Stanford ML

WEEK 3:

Classification: Logistic Regression

- Email: Spam / Not Spam?
- Online Transactions: Fraudulent (Y/N?)
- Tumour: Malignant / Benign?

OUTPUT

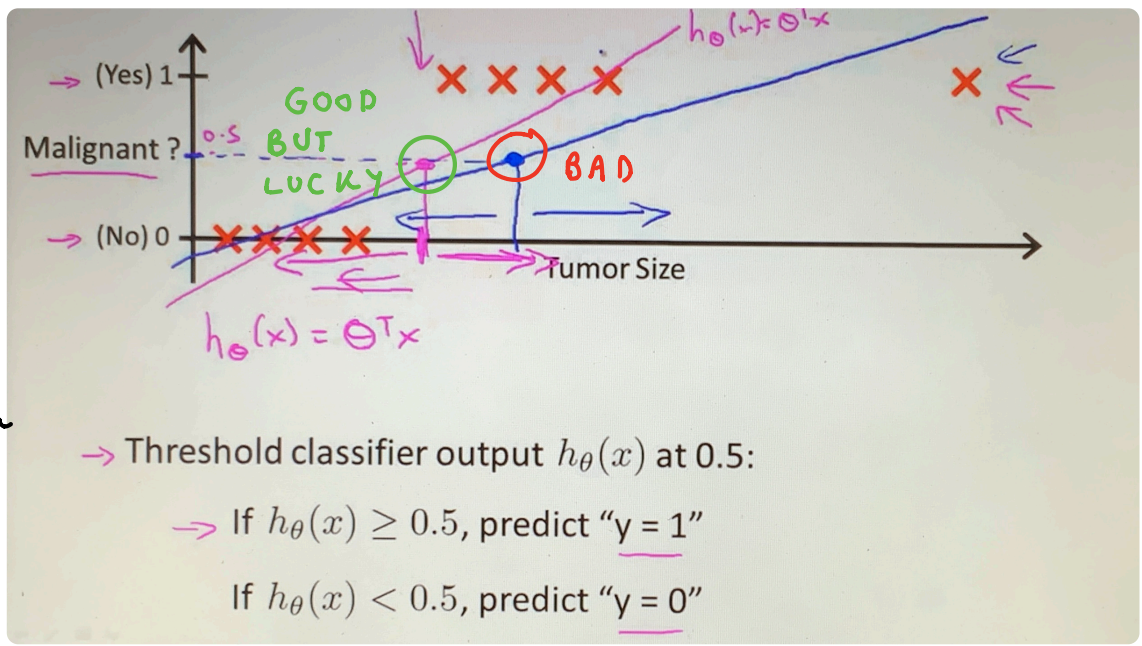
$y \in \{0, 1\}$

0: "Negative Class"

1: "Positive Class"

- Of course, multiclass classification is possible and common

- Example prob. with linear regression and justifies need for classification algorithm



- For linear regression, we saw the hypothesis must take the values

$h_\theta(x)$ can be > 1 or < 0

LOGISTIC REGRESSION

$$0 \leq h_\theta(x) \leq 1$$

LIN. REG.

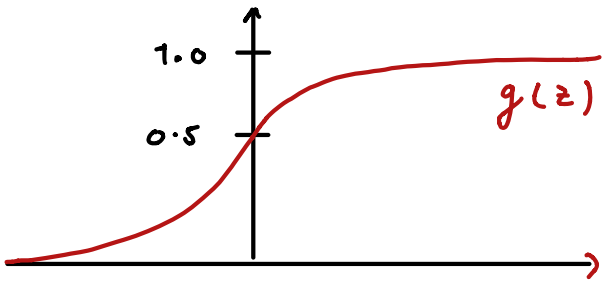
$$h_\theta(x) = \theta^T x$$

LOG. REG.

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

where $g(z)$ is the SIGMOID function

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



$$\therefore \boxed{h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}}$$

Interpretation of Hypothesis Output

- $h_\theta(x)$ = estimated probability that $y = 1$ on input x

Ex:

if $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumor Size} \end{bmatrix}$ and $h_\theta(x) = 0.7$

$\therefore$ 70% chance of being malignant

$$\Rightarrow h_\theta(x) = P(y = 1 \mid x; \theta)$$

"probability that $y = 1$, given x, parameterized by θ"

$$\therefore P(y = 0 \mid x; \theta) = 1 - P(y = 1 \mid x; \theta)$$

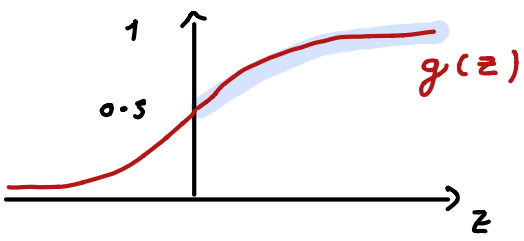
for BINARY CLASSIFICATION

Desecision Boundary

- At what value of $h_\theta(x)$ do we decide when the "$y = 1$"

