



Machine Learning

Logistic Regression

Classification

Classification

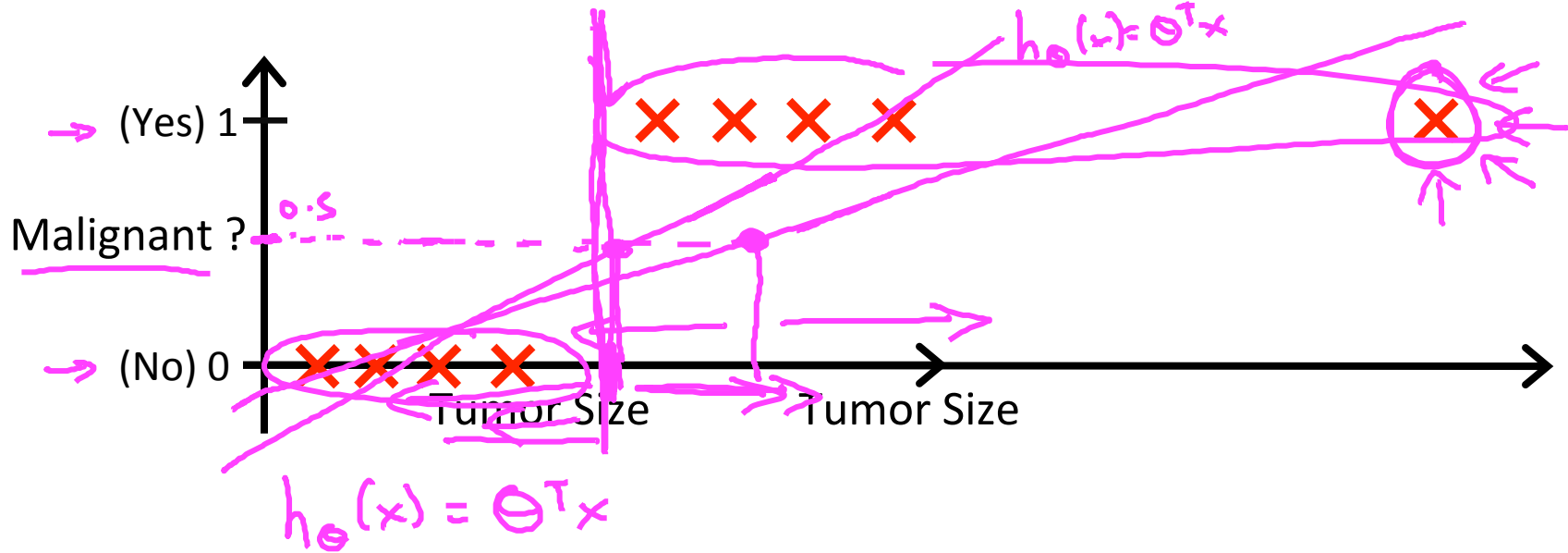
- Email: Spam / Not Spam?
- Online Transactions: Fraudulent (Yes / No)?
- Tumor: Malignant / Benign ?

→ $y \in \{0, 1\}$

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:

$$y = 0 \text{ or } 1$$

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

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

Classification



Machine Learning

Logistic Regression

Hypothesis Representation

Logistic Regression Model

Want $0 \leq h_{\theta}(x) \leq 1$

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

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

$\theta^T x$

Sigmoid function

Logistic function

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



Parameters θ .

Interpretation of Hypothesis Output

$$h_{\theta}(x)$$

$h_{\theta}(x)$ = estimated probability that $y = 1$ on input x ←

Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$ = $\begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$ ←

$h_{\theta}(x) = 0.7$ ←

$y = 1$

Tell patient that 70% chance of tumor being malignant

$$\underline{h_{\theta}(x) = P(y=1|x;\theta)}$$

$$\underline{y = 0 \text{ or } 1}$$

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

→ $P(y = 0|x;\theta) + P(y = 1|x;\theta) = 1$

→ $P(y = 0|x;\theta) = 1 - P(y = 1|x;\theta)$



Machine Learning

Logistic Regression

Decision boundary

Logistic regression

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

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

Suppose predict “ $y = 1$ ” if $h_{\theta}(x) \geq 0.5$

$$\theta^T x \geq 0$$

predict “ $y = 0$ ” if $h_{\theta}(x) < 0.5$

$$\theta^T x < 0$$



$$g(z) \geq 0.5 \\ \text{when } z \geq 0 \\ h_{\theta}(x) = g(\theta^T x)$$

$$g(z) < 0.5 \\ \text{when } z < 0$$

Decision Boundary



$$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix} \leftarrow$$

$$h_{\theta}(x) = g(\underbrace{\theta_0}_{-3} + \underbrace{\theta_1}_{1}x_1 + \underbrace{\theta_2}_{1}x_2)$$

Decision boundary

Predict " $y = 1$ " if $-3 + x_1 + x_2 \geq 0$

$\theta^T x$

x_1, x_2

$\rightarrow h_{\theta}(x) = 0.5$

$x_1 + x_2 = 3$

\rightarrow $x_1 + x_2 \geq 3$

$\rightarrow x_1 + x_2 < 3$
 $y = 0$

Non-linear decision boundaries



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

Handwritten blue annotations: $\theta_0 = -1$, $\theta_1 = 1$, $\theta_2 = 1$, $\theta_3 = 1$, $\theta_4 = 1$.

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

Predict "y = 1" if $-1 + x_1^2 + x_2^2 \geq 0$

Handwritten blue annotations: $x_1^2 + x_2^2 = 1$ (circled), $x_1^2 + x_2^2 \geq 1$ (underlined).



