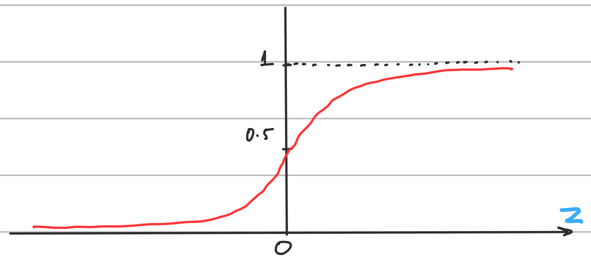
Week 3

○ Classification with Logistic Regression.

We use Sigmoid Function to classify values as 0 or 1



We need an equation which results value in the range 0 to 1, hence we use sigmoid function

$$\boxed{g(z) = \frac{1}{1+e^{-z}}}$$

□ Logistic Regression Model

$$f_{\vec{w},b}(\vec{x}) = z = \vec{w}\cdot\vec{x} + b$$

As,

$$g(z) = \frac{1}{1+e^{-z}}$$

$$f_{\vec{w},b}(\vec{x}) = g(\vec{w}\cdot\vec{x}+b)$$

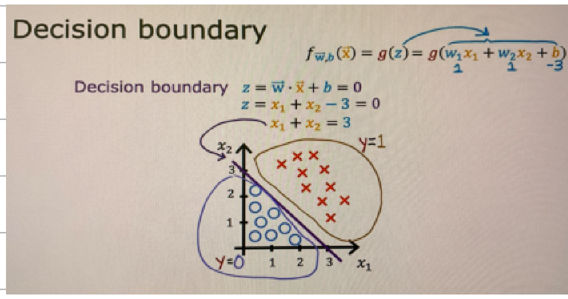
$$\boxed{f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}}$$

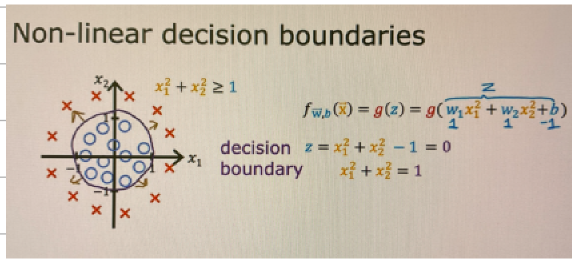
- It's also represented as,

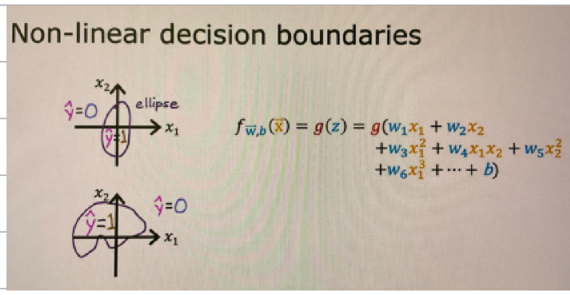
$$f_{\vec{w},b}(\vec{x}) = P(y=1 \mid \vec{x};\vec{w},b)$$

Which means, what is probability that y is 1 given input $\vec{x}$ and parameters $\vec{w}$, b

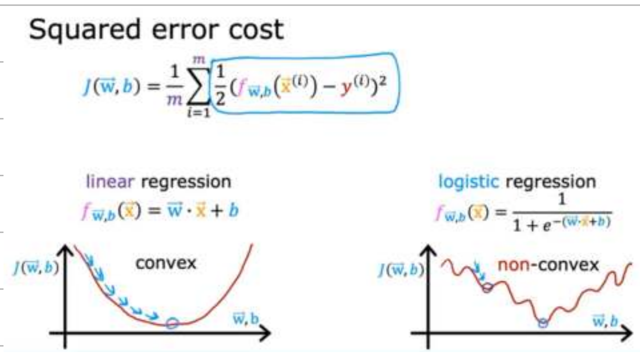
○ Decision Boundary







○ Why can't we use squared error cost function for Classification?