Ex:

- $h_\theta(x) = g(\theta^T x)$
- $g(z) = 1 / 1 + e^{-z}$



- Suppose predict $y = 1$ if $h_\theta(x) \geq 0.5$
- Suppose predict $y = 0$ if $h_\theta(x) < 0.5$

* By analogous argument

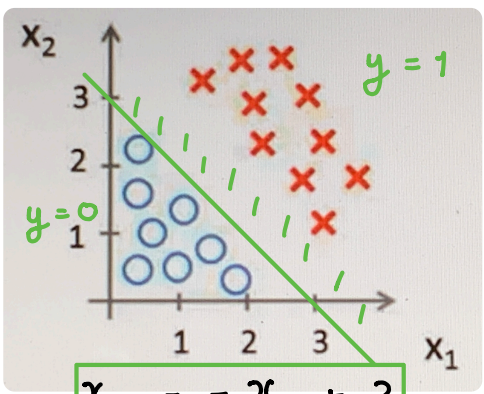
$h_\theta(x) = g(\theta^T x) < 0.5$ when $\theta^T x < 0$ so predict $y = 0$

* $g(z) \geq 0.5$ when $z \geq 0$

* $h_\theta(x) = g(\theta^T x) \geq 0.5$ whenever $\underbrace{\theta^T x}_{z} \geq 0$ predict $y = 1$

Ex: Constructing the Desicion Boundary

TRAINING SET



OR

$x_1 + x_2 \geq 3$

→ $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

" -3 " 1 " 1

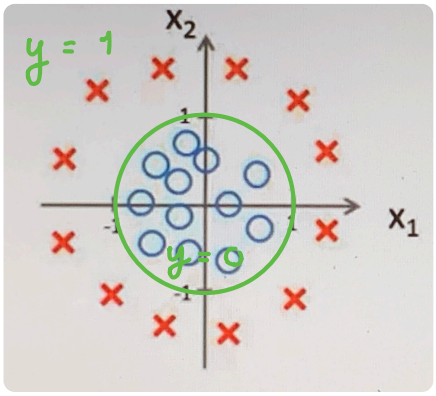
→ $\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$ chosen by some optimization procedure

∴ Predict "$y = 1$" if:

$\underbrace{-3 + x_1 + x_2}_{\theta^T x} \geq 0$

this is the DESCISION BOUNDARY

Non - Linear Decision Boundary



- In the same way that we can add higher orders of original features to conduct polynomial regression

- The same can be done for non-linear decision boundaries

EQN. OF CIRCLE

We

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

We

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

$\therefore$ predict $y = 1$ if

$$-1 + x_1^2 + x_2^2$$

$$\boxed{x_1^2 + x_2^2 \geq 1}$$

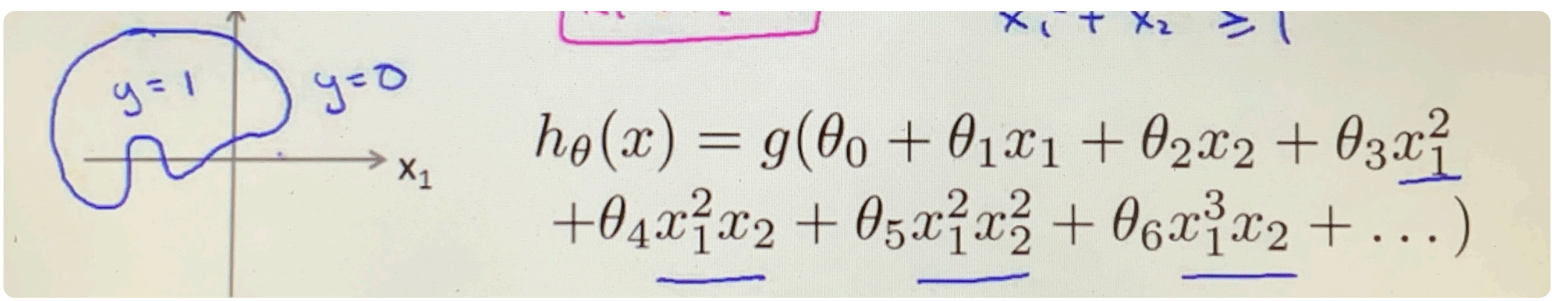
CIRCLE

$(0, 0)$

$r^2 = 1$

NOTE:

