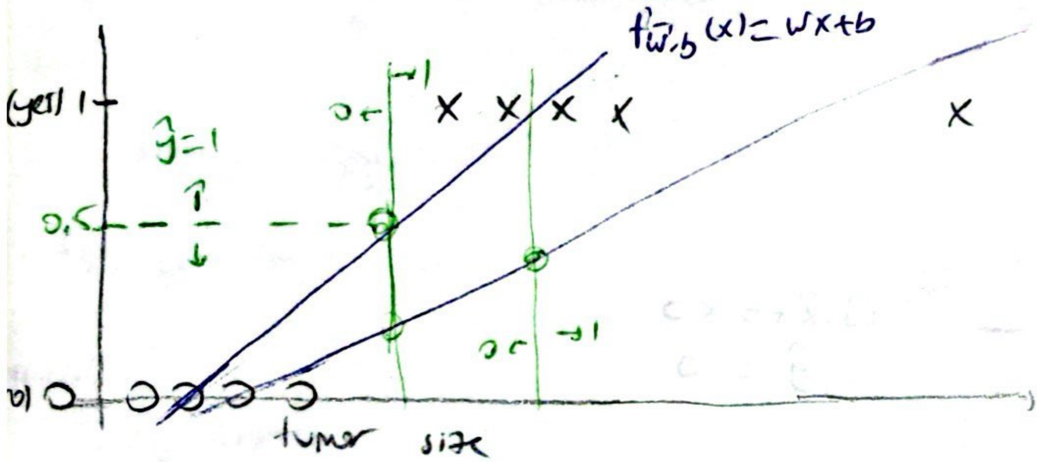
* Linear regression is not a good idea algorithm for classification problems. → logistic regression

Classification

	Answer "y"	
• Is this email spam?	no (0)	yes (1)
• Is the tumor malignant		

- y can only be one of two values. → binary classification

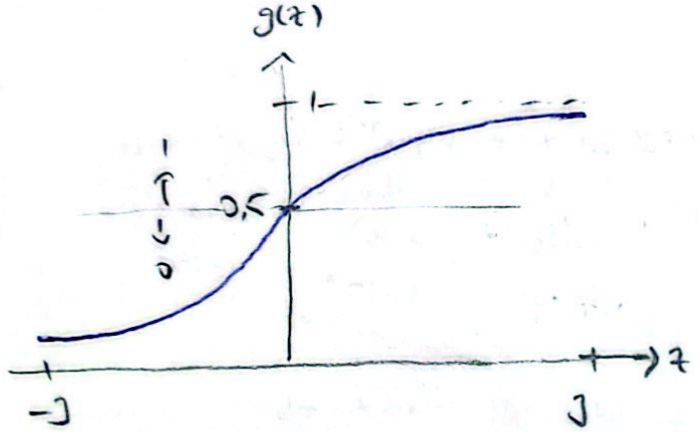


- if $f_{w,b}(x) < 0.5 \rightarrow \hat{y}=0$
- if $f_{w,b}(x) \geq 0.5 \rightarrow \hat{y}=1$

→ linear regression is not good idea for this classification

Logistic Regression

Want outputs between 0 and 1



- sigmoid function
- logistic function

$$\boxed{g(z) = \frac{1}{1+e^{-z}}}, \quad 0 < g(z) < 1$$

$$f_{\vec{w},b}(\vec{x}) = g(\vec{w}\cdot\vec{x}+b) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}} \Rightarrow \text{logistic regression}$$

- "probability" that class is 1

x : tumor size

y: 0 (not malignant)
 1 (malignant)

