Logistic Regression

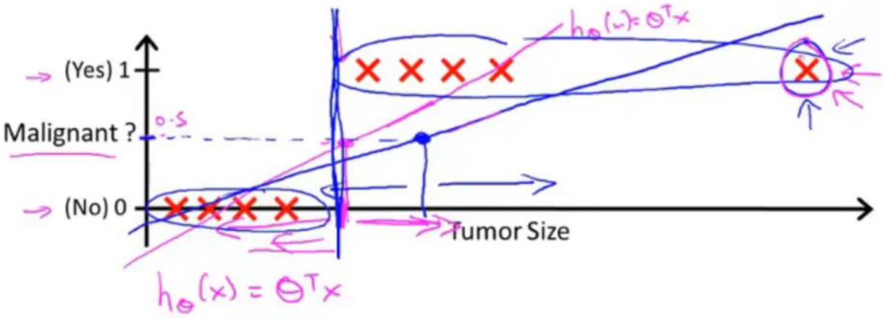
1 Classification

Classification

- Email: Spam / Not Spam?
- Online Transactions: Fraudulent (Yes / No)?
- Tumor: Malignant / Benign ?

$y \in \{0, 1\}$
0: "Negative Class" (e.g., benign tumor)
1: "Positive Class" (e.g., malignant tumor)

$y \in \{0, 1, 2, 3\}$



Threshold classifier output $h_\theta(x)$ at 0.5:

If $h_\theta(x) \geq 0.5$, predict "y = 1"

If $h_\theta(x) < 0.5$, predict "y = 0"

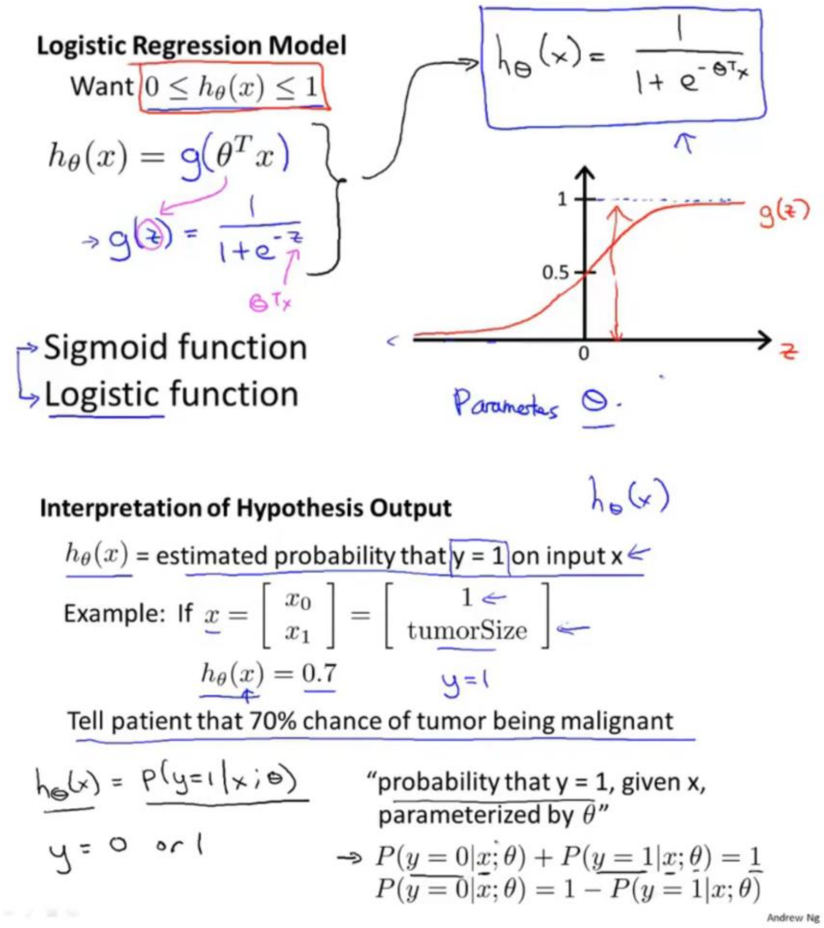
Classification: $\boxed{y = 0 \text{ or } 1}$

