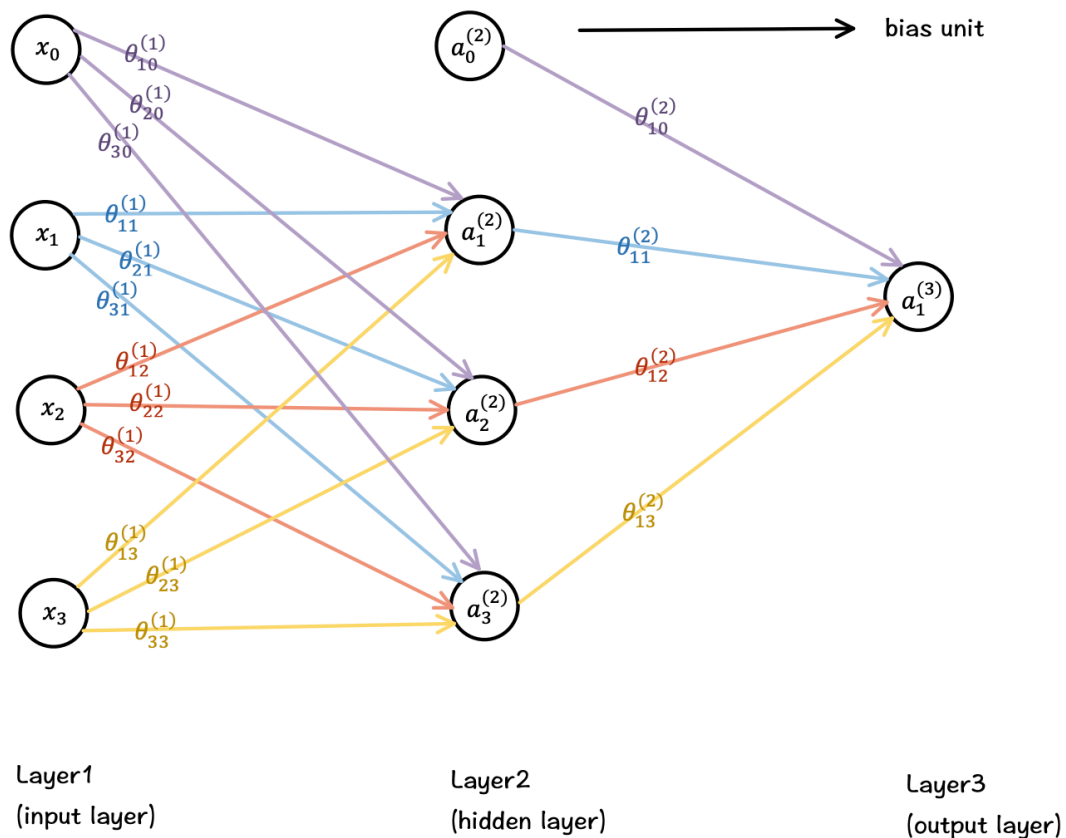
$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right]$$

($j = 1, 2, 3, \dots, n$)

}

Neural Network

Architecture



$a_i^{(j)}$ = activation of unit i in layer j

$\theta^{(j)}$ = matrix of weights controlling function mapping from layer j to layer $j + 1$

E.g.

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta^{(1)} = \begin{bmatrix} \theta_{10}^{(1)} & \theta_{11}^{(1)} & \theta_{12}^{(1)} & \theta_{13}^{(1)} \\ \theta_{20}^{(1)} & \theta_{21}^{(1)} & \theta_{22}^{(1)} & \theta_{23}^{(1)} \\ \theta_{30}^{(1)} & \theta_{31}^{(1)} & \theta_{32}^{(1)} & \theta_{33}^{(1)} \end{bmatrix} \in R^{3 \times 4}$$

$$a_1^{(2)} = g(\theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3) \text{ Define: } z_1^{(2)} = \theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3$$

$$a_2^{(2)} = g(\theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3) \text{ Define: } z_2^{(2)} = \theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3$$

$$a_3^{(2)} = g(\theta_{30}^{(1)} x_0 + \theta_{31}^{(1)} x_1 + \theta_{32}^{(1)} x_2 + \theta_{33}^{(1)} x_3) \text{ Define: } z_3^{(2)} = \theta_{30}^{(1)} x_0 + \theta_{31}^{(1)} x_1 + \theta_{32}^{(1)} x_2 + \theta_{33}^{(1)} x_3$$

$$\iff a^{(2)} = g(\theta^{(1)} X) \iff a^{(2)} = g(z^{(2)})$$

Cost Function

L = total number of layers in network

S_l = number of units(not counting bias unit) in lay l .

$h_\theta(x) \in R^K$ (which means there are K units in the output layer)

$(h_\theta(x))_i = i^{th}$ output

$$J(\theta) = -\frac{1}{m} [\sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log(h_\theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\theta(x^{(i)}))_k)] + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{S_l} \sum_{j=1}^{S_{l+1}} (\theta_{ji}^{(l)})^2$$

Back Propagation

Intuition: $\delta_j^{(l)}$ = error of node j in layer l

Suppose $L = 4$, that is, there are 4 layers in the network in total:

For each output unit($l = 4$):

$$\delta_j^{(4)} = a_j^{(4)} - y_j$$

Vectorization: $\delta^{(4)} = a^{(4)} - y$

For other layers:

$$\delta^{(3)} = (\theta^{(3)})^T \delta^{(4)} \cdot *g'(z^{(3)}) = a^{(3)} \cdot *(1 - a^{(3)})$$

$$\delta^{(2)} = (\theta^{(2)})^T \delta^{(3)} \cdot *g'(z^{(2)}) = a^{(2)} \cdot *(1 - a^{(2)})$$

...

Algorithm:

Training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

Set $\Delta_{ij}^{(l)} = 0$ (for all l, i, j)

For $i = 1$ to m

Set $a^{(1)} = x^{(i)}$

Perform forward propagation to compute $a^{(l)}$ for $l = 1, 2, 3, \dots, L$

Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$

Compute $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$

$$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$$

$$\text{Define } D_{ij}^{(l)} = \frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta) \iff D_{ij}^{(l)} = \begin{cases} \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \theta_{ij}^{(l)} & \text{if } j \neq 0 \\ \frac{1}{m} \Delta_{ij}^{(l)} & \text{if } j = 0 \end{cases}$$

$$\theta_{ij}^{(l)} := \theta_{ij}^{(l)} - \alpha D_{ij}^{(l)}$$