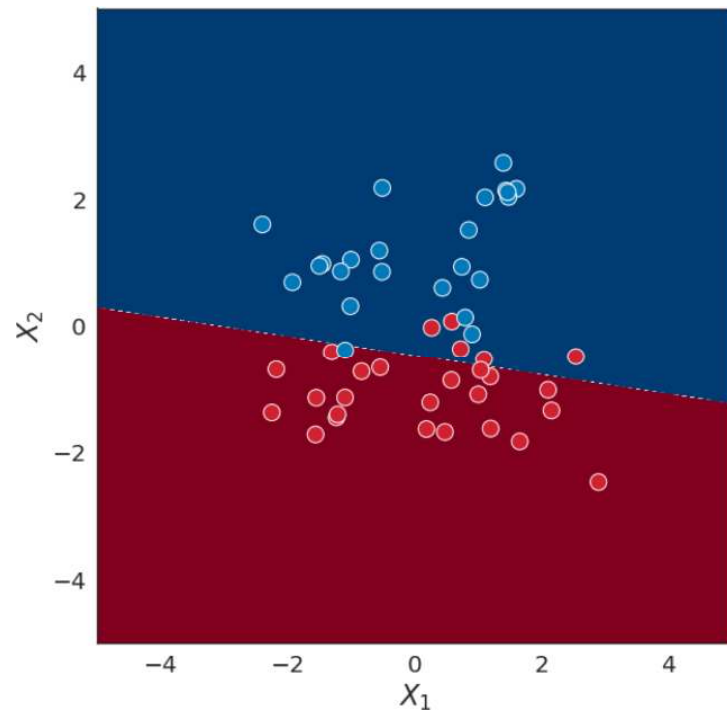


BBM406: Fundamentals of Machine Learning

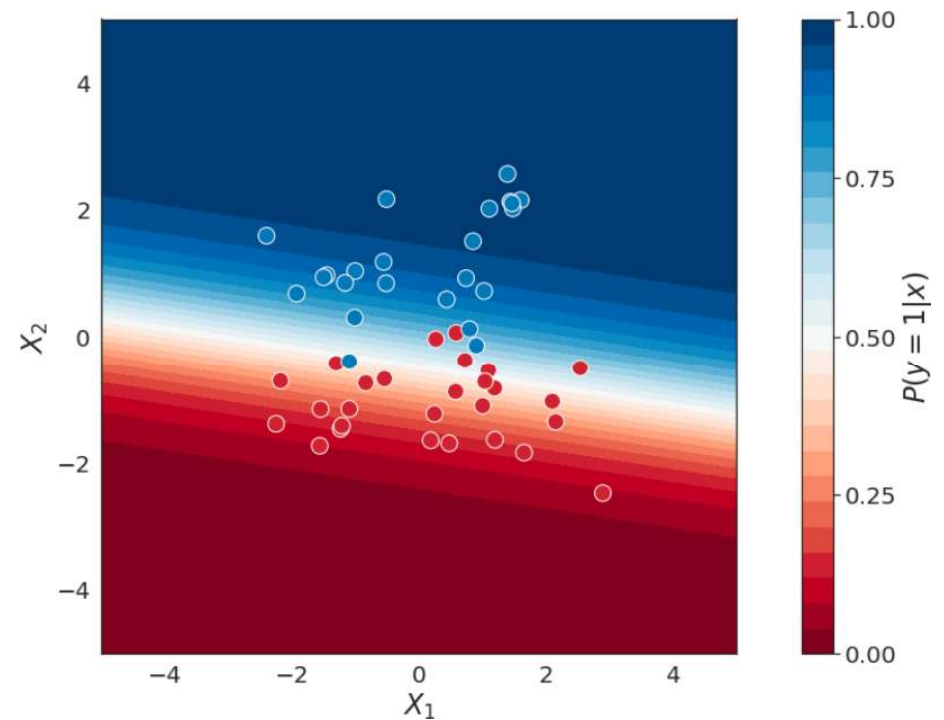
Logistic Regression

Types of Discriminative Methods

Class prediction



Probability prediction

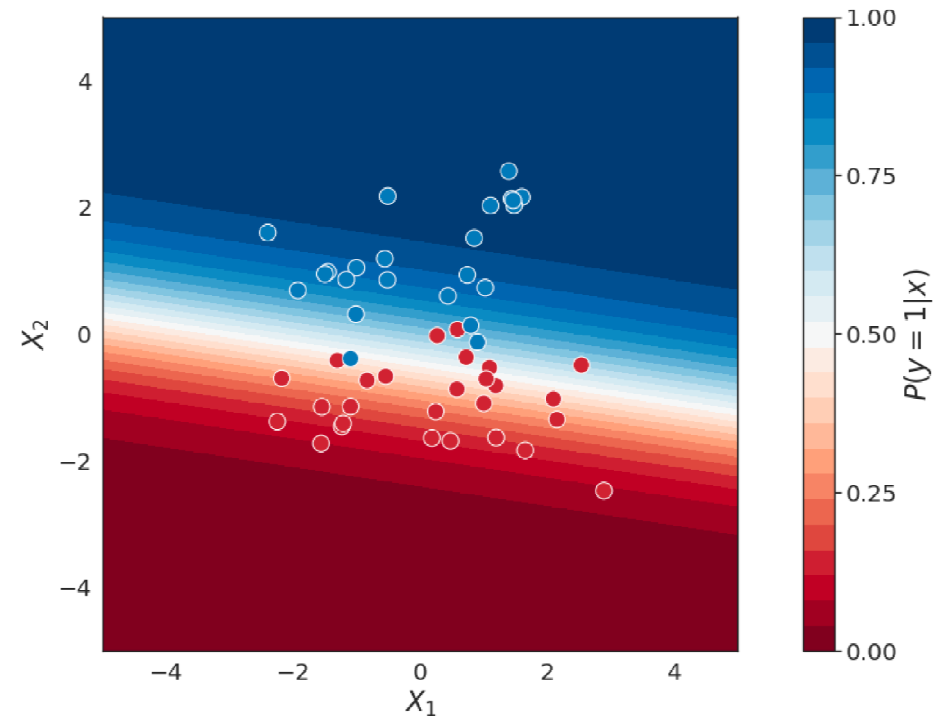


Classification Based on Probability

Class prediction

- Effective and useful
