Logistic Regression notes

Edward Hoang

December 23, 2021

Abstract

This is a Course note from week 2 of Neural Networks and Deep Learning Course

1 Neural Network notations

General comments:

- Superscript (i) will denote the i^{th} training example while superscript [l] will denote the l^{th} layer

Sizes:

- m: number of examples in the dataset.

- n_x: input size

- n_y: output size (or number of classes)

- $n_h^{[l]}$: number of hidden unit of the l^{th} layer

In a loop, it is possible to denote $n_x = n_h^{[0]}$ and $n_y = n_j^{[number\ of\ layers\ +\ 1]}$

- L: number of layers in the network.

Objects:

- $X \in R^{n_x \times m}$ is the input matrix

- $x^{(i)} \in R^{n_x}$ is the i^{th} example represented as a column vector

- $Y \in R^{n_y \times m}$ is the label matrix

- $y^i \in R^{n_y}$ is the output label for the i^{th} example

- $W^l \in R^{number\ of\ units\ in\ next\ layer\ \times\ numbers\ of\ units\ in\ the\ previous\ layer}$ is the weight matrix, superscript [l] indicates the layer

- $b^{[l]} \in R^{number\ of\ units\ in\ next\ layer}$ is the bias vector in the l^{th} layer

- $\hat{y} \in R^{n_y}$ is the predicted output vector. It can also be denoted $a^[L]$ where L is the number of layers in the network.

Common forward propagation equation examples

$a = g^{[l]}(W_x x^{(i)} + b_1) = g^{[l]}(z_1)$ where $g^{[l]}$ denotes the l^{th} layer activation function

$\hat{y}^{(i)} = softmax(W_h h + b2)$

- General Activation Formula: $a_j^{[l]} = g^{[l]}(\sum_k w_{jk}^{[l-1]} a_k^{[l-1]} + b_k^{[l]}) = g^{[l]}(z_j^{[l]})$

- $J(x, W, b, y)$ or $J(\hat{y}, y)$ denote the cost function.

Examples of cost function:

- $J_{CE}(\hat{y}, y) = -\sum_{i=0}^{m} y(i) log(\hat{y}^{(i)})$

- $J_1(\hat{y}, y) = \sum_{i=0}^{m} |y^{(i)} - \hat{y}^{(i)}|$

2 Logistic Regression notes

Personal notes:

- **Form**: $\sigma(z) = \frac{1}{1+e^{-z}}$ with $z = W^T x + b$

- **Dataset:** $\{(x^{(1)}, y^{(1)}),\ (x^{(2)}, y^{(2)}),\ ...,\ (x^{(m)}, y^{(m)})\}$

- **Want** $\hat{y}^{(i)} \approx y^{(i)}$

- **Need:**

+ Lost function to evaluate how well the algorithm's doing for a single training example

+ Form: $l(\hat{y},\ y) \ = \ -(y \log \hat{y} + (1-y) \log(1-\hat{y}))$

+ Cost function to evaluate how well the algorithm's doing for the entire training set

+ Form: $Cost \ = \ \frac{1}{m} \sum_{i=1}^{m} l(\hat{y},\ y)$

$or = -\frac{1}{m} \sum_{i=1}^{m} (y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)}))$

+ Gradient Descent to take the model closer to the global minimum of the function

* Learning rate: how big a step we take on each iteration.

* Form:

Repeat {

$w := w - \alpha \frac{\partial J(w,b)}{\partial w}$

$b := b - \alpha \frac{\partial J(w,b)}{\partial b}$

}

Note: if J is a function of 2 or more variables, we use the symbol ∂ instead of "d" to denote the "partial derivative".