- The decision boundary is a PROPERTY OF THE HYPOTHESIS



$$\theta^T x \geq 0 \implies y = 1$$

$$\theta^T x < 0 \implies y = 0$$

Cost Function

- **Set up**

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

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad x_0 = 1, \; y \in \{0, 1\}$$

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

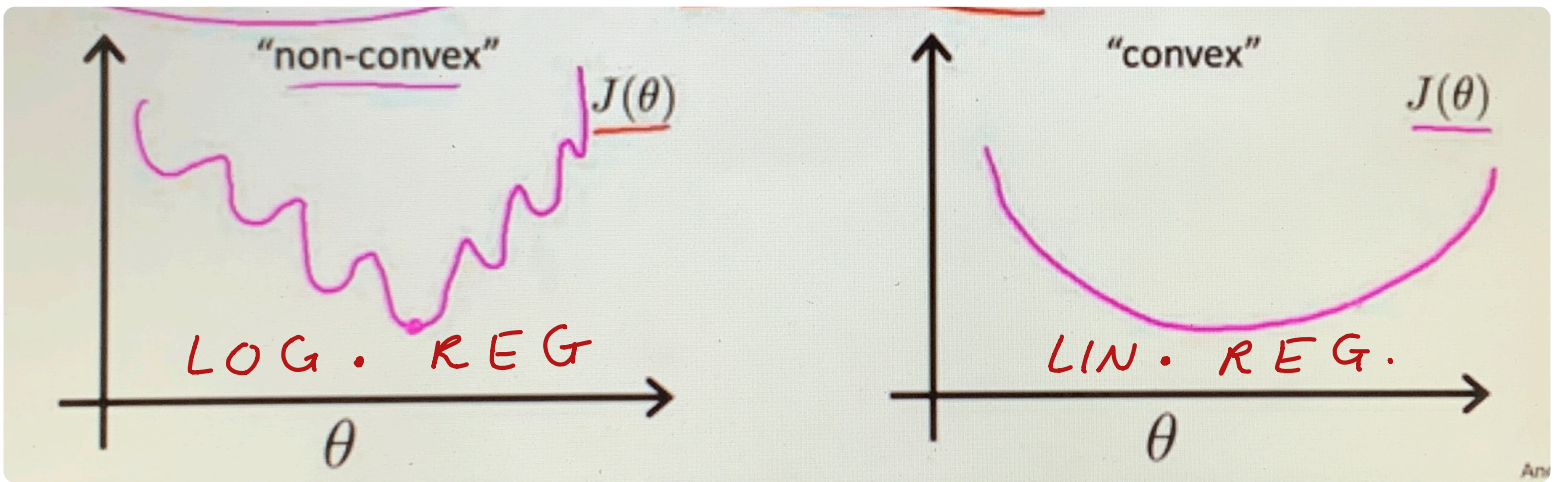
- Need to choose parameters θ
- Linear Regression

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left[h_\theta(x^{(i)}) - y^{(i)} \right]^2 = \frac{1}{m} \text{cost}(h_\theta(x), y)$$

$$\therefore \text{Cost}(h_\theta(x), y) = \frac{1}{2} \left[h_\theta(x) - y \right]^2$$

$$\forall \; i = 1, \dots, m$$

↳ For LOGISTIC REGRESSION where $h_\theta(x) = g(\theta^T x)$

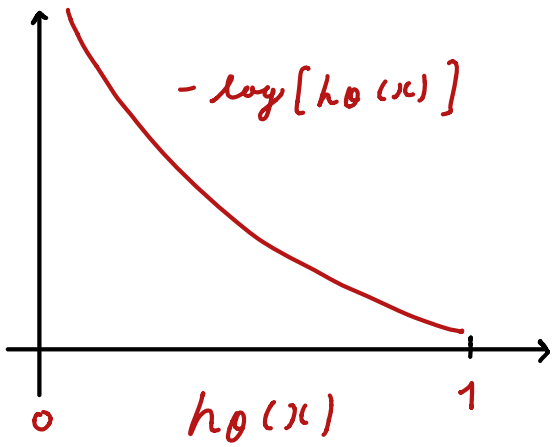


- Using the SQ. ERR cost function for for LOG REG creates a "non-convex" $J(\theta)$ meaning gradient descent CANNOT be applied.