- Borderline cases are problematic

Key Idea: Instead of predicting only the class label, output the probability of the instance being within that class
o i.e., learn $p(y | x)$



Classification via Linear Regression ?

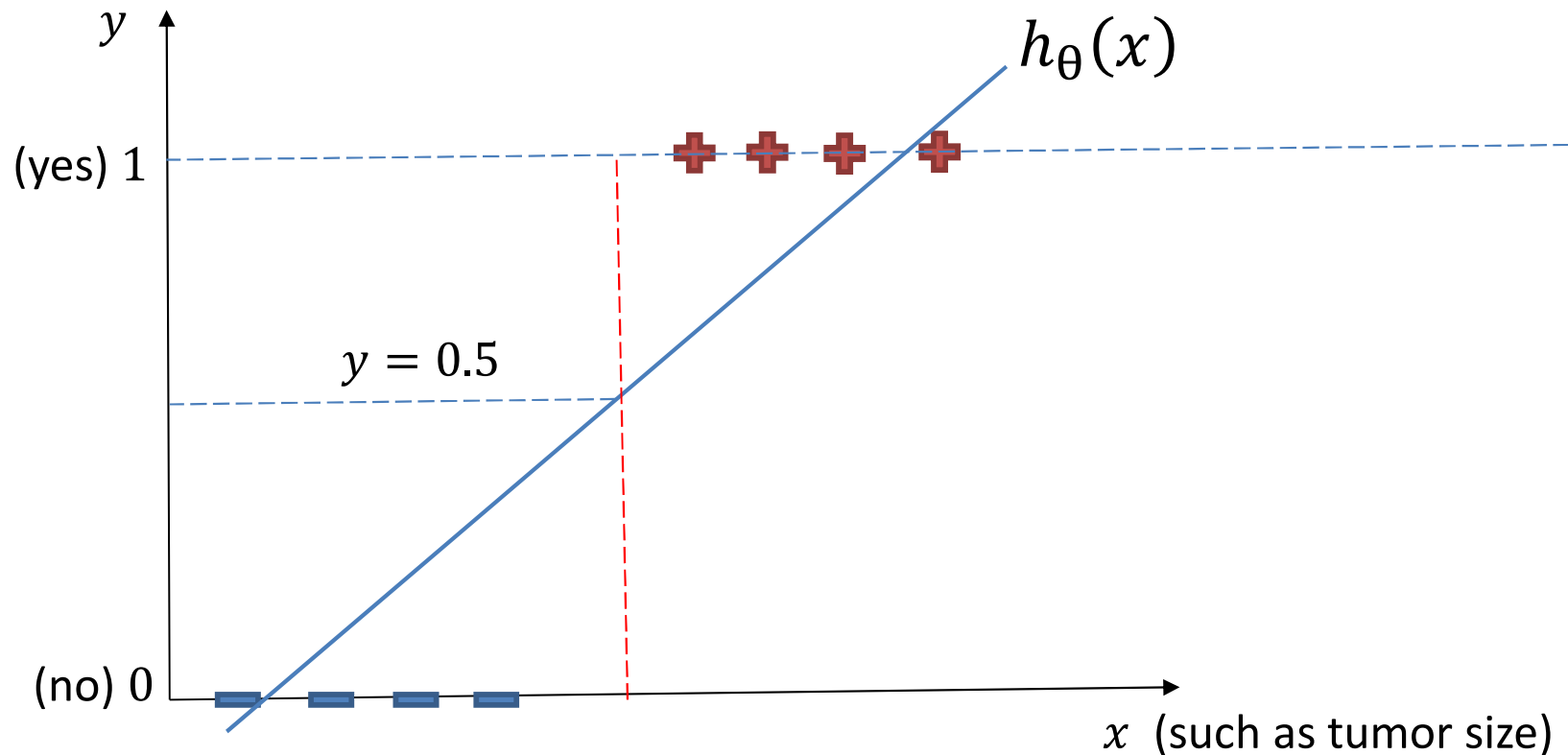
- Can we use linear regression for classification?
- Hypothesis function for linear regression is :