- As seen in the image above, Squared error cost function would create a non-convex curve with multiple local minima.

○ Logistic Loss Function

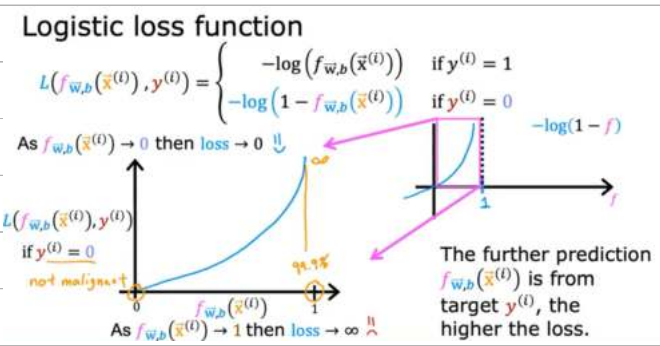
- We find loss for each example & then get the average of all the losses

$$J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m} \underbrace{L\left(f_{\vec{w},b}(x^{(i)}),\ y^{(i)}\right)}_{\text{Loss}}$$

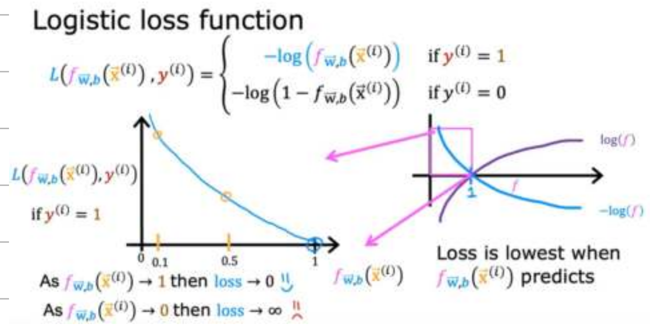
Here Loss Equals,

$$\boxed{L = \begin{cases} -\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 0 \end{cases}}$$

○ Logistic Loss for $y^{(i)} = 0$



○ Logistic Loss for $y^{(i)} = 1$



○ Simplified Loss Function

$$L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right) = \begin{cases} -\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)} = 0 \end{cases}$$

$$\boxed{L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right) = -y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) - (1-y^{(i)})\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right)}$$

- This is a simple extension of the given conditional value which further helps in simplifying the cost function.

○ Simplified Cost Function for Logistic Regression.

loss

$$L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right) = -y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) - (1-y^{(i)})\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right)$$

cost

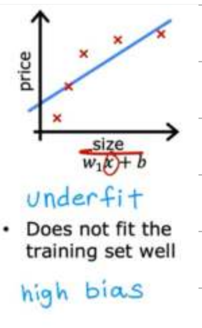
$$J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}\left[L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right)\right]$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1-y^{(i)})\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right)\right]$$

o Fitting issue in Linear Regression.

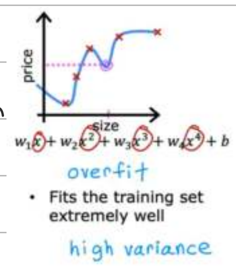
A. Underfitting :-

This happens when the model has few features.



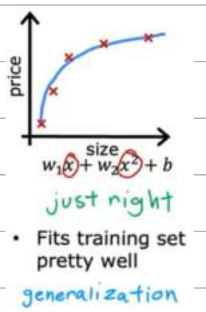
B. Overfitting :-

This happens when the model has a lot of features



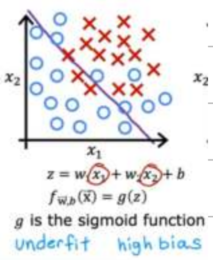
C. Generalization :-

This happens when the model has the perfect amount of features.