→ LOG REG Cost Function

$$\text{Cost}[h_\theta(x), y] = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

$y = 1$

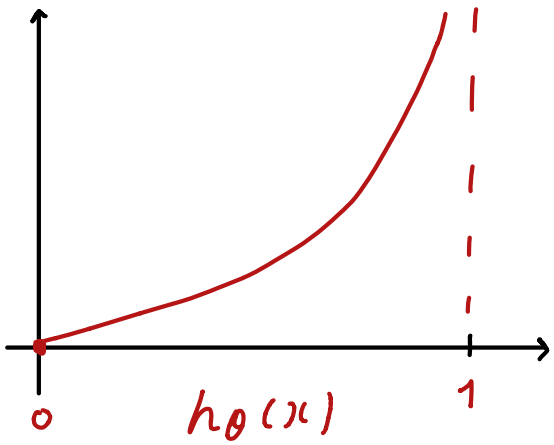


- Cost = 0 if $y = 1$, $h_\theta(x) = 1$
- As $h_\theta(x) \longrightarrow 0$

  Cost $\longrightarrow \infty$
- Captures intuition that if $h_\theta(x) = 0$ (predict $P(y = 1 | x; \theta) = 0$) but $y = 1$, will penalize with a very large cost

"Analogous insights"

$y = 0$



$\text{Cost}(h_\theta(x), y) = 0$ if $h_\theta(x) = y$

$\text{Cost}(h_\theta(x), y) \longrightarrow \infty$ if $y = 0$ and $h_\theta(x) \longrightarrow 1$

$\text{Cost}(h_\theta(x), y) \longrightarrow \infty$ if $y = 1$ and $h_\theta(x) \longrightarrow 0$

Logistic regression cost function

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

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

Note: $y = 0$ or 1 always

} COMBINE

$$\text{Cost}(h_\theta(x), y) = -y \log(h_\theta(x)) - (1-y)\log(1 - h_\theta(x))$$

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

$$= -\frac{1}{m}\left\{\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1-y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right\}$$

- To fit parameters θ:

  minimize $J(\theta)$, to allow us to make a prediction given new x

OUTPUT: $h_\theta(x) = \dfrac{1}{1 + e^{-\theta^T x}}$ $\{ p(y=1 \mid x; \theta) \}$

- Use GRADIENT DESCENT AGAIN

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$ ↗ SIMULTANEOUS θ UPDATE

}

$$\frac{\partial}{\partial \theta_j} J(\theta_i) = \frac{\partial}{\partial \theta_j}\left\{-\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log(\theta_j^T x) + (1-y^{(i)}) \times \log(1 - \theta_j^T x)\right\}$$

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

$\Rightarrow$ Repeat {

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

ONLY DEF OF $h_\theta(x^{(i)})$ CHANGE

identical to linear regression

}

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\partial}{\partial \theta_j} \left\{ -\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log\left[g(\theta^T x) \right] + (1 - y^{(i)}) \cdot \log\left[1 - g(\theta^T x) \right] \right\}$$

$$g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} \quad \therefore$$

$$\frac{\partial}{\partial \theta_j} g(\theta^T x) \equiv g'(\theta^T x) = -\left(1 + e^{-\theta_k x_k}\right)^2 e^{-\theta_k x_k} \cdot x_j$$

$$= \frac{-e^{-\theta_k x_k} x_j}{\left(1 + e^{-\theta_k x_k}\right)^2} = \underbrace{g(\theta^T x)}_{\frac{1}{1 + e^{-\theta^T x}}} \underbrace{\left(1 - g(\theta^T x)\right)}_{\frac{-e^{-\theta^T x}}{1 + e^{-\theta^T x}}} \underbrace{x_j}_{x_j}$$

$$\Rightarrow \frac{\partial}{\partial \theta_j} J(\theta) = \left(y \frac{1}{g(\theta^T x)} - (1 - y) \frac{1}{1 - g(\theta^T x)} \right) \frac{\partial}{\partial \theta_j} g(\theta^T x)$$

$$= \left(\frac{y}{g(\theta^T x)} - \frac{(1 - y)}{1 - g(\theta^T x)} \right) g(\theta^T x) \left(1 - g(\theta^T x)\right) x_j$$

$$\Rightarrow \left\{ y \left(1 - g(\theta^T x)\right) - (1 - y)\, g(\theta^T x) \right\} x_j$$

$$\therefore \partial / \partial \theta_j J(\theta) = \left\{ y - g(\theta^T x) \right\} x_j \cdot -\frac{1}{m} \Sigma$$

$$\Rightarrow \boxed{\frac{\partial}{\partial \theta_j} J(\theta_i) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}}$$

where $h_\theta(x) = g(\theta^T x) = \dfrac{1}{1 + e^{-\theta^T x}}$

Gradient Descent

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

where $h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$

$\forall \; j = 0, \ldots, n$

- A vectorized implementation would take the form

$h = g(X\theta)$ — Rows $m \times (n+1)$, Cols $(n+1) \times 1$ — $\}$ $m \times 1$ VECTOR

$$J(\theta) = \frac{1}{m} \left\{ -y^T \log(h) - (1-y)^T \log(1-h) \right\}$$

Row Col Row Col

$$\therefore \quad \boxed{\theta := \theta - \frac{\alpha}{m} X^T \left\{ g(X\theta) - y \right\}}$$

$[X] = m \times (n+1) \longrightarrow [X^T] = (n+1) \times m$

$[y] = [g(X\theta)] = m \times 1$

$\Rightarrow [\theta] = (n+1) \times 1$ VECTOR

Advanced Optimisation

Optimization algorithm

Cost function $J(\theta)$. Want $\min_\theta J(\theta)$.