$$y = h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d = \theta^T x$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix} \quad x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \quad h_{\theta}(x) = \theta^T x$$

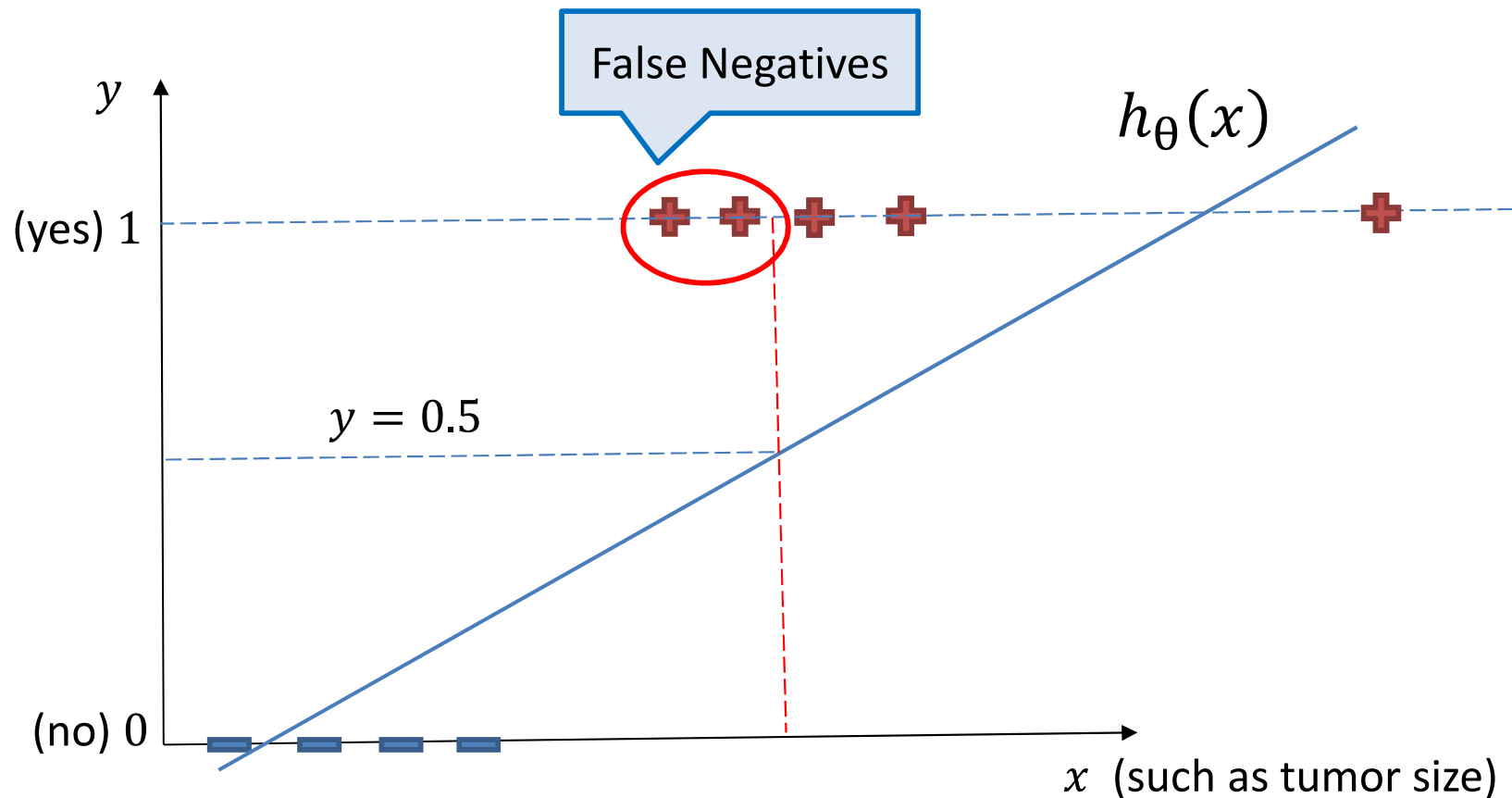
- $h_{\theta}(x)$ can take real values, so we can not use it for classification directly.