$f_{\vec{w},b}(\vec{x}) = 0.7$

70% chance that y is 1

30% chance that y is 0

* Is $f_{\vec{w},b}(\vec{x}) \geq 0.5$

yes: $\hat{y} = 1$, no: $\hat{y} = 0$

* When is $f_{\vec{w},b}(\vec{x}) \geq 0.5$

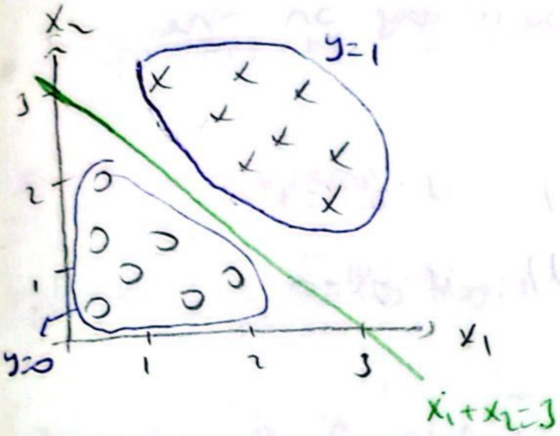
$g(z) \geq 0.5$

$z \geq 0$

$\vec{w}\cdot\vec{x}+b \geq 0$ — $\vec{w}\cdot\vec{x}+b < 0$

$\hat{y} = 1$ $\hat{y} = 0$

Decision boundary:



$$f_{\vec{w},b}(\vec{x}) = g(z) = g(\underset{1}{w_1}x_1 + \underset{1}{w_2}x_2 + \underset{-3}{b})$$

$$z = \vec{w}\cdot\vec{x} + b = 0$$

$$z = x_1 + x_2 - 3 = 0$$

$\boxed{x_1 + x_2 = 3}$ → decision boundary

* Logistic regression can learn to fit pretty complex data.
* The decision boundary for logistic regression will always be linear, will always be a straight line

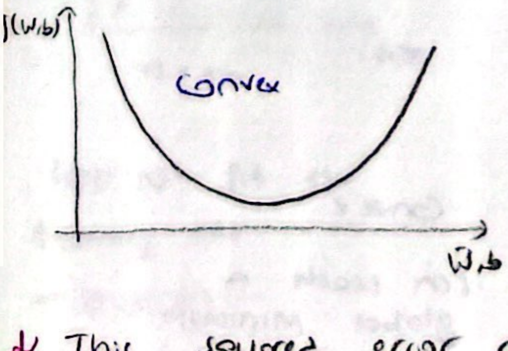
Cost Func for Logistic Regression

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}} \qquad \text{target } y \text{ is } 0 \text{ or } 1$$

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

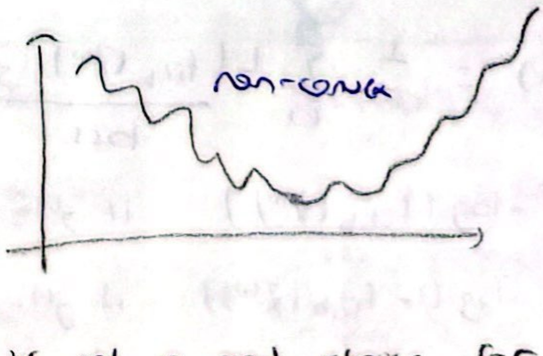
linear regression

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



logistic regression

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



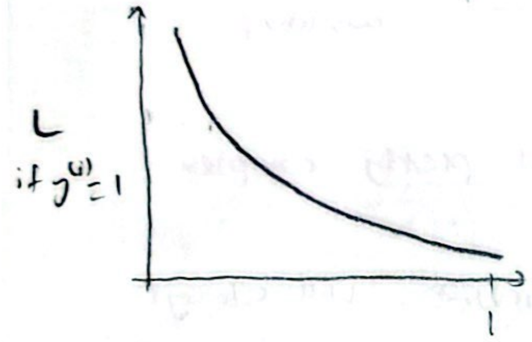
* This squared error cost func is not a good choice for logistic regression → there are lots of local minima, non-convex

There will be different cost func

Logistic loss function :

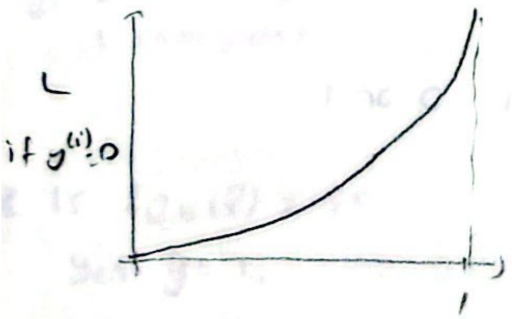
The loss function measures how well you're doing on one training example