Given θ, we have code that can compute

- $J(\theta)$
- $\frac{\partial}{\partial \theta_j} J(\theta)$ (for $j = 0, 1, \ldots, n$)

Gradient descent:

Repeat {

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

}

PROVIDE

Other Examples

1. Conjugate gradient
2. BFGS
3. L - BFGS
   - quicker than standard gradient descent

PROS:

- No need to pick α
- Faster

CONS:

- More complex

Example:

$\min_\theta J(\theta)$

$\theta_1 = 5, \theta_2 = 5.$

$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$

$J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$

$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$

$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$

```
function [jVal, gradient]
        = costFunction(theta)
  jVal = (theta(1)-5)^2 + ...
         (theta(2)-5)^2;
  gradient = zeros(2,1);
  gradient(1) = 2*(theta(1)-5);
  gradient(2) = 2*(theta(2)-5);
```

```
options = optimset('GradObj', 'on', 'MaxIter', '100');
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] ...
      = fminunc(@costFunction, initialTheta, options);
```

POINTER TO COST FUNC

$\theta \in \mathbb{R}^d \quad d \geq 2$

Algorithm

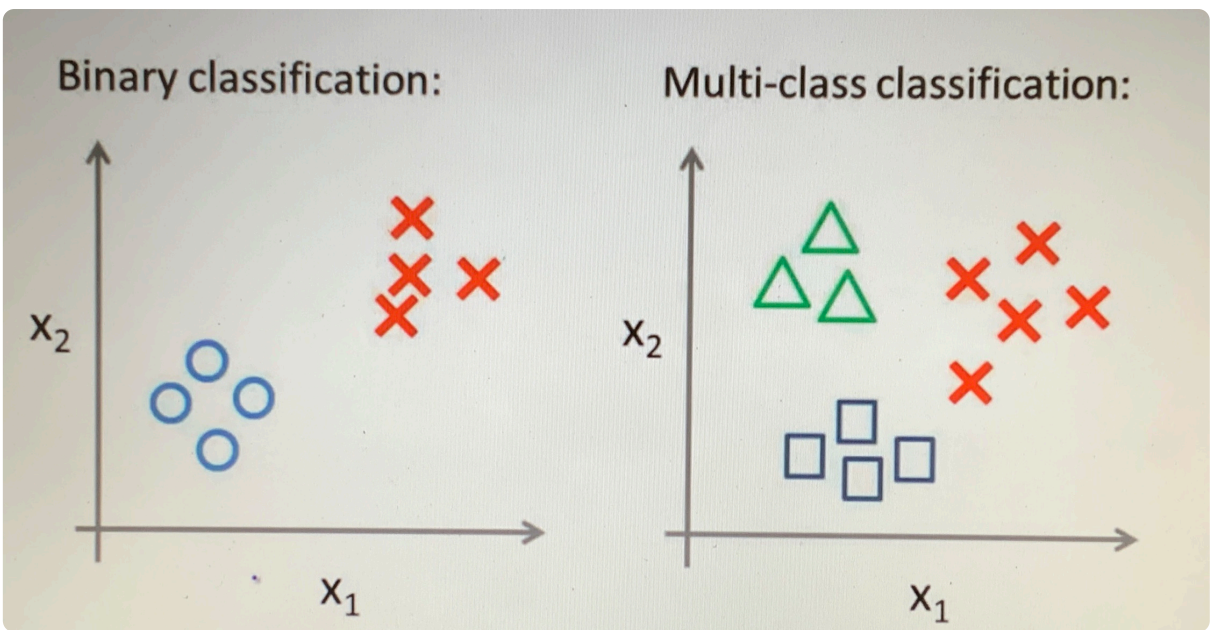
$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$ — θ_0: theta(1), θ_1: theta(2), θ_n: theta(n+1)

```
function [jVal, gradient] = costFunction(theta)
    jVal = [code to compute J(θ)];
    gradient(1) = [code to compute ∂/∂θ0 J(θ)];
    gradient(2) = [code to compute ∂/∂θ1 J(θ)];
    ⋮
    gradient(n+1) = [code to compute ∂/∂θn J(θ)];
```