Linear Regression For Classification



- For this data, $y=0.5$ threshold looks good.
- However, if the new data arrives and $h_{\theta}(x)$ changes, is it still good?

Linear Regression For Classification



- With the new data, $h_{\theta}(x)$ changed.
- And two positive instances are classified as negative

Classification via Linear Regression ?

- We can apply thresholding but it is very dependent to data and sensitive to noise.
- Therefore, we need another approach for applying this for classification.

Logistic Regression

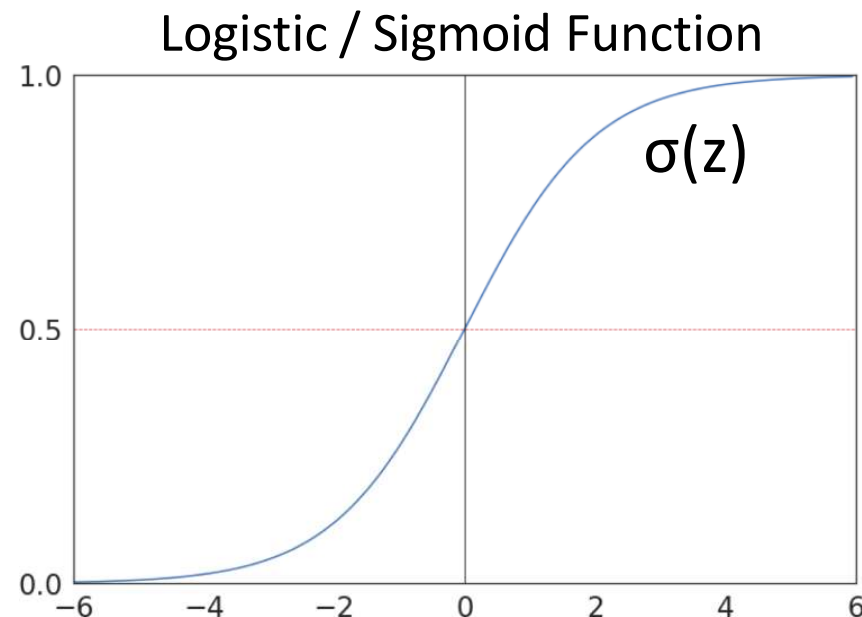
- Takes a probabilistic approach to learn discriminative functions

- $h_{\theta}(x)$ should give us $p(y = 1|x; \theta)$
- We want $0 \leq h_{\theta}(x) \leq 1$

Can't just use linear regression with a threshold

- For this purpose we can use sigmoid (logistic) function:

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

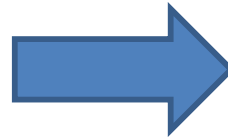


Logistic Regression

- By replacing our hypothesis in linear regression $h_{\theta}(x) = \theta^T x$ with logistic function, we obtain Logistic regression model :