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

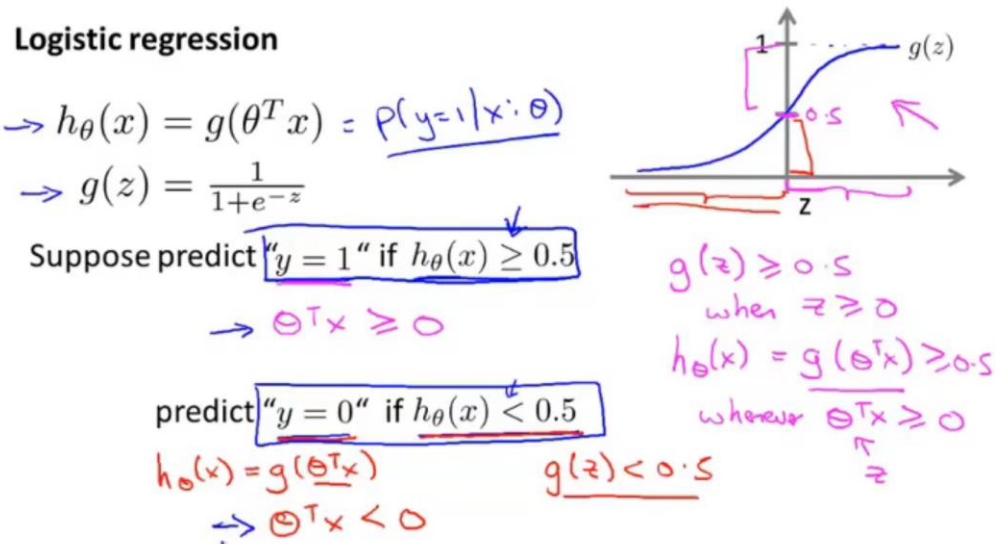
Logistic Regression: $0 \leq h_\theta(x) \leq 1$