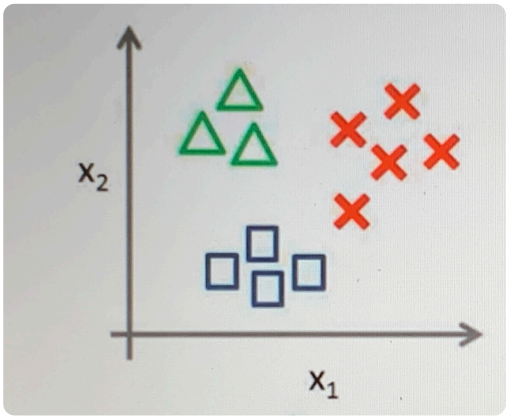
* Use for much larger problems

Multi-Class Classification

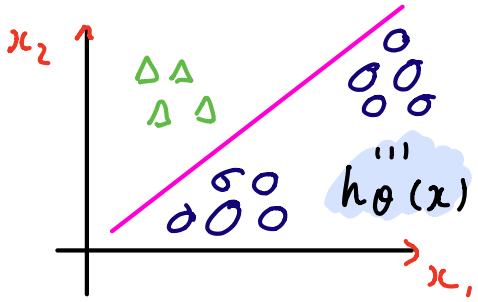
- E-mail tagging : Work, friends, hobby
- Weather : Sunny, Cloudy, Rainy, Snow
  0 1 2 3

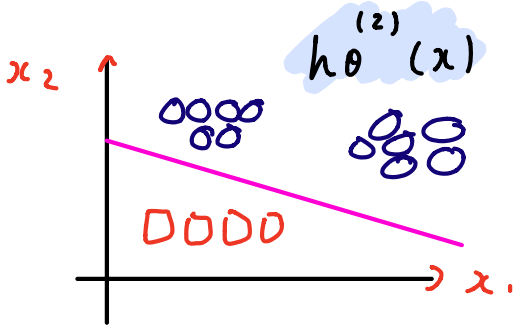


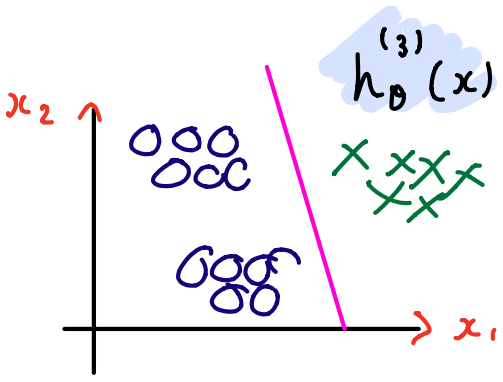
ONE Vs. ALL CLASSIFICATION



○ = -VE







- Turn into 3 separate Binary problems
- Create new training set for each
- Train standard logistic regression

$$h_\theta^{(i)}(x) = P(y = i \mid x; \theta) \quad (i = 1, 2, 3)$$

One-vs-all

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

$$\max_i h_\theta^{(i)}(x)$$

$$y \in \{0, 1, \ldots, n\}$$

$$h_\theta^{(0)}(x) = P(y = 0 \mid x; \theta)$$

$$h_\theta^{(1)}(x) = P(y = 1 \mid x; \theta)$$

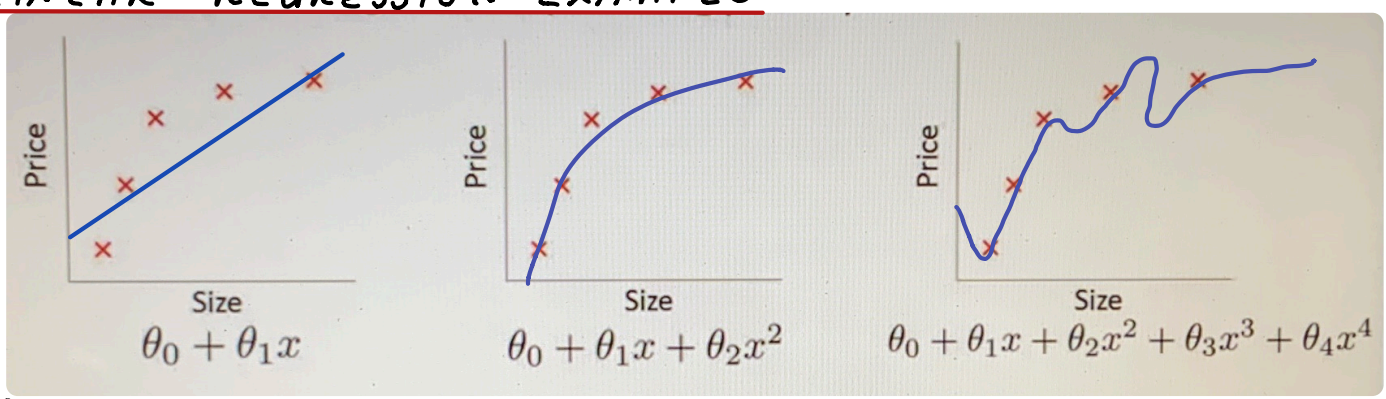
$$\vdots$$

$$h_\theta^{(n)}(x) = P(y = n \mid x; \theta)$$

$$\text{prediction} = \max_i \left[h_\theta^{(i)}(x) \right]$$

Overfitting

LINEAR REGRESSION EXAMPLE



'UNDERFITTING'

'HIGH BIAS'

* Too few features!