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

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



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

Interpretation of Hypothesis Output

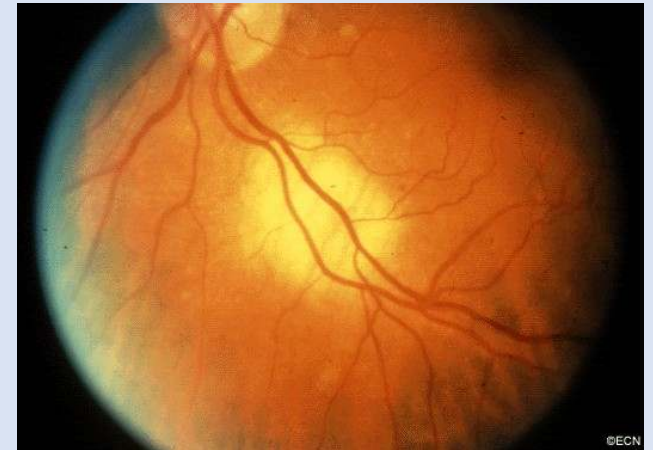
$$h_{\theta}(x) = \text{estimated } p(y = 1|x; \theta)$$

Example: Ocular tumor diagnosis from size

$$x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ tumorSize \end{bmatrix}$$

$$h_{\theta}(x) = 0.75$$

Tumor has a 75% chance of being malignant



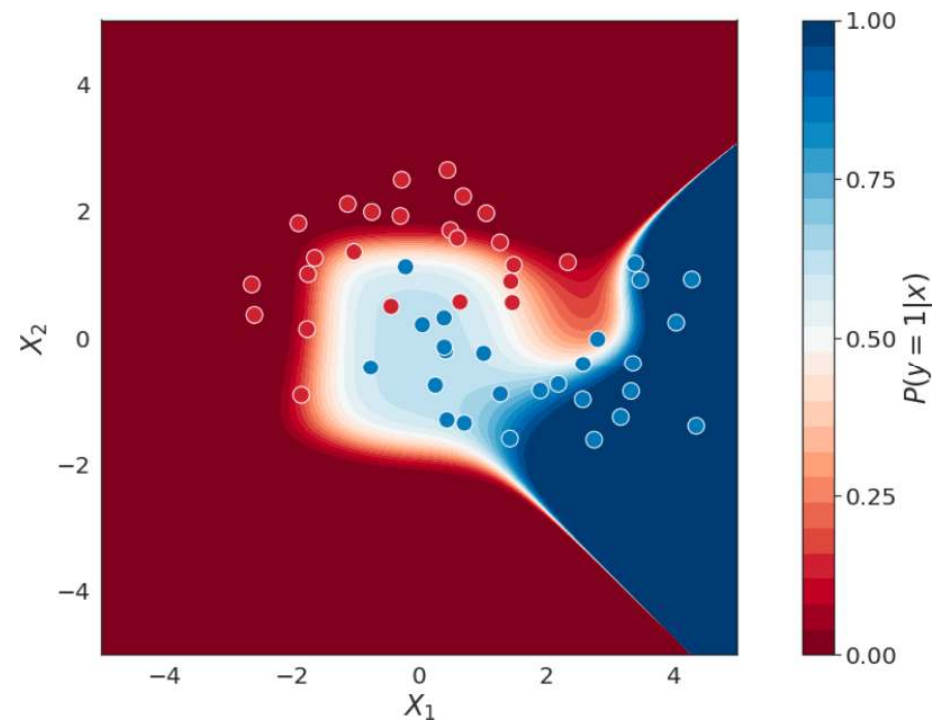
Note that: $p(y = 0|x; \theta) + p(y = 1|x; \theta) = 1$

Therefore, $p(y = 0|x; \theta) = 1 - p(y = 1|x; \theta)$

Non-Linear Decision Boundary

Can apply basis expansion to features, same as with linear regression

$$x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ x_2^2 \\ x_1^2 x_2 \\ x_2^2 x_1^2 \\ \dots \end{bmatrix}$$



Logistic Regression

- Given $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})\}$

Where $x^{(i)} \in R^d, y^{(i)} \in \{0,1\}$

- Model: $h_{\theta}(x) = \sigma(\theta^T x)$

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

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \dots \\ \theta_d \end{bmatrix} \quad x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \dots \\ x_d \end{bmatrix}$$

