Basics of Neural Network Programming

Logistic Regression

deeplearning.ai

Logistic Regression

Given X, want $\hat{y} = P(y=1 \mid X)$

$X \in \mathbb{R}^{n_x}$ → say the pixel intensities in RGB

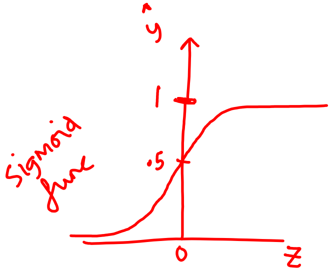
Params : $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$

output $\hat{y}$

↳ one way is to make $\hat{y} = w^T x + b$

not good, why? we want $\hat{y} \in [0,1]$

$\Rightarrow \hat{y} = \sigma(w^T X + b)$



$\Rightarrow \sigma(z) = \frac{1}{1+e^{-z}}$ $\Rightarrow$ If $z \to$ large +ve, $e^{-z} = 0$

$\Rightarrow \sigma(z) = 1$

If z is large -ve $\to \sigma(z) = \frac{1}{1+\infty} = 0$

If z is 2 $\Rightarrow \sigma(2) = \frac{1}{1+e^{-2}}$ = small +ve number $= \frac{e^2}{1+e^2}$

$X_0 = 1$ $\quad X \in \mathbb{R}^{n_x+1}$, we need to accommodate for X_0, hence $n_x + 1$

$\hat{y} = \sigma(\theta^T X)$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_x} \end{bmatrix} \quad \theta_0 \to b, \quad \theta_1 \ldots \theta_{n_x} \to w$$

$\Rightarrow \theta_0 = b$

$\theta_1 \ldots \theta_{n_x} = w$

Sometimes notations dont differentiate b/w θ_0 and $\theta_1 \ldots \theta_n$ but we will!

Basics of Neural Network Programming

Logistic Regression cost function

deeplearning.ai

Logistic Regression cost function

$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, where $\sigma(z^{(i)}) = \frac{1}{1+e^{-z^{(i)}}}$

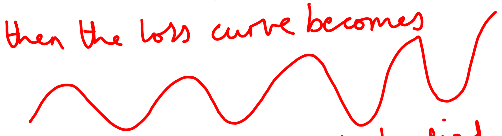
i = i^{th} training example

Given $\{(x^{(1)}, y^{(1)}),\ldots,(x^{(m)}, y^{(m)})\}$, want $\hat{y}^{(i)} \approx y^{(i)}$.

Loss (error) function:

one way is to use MSE
↳ $L(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$

Bad, because then the loss curve becomes non convex



⇒ difficult to use gradient descent to find global optima, can only use local optima

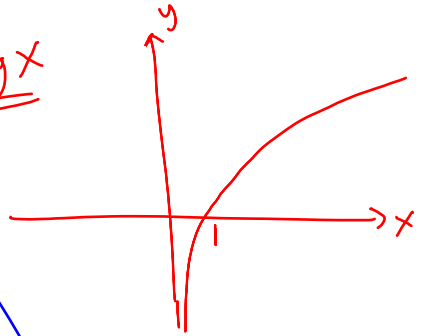
Loss func was over 1 example

Cost func → over entire training set

$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$

objective is to find params w & b that minimize the cost func

log x



So we use

$L(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y}))$ ☆

Why is this a good loss func for logit?

We know y can only take values (0,1)

- If $y=1$, our objective is to make $\hat{y}=1$ with our model, so we can be as close to the Actual answer ($y=1$), same with $y=0$, If $y=0$, we want our model to predict $\hat{y}=0$, so we can be as close to the Ans as possible

⇒ we need a loss func that is minimal, when we predict the correct Ans & penalizes us when we predict the wrong Ans

If $y=1$ & $\hat{y}=1$, then loss $= -(1 \cdot \log 1 + 0) = 0$
$y=0$ & $\hat{y}=0$, then loss $= -(0 + 1 \cdot \log 1) = 0$
less penalization when we are correct

If $y=1$ & $\hat{y}=0$, then loss $= -(1 \cdot \log 0 + 0) = \infty$
If $y=0$ & $\hat{y}=1$, then loss $= -(0 + 1 \cdot \log 0) = \infty$
more penalization when we are incorrect

Andrew Ng