:)

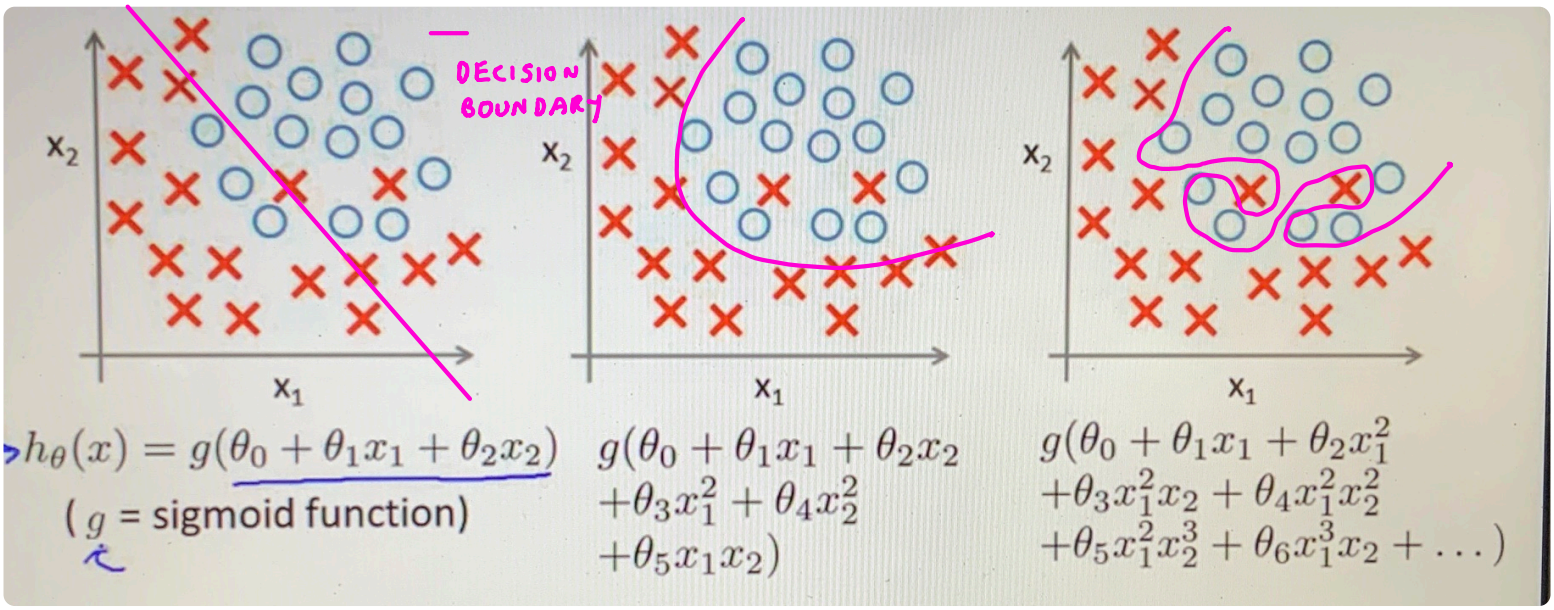
'OVERFITTING'

'HIGH VARIANCE OF HYPOTHESES'

* Too many features

Overfitting: If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) \approx 0$), but FAILS TO GENERALIZE TO NEW CASES

LOGISTIC REGRESSION EXAMPLE



UNDERFIT: Biased — Just Right! — OVERFIT: large variance

Addressing Overfitting:

1. Plotting hypotheses for the case of a LOW number of features - decide yourself

Options:

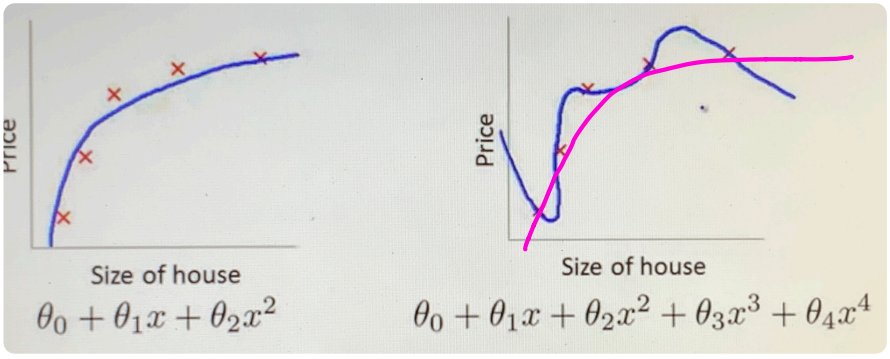
2. Reduce # of features
   - Manually select features to keep
   - Model selection algorithm (later)
3. Regularization
   - Keep all features, $\{x_i\}$, but reduce magnitudes of $\{\theta_i\}$

- Work well when

↳ LOTS OF FEATURES ~ EQUALLY IMPORTANT

USE REGULARIZATION

Regularization :: Cost Function



- Suppose we penalize and make θ_3 and θ_4 SMALL

PENALIZE CERTAIN FEATURES

→ $\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} \left[h_\theta(x^{(i)}) - y^{(i)} \right]^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2$

- When this modified $J(\theta)$ is minimized, since 1000 ≈ LARGE...

$\theta_3, \theta_4 \approx 0$

in order for successful minimization of $J(\theta)$

In general

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

- Note that θ_0 is never penalized, we use this when we're not sure which params to penalize usually due to there being a large # of them.

- λ - REGULARIZATION PARAMETER

  ↳ Controls the trade-off between fitting the training data well and keeping a simple $h_\theta(x)$ avoiding OVERFITTING

- LARGE λ ⟶ LARGE PENALIZATION

  ↳ All $\{\theta_i\} \approx 0$ ∴ UNDERFITTING

  ↳ $h_\theta(x) = \theta_0$

  * λ INFLATES COST *

Regularized Linear Regression

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

$\min_\theta J(\theta)$

Gradient Descent

Repeat {

$$\theta_0 := \theta_0 - \frac{\alpha}{m}\sum_{i=1}^{m}[h_\theta(x^{(i)}) - y^{(i)}]\,x_0^{(i)} \qquad \partial/\partial\theta_0\, J(\theta)$$

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

}

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

- since $\alpha\lambda/m > 0$, $(1 - \alpha\lambda/m) < 1 \approx 0.99$

  $\therefore$ We have added a term which effectively shrinks θ_j

Normal Equation

$$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \quad ; \quad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$m \times (n+1)$ $\qquad$ $m \times 1$

$$\Rightarrow \boxed{\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & & \\ & 1 & \\ & & \ddots \end{bmatrix}\right)^{-1} X^T y}$$

$(n+1) \times (n+1)$

- Note that $m < n$; $(X^T X)^{-1}$ ⟶ singular
- If $\lambda > 0$

$$\left(X^T X + \lambda \begin{bmatrix} 0 & & \\ & 1 & \\ & & \ddots \end{bmatrix}\right)^{-1}$$

$(n+1) \times (n+1)$

ALWAYS INVERTIBLE

Regularized Logistic Regression

* LOTS OF FEATURES ——→ OVERFITTING

——→ REGULARIZATION

- New cost function:

$$J(\theta) = -\left\{ \frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log (1-h_\theta(x^{(i)})) \right\}$$

$$+ \lambda / 2m \sum_{j=1}^{n} \theta_j^2$$

↳ Keeping parameters small by including the cost reduces the chances of overfitting

- Same gradient descent but

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

Advanced optimization

fminunc (@costFunction, $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$ theta(1), theta(2), theta(n+1))

```
function [jVal, gradient] = costFunction(theta)
    jVal = [code to compute J(θ)];
        → J(θ) = [ −(1/m) Σ_{i=1}^{m} y^(i) log(h_θ(x^(i))) + (1 − y^(i)) log 1 − h_θ(x^(i)) ] + (λ/2m) Σ_{j=1}^{n} θ_j^2
    gradient(1) = [code to compute ∂/∂θ_0 J(θ)];
        (1/m) Σ_{i=1}^{m} (h_θ(x^(i)) − y^(i)) x_0^(i)
    gradient(2) = [code to compute ∂/∂θ_1 J(θ)];
        ( (1/m) Σ_{i=1}^{m} (h_θ(x^(i)) − y^(i)) x_1^(i) ) + (λ/m) θ_1
    gradient(3) = [code to compute ∂/∂θ_2 J(θ)];
        ( (1/m) Σ_{i=1}^{m} (h_θ(x^(i)) − y^(i)) x_2^(i) ) + (λ/m) θ_2
    ⋮
    gradient(n+1) = [code to compute ∂/∂θ_n J(θ)];
```

J(θ)