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



- $f_{\vec{w},b}(x^{(i)}) \to 1$ then loss $\to 0$
- $f_{\vec{w},b}(\vec{x}^{(i)}) \to 0$ then loss $\to \infty$

* Loss is lowest when $f_{\vec{w},b}(\vec{x}^{(i)})$ predicts close to true label $y^{(i)}$



- $f_{\vec{w},b}(\vec{x}) \to 0$ then loss 0
- $f_{\vec{w},b}(\vec{x}^{(i)}) \to 1$ then loss ∞

* The further prediction $f_{\vec{w},b}(\vec{x}^{(i)})$ is from target $y^{(i)}$, the higher the loss

cost

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

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

- can reach a global minimum

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

y can only be 0 or 1

Gradient descent

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

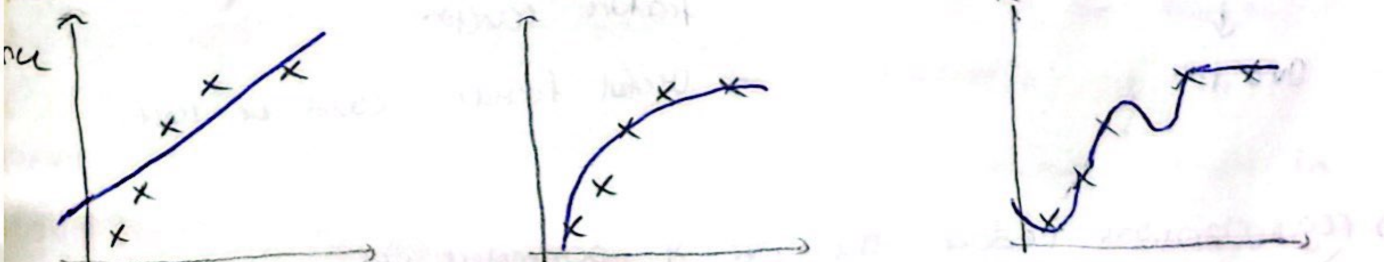
- $w_j = w_j - a.\underbrace{\frac{\partial}{\partial w_j}J(\vec{w},b)}$ $\rightarrow \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)}).x_j^{(i)}$
- $b = b - a.\underbrace{\frac{\partial}{\partial b}J(\vec{w},b)}$ $\rightarrow \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})$

simultaneously update

⇒ Same concepts :
- monitor gradient descent (learning curve)
- vectorized implementation.
- Feature scaling

The Problem of Overfitting

Regression



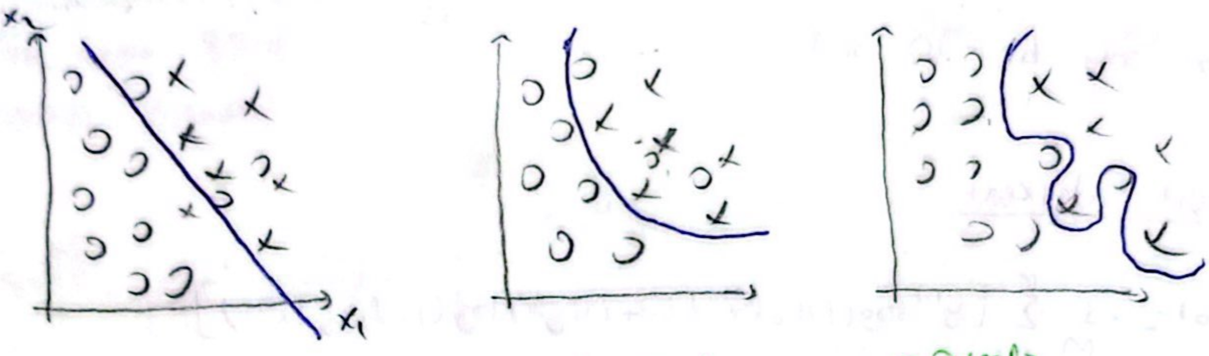
w_1x+b | $w_1x+w_2x^2+b$ | $w_1x+w_2x^2+w_3x^3+w_4x^4+b$

- Does not fit the training set
  - Underfit
  - high bias
- Fits training set pretty well
  - Just right
  - generalization
- Fits the training set extremely well
  - Overfit
  - high variance

- Machine learning goal → a model that has neither high bias nor high variance

Scanned with CamScanner

Classification



- underfit - just right - overfit

- Its model despite doing very well on the training set, doesn't look like it'll generalize well to new examples.