$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \dots)$$



Machine Learning

Logistic Regression

Cost function

Training
set:

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

m examples

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix}$$

\mathbb{R}^{n+1}

$$\underline{x_0 = 1}, \underline{y \in \{0, 1\}}$$

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

How to choose parameters θ ?

Cost function

→ Linear regression:
logistic

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$\text{cost}(h_{\theta}(x^{(i)}), y)$$

$$\text{Cost}(h_{\theta}(x), y) = \frac{1}{2} (h_{\theta}(x) - y)^2$$

$$\frac{1}{1 + e^{-\theta^T x}}$$



Logistic regression cost function

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

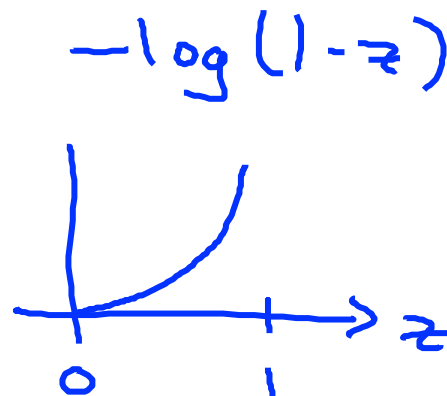
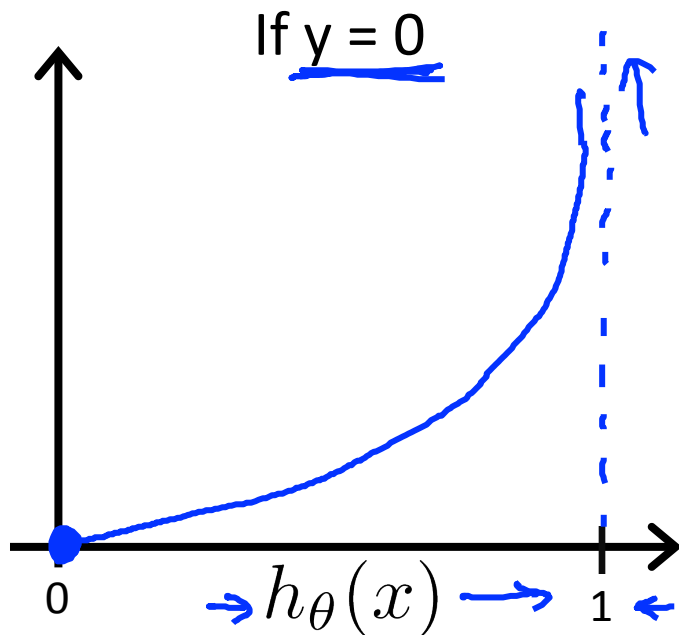


→ Cost = 0 if $y = 1, h_\theta(x) = 1$
But as $h_\theta(x) \rightarrow 0$
 $\text{Cost} \rightarrow \infty$

→ Captures intuition that if $h_\theta(x) = 0$,
(predict $P(y = 1|x; \theta) = 0$), but $y = 1$,
we'll penalize learning algorithm by a very
large cost.

Logistic regression cost function

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





Machine Learning

Logistic Regression

Simplified cost function
and gradient descent

Logistic regression cost function

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

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

Note: $y = 0$ or 1 always

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

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

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

Logistic regression cost function

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

To fit parameters θ :

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

Get $\underline{\theta}$

To make a prediction given new \underline{x} :

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

$$\underline{p(y=1 | x; \theta)}$$

Gradient Descent

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

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

Repeat {

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

}

(simultaneously update all θ_j)

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

Gradient Descent

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

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

Repeat {

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

}

(simultaneously update all θ_j)

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

$h_{\theta}(x) = \Theta^T x$

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

Algorithm looks identical to linear regression!



Machine Learning

Logistic Regression

Advanced optimization

Optimization algorithm

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

Given θ , we have code that can compute

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

Gradient descent:

Repeat {

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

}

Optimization algorithm

Given θ , we have code that can compute

$$\begin{aligned} & - J(\theta) \\ & - \frac{\partial}{\partial \theta_j} J(\theta) \end{aligned} \quad \leftarrow \quad (\text{for } j = 0, 1, \dots, n)$$

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:

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

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

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

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

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

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

```
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);
```

↑

↑

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

$$\underline{\text{theta}} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

Handwritten annotations in blue:

- θ_0 is labeled $\text{theta}(1)$ with an arrow pointing to it.
- θ_1 is labeled $\text{theta}(2)$ with an arrow pointing to it.
- θ_n is labeled $\text{theta}(n+1)$ with an arrow pointing to it.

```
function [jVal, gradient] = costFunction(theta)
```

```
    jVal = [code to compute  $J(\theta)$ ];
```

```
    gradient(1) = [code to compute  $\frac{\partial}{\partial \theta_0} J(\theta)$ ];
```

```
    gradient(2) = [code to compute  $\frac{\partial}{\partial \theta_1} J(\theta)$ ];
```

```
    :
```

```
    gradient(n+1) = [code to compute  $\frac{\partial}{\partial \theta_n} J(\theta)$ ];
```



Machine Learning

Logistic Regression

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 ←



Binary classification:



Multi-class classification:





One-vs-all (one-vs-rest):



Class 1:  

Class 2:  

Class 3:  

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



One-vs-all

Train a logistic regression classifier $\underline{h_{\theta}^{(i)}(x)}$ for each class \underline{i} to predict the probability that $\underline{y = i}$.

On a new input \underline{x} , to make a prediction, pick the class i that maximizes

$$\max_{\underline{i}} \underline{h_{\theta}^{(i)}(x)}$$