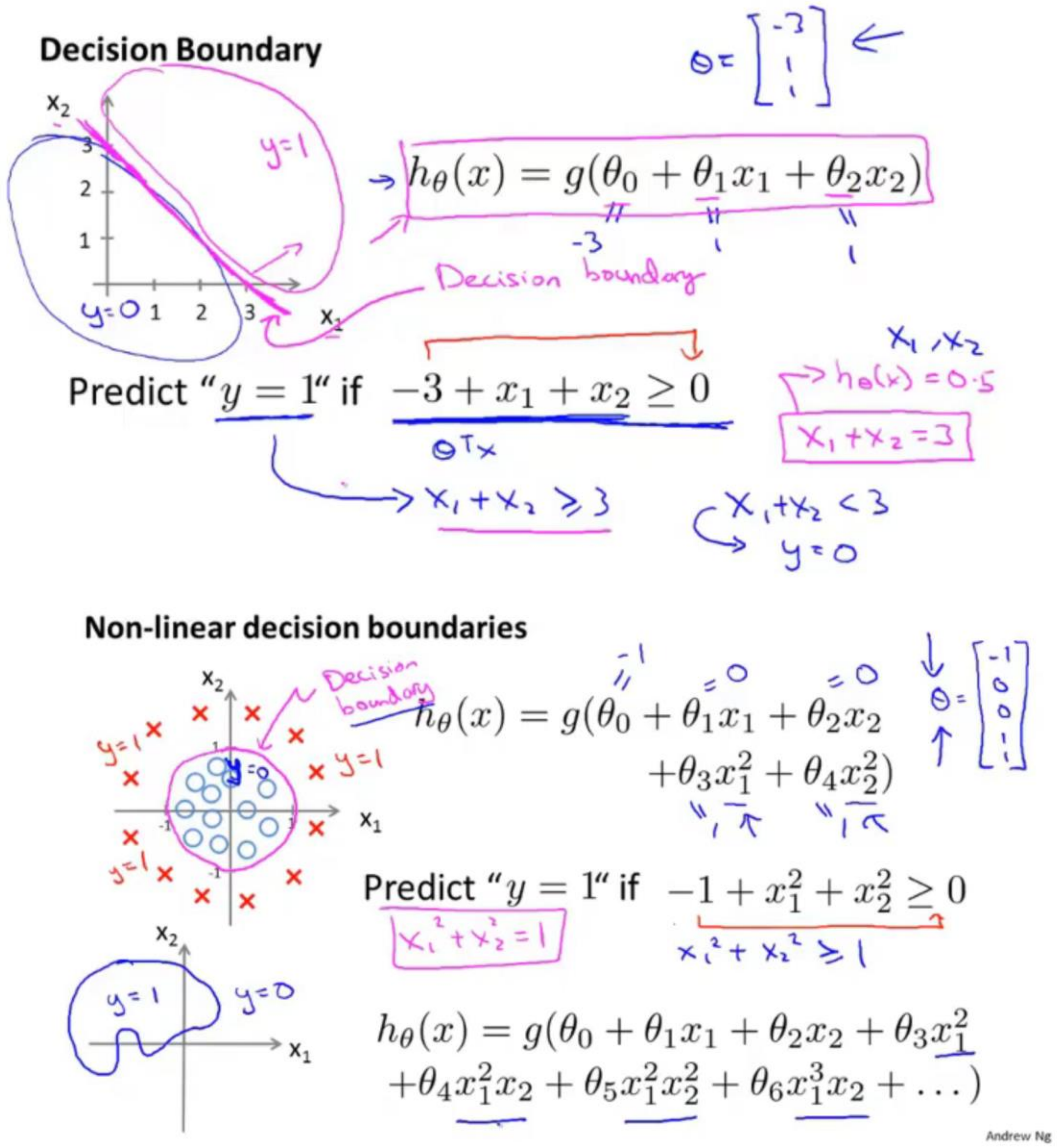
Classification

2 Hypothesis representation

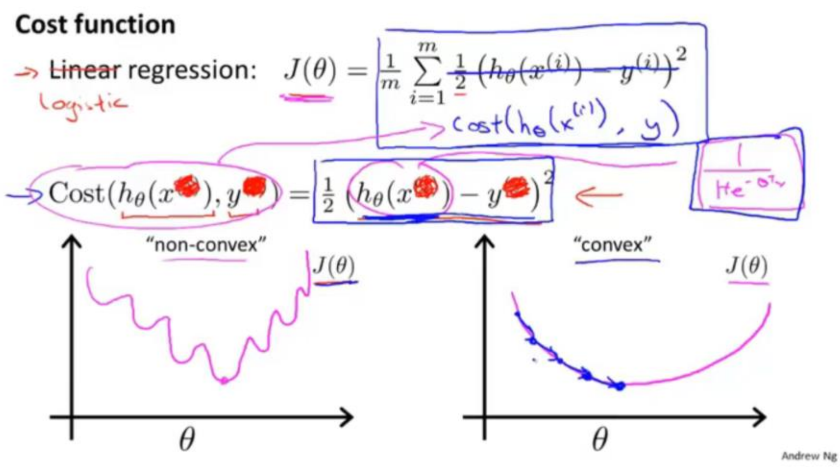


3 Decision boundary



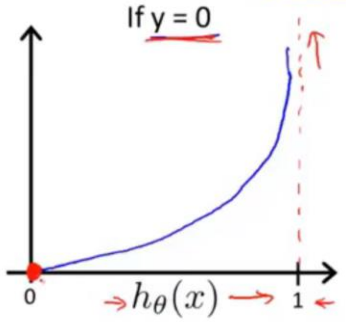


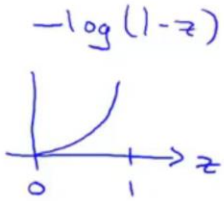
4 Cost function



Logistic regression cost function

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





Andrew Ng

5 Simplified cost function and gradient descent

Logistic regression cost function

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

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

Note: $y = \boxed{0}$ or $\boxed{1}$ always

$$\mathrm{Cost}(h_\theta(x), y) = -\underbrace{y}_{=1,\ =0}\log(h_\theta(x)) - \underbrace{(1-y)}_{=0,\ =1}\log(1-h_\theta(x))$$