Addressing overfitting

If there is overfit prediction model

→ Collect more training data, learning algorithm will learn to fit a function that is less wiggly.

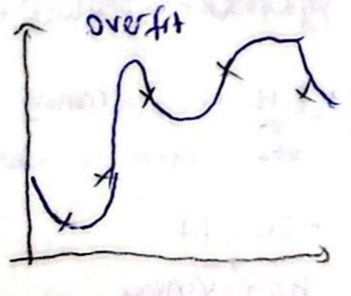
* more data isn't always an option.

all features
+
insufficient data
↓
Overfit

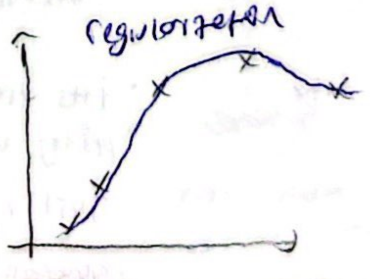
selected features
↓
Just right - feature selection

* Useful features could be lost

→ regularizations reduce the size of parameters w_j



$f(x) = 28x + 39x^3 + 100 ...$



$f(x) = 13x + 0.00001 4x^3 - 0.0001x^4 + 10$

* regularization: • keep all of your features

• but this just prevents the features from having an overly large effect.

• sometimes can cause overfitting

Addressing overfitting

Options
1) collect more data

2) Select features
- feature selection

3) Reduce size of parameters - "Regularization"

Cost function with regularization

• $w_1x + w_2x^2 + b$

• $w_1x + w_2x^2 + \cancel{w_3x^3} + \cancel{w_4x^4} + 4$

make w_3, w_4 very small (≈ 0)

$$\min_{\vec{w}, b} \frac{1}{2m} \sum_{i=1}^{m} (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})^2 + 1000 \underset{0{,}001}{\cancel{w_3^2}} + 1000 \underset{0{,}001}{\cancel{w_4^2}}$$

* simpler model → less likely to overfit

* regularization implemented → to penalize all of the features or more precisely

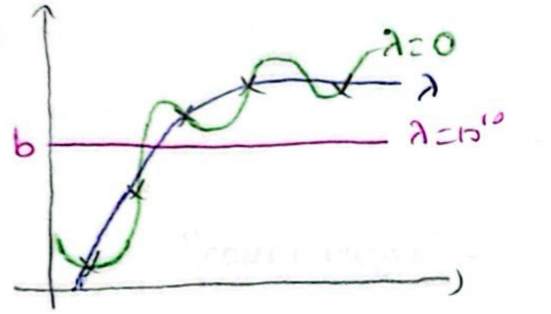
and its possible to show that this will usually result in fitting a smoother simpler, less weekly function that's less prone to overfitting.

$$J(\vec{w}, b) = \underbrace{\frac{1}{2m} \sum_{i=1}^{m} (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})^2}_{\text{mean squared error}} + \underbrace{\frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2}_{\text{regularization term}}$$

$\min J(\vec{w}, b)$ → The algorithm also tries to keep w_j small, which will tend to reduce overfitting.

$$\min_{\vec{w},b} J(\vec{w},b) = \min_{\vec{w},b} \underbrace{\frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2}_{\text{fit data}} + \underbrace{\frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2}_{\text{keep } w_j \text{ small}}$$

λ balanced both goals



λ: balanced

- minimizing the mean squared error
- keeping parameters small

Regularized linear regression

implementing gradient descent

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

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

simultaneous update

Regularized logistic regression

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

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

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

not same linear