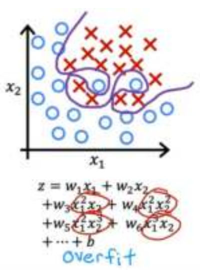


o Fitting issue in Classification

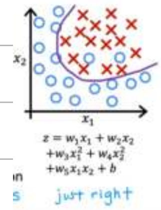
– Underfitting ⟶



– Overfitting ⟶



– Generalized Fitting ⟶



o Ways to handle Overfitting

- Collect more data. If that's not possible then
- Improve feature selection. But it means you need to sacrifice data of few features. Hence you can use

★ Regularization :- Reduce size of parameters.

o Regularization in Regression.

– We penalise all the features by multiplying it with 'λ'.

$$\frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2$$

Here 2m is used so that regularization & cost function is scaled equally

Cost function after Regularization

$$\boxed{J(\vec{w},b) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2}$$

Regularized Linear Regression & Gradient Descent

Regularized cost function is given as :-

$$\min_{\vec{w},b} J(\vec{w},b) = \min_{\vec{w},b}\left[\underbrace{\frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2}_{\text{Squared Error cost function}} + \underbrace{\frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2}_{\text{Regularization}}\right]$$

Gradient Descent on Regularized cost function :-

repeat {

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w},b) \qquad = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)x_j^{(i)} + \frac{\lambda}{m} w_j$$

$j = 1, \ldots, n$

$$b = b - \alpha \frac{\partial}{\partial b} J(\vec{w},b) \qquad = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)$$

don't have to regularize b

} simultaneous update

Let's expand w_j :-

$$w_j = w_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left[\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)x_j^{(i)}\right] + \frac{\lambda}{m} w_j\right]$$

$$= 1 w_j - \alpha \frac{\lambda}{m} w_j - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}(\vec{x}^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

$$= w_j\left(1 - \alpha\frac{\lambda}{m}\right) - \text{usual update}$$

Let's assume $\alpha = 0.01$, $\lambda = 2$, $m = 50$

$$\therefore \quad w_j\left(1 - 0.01 \times \frac{2}{50}\right) = w_j(1 - 0.0002) = w_j(0.9998)$$

Hence regularization basically makes sure that we slowly reduce w_j and reduce overfitting.

★ Derivation for $\frac{\partial}{\partial w_j} J(\vec{w}, b)$

$$\frac{\partial}{\partial w_j} J(\vec{w}, b) = \frac{\partial}{\partial w_j}\left[\frac{1}{2m}\sum_{i=1}^{m}\big(\underbrace{f(\vec{x}^{(i)})}_{\vec{w}\cdot\vec{x}^{(i)}+b} - y^{(i)}\big)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2\right]$$

$$= \frac{1}{\cancel{2}m}\sum_{i=1}^{m}\left[(\vec{w}\cdot\vec{x}^{(i)} + b - y^{(i)})\,\cancel{2}x_j^{(i)}\right] + \frac{\lambda}{\cancel{2}m}\cancel{2}w_j \quad \text{No } \sum_{j=1}^{n}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left[(\underbrace{\vec{w}\cdot\vec{x}^{(i)} + b}_{f(\vec{x})} - y^{(i)})\,x_j^{(i)}\right] + \frac{\lambda}{m}w_j$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left[(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})x_j^{(i)}\right] + \frac{\lambda}{m}w_j$$

○ Regularized Logistic Regression & Gradient Descent.

– we do the same thing we did for linear regression

★ Cost function looks like this:-

$$\min_{\vec{w},b} J(\vec{w}, b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1-y^{(i)})\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

★ In the same way, updated gradient descent looks like this:-

Gradient descent

repeat {

$$w_j = w_j - \alpha\frac{\partial}{\partial w_j}J(\vec{w}, b) \quad j=1\ldots n$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left[(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})\,x_j^{(i)}\right] + \frac{\lambda}{m}w_j$$

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

$$= \frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})$$

}

Looks same as for linear regression!

logistic regression

don't have to re