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

If $y=0$: $\mathrm{Cost}(h_\theta(x), y) = -\log(1-h_\theta(x))$

Logistic regression cost function

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

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

To fit parameters θ:

$$\boxed{\min_\theta J(\theta) \quad \text{Get } \theta}$$

To make a prediction given new x:

Output $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$ $\quad p(y=1 \mid x;\theta)$

Gradient Descent

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

Want $\min_\theta J(\theta)$:

$$\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$ for $i = 0$ to n

Repeat {

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

(simultaneously update all θ_j)

}

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

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

Algorithm looks identical to linear regression!

Andrew Ng

6 Advanced optimization

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 {

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

}

Optimization algorithm

Given θ, we have code that can compute

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

Optimization algorithms:

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Advantages:

- No need to manually pick α
- Often faster than gradient descent.

Disadvantages:

- More complex

Example: $\min_{\theta} J(\theta)$

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \qquad \theta_1 = 5,\ \theta_2 = 5.$$

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

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

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

```
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);
```

$$\text{theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \begin{array}{l} \text{theta(1)} \\ \text{theta(2)} \\ \\ \text{theta(n+1)} \end{array}$$

```
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)];
```

7 Multi-class classification: One-vs-all

Multiclass classification

Email foldering/tagging: Work, Friends, Family, Hobby

y=1 y=2 y=3 y=4

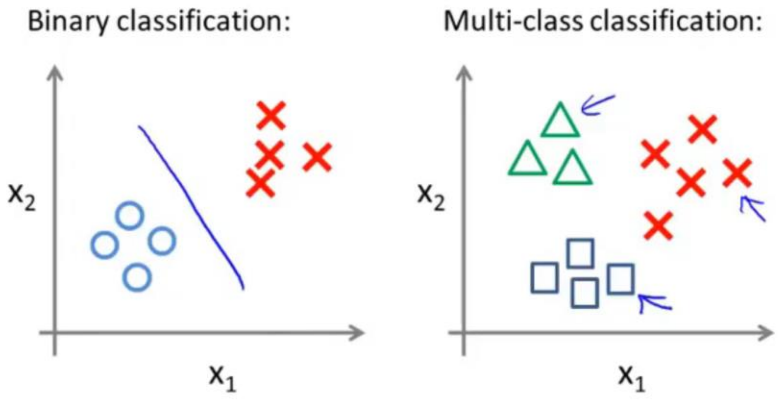
Medical diagrams: Not ill, Cold, Flu

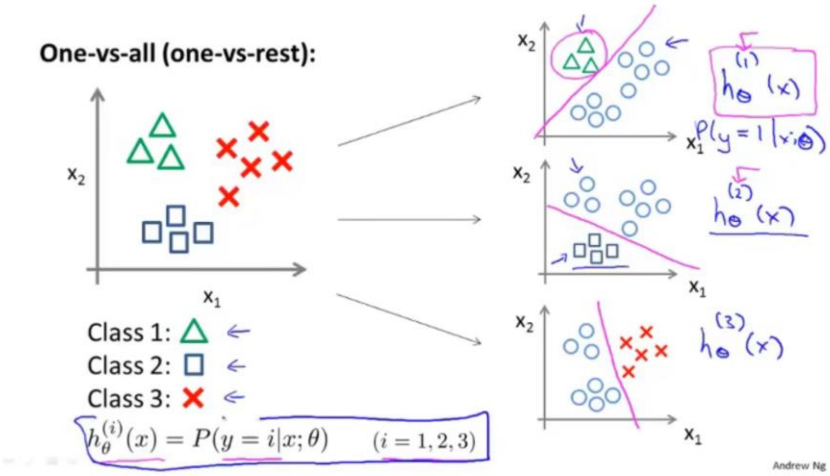
y=1 2 3

Weather: Sunny, Cloudy, Rain, Snow

y=1 2 3 4

0 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

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