Logistic Regression Objective Function

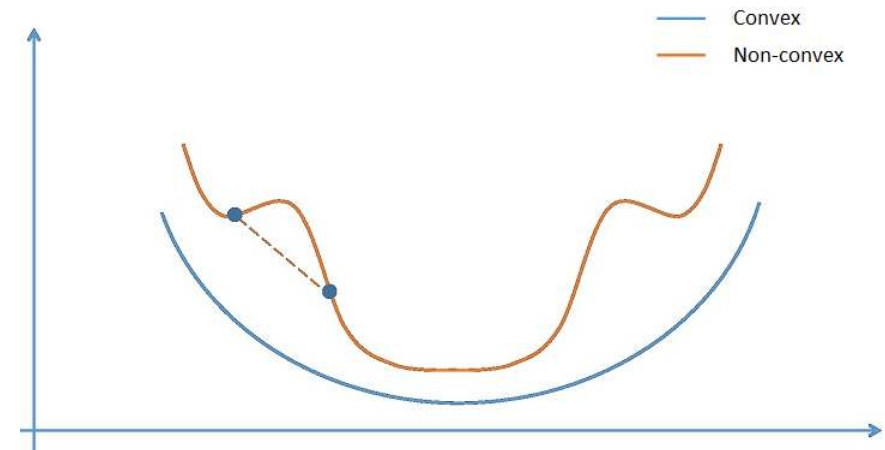
- Can't just use squared loss as in linear regression:

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

- Using the logistic regression model

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

results in a **non-convex** optimization



Deriving the Cost Function via Maximum Likelihood Estimation

- Likelihood of data is given by:

$$l(\theta) = \prod_{i=1}^n P(y^{(i)} | x^{(i)}; \theta)$$

- So, looking for the θ that maximizes the likelihood

$$\theta_{MLE} = \operatorname{argmax}_{\theta} l(\theta) = \operatorname{argmax}_{\theta} \prod_{i=1}^n P(y^{(i)} | x^{(i)}; \theta)$$

- Can take the log without changing the solution:

$$\theta_{MLE} = \operatorname{argmax}_{\theta} \log \prod_{i=1}^n P(y^{(i)} | x^{(i)}; \theta) = \operatorname{argmax}_{\theta} \sum_{i=1}^n \log P(y^{(i)} | x^{(i)}; \theta)$$

Deriving the Cost Function via Maximum Likelihood Estimation

- Expand as follows:

$$\begin{aligned}\theta_{MLE} &= \operatorname{argmax}_{\theta} \sum_{i=1}^n \log P(y^{(i)} | x^{(i)}; \theta) \\ &= \operatorname{argmax}_{\theta} \sum_{i=1}^n y^{(i)} \log P(y^{(i)} = 1 | x^{(i)}; \theta) + (1 - y^{(i)}) \log(1 - P(y^{(i)} = 1 | x^{(i)}; \theta))\end{aligned}$$

- Substitute in model, and take negative to yield

Logistic regression objective: $\min_{\theta} J(\theta)$

$$J(\theta) = - \sum_{i=1}^n (y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})))$$

Intuition Behind the Objective

$$J(\theta) = - \sum_{i=1}^n (y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})))$$

- Cost of a single instance:

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

- Can re-write objective function as

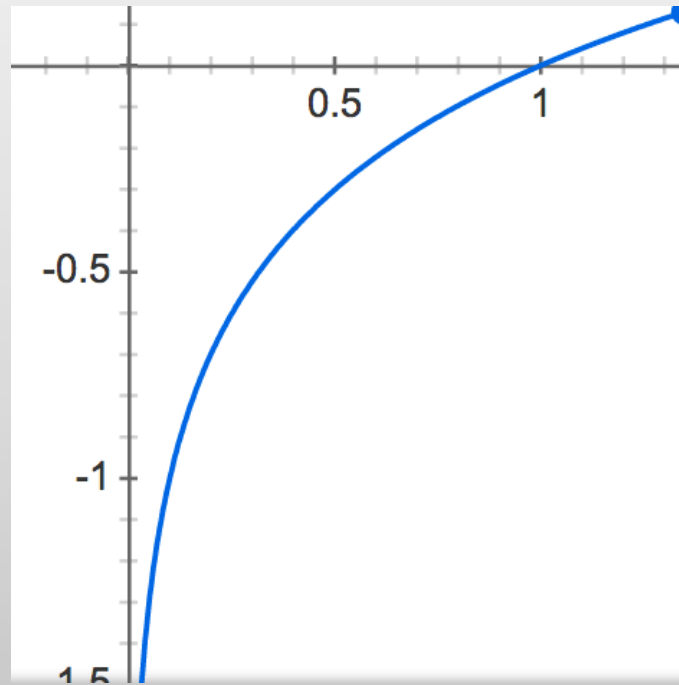
$$J(\theta) = - \sum_{i=1}^n \text{cost}(h_{\theta}(x^{(i)}), y^{(i)})$$

Compare to linear regression: $J(\theta) = \frac{1}{2n} \sum_{i=1}^n (h_{\theta}(x^{(i)}) - y^{(i)})^2$

Intuition Behind the Objective

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

Aside: Recall the plot of $\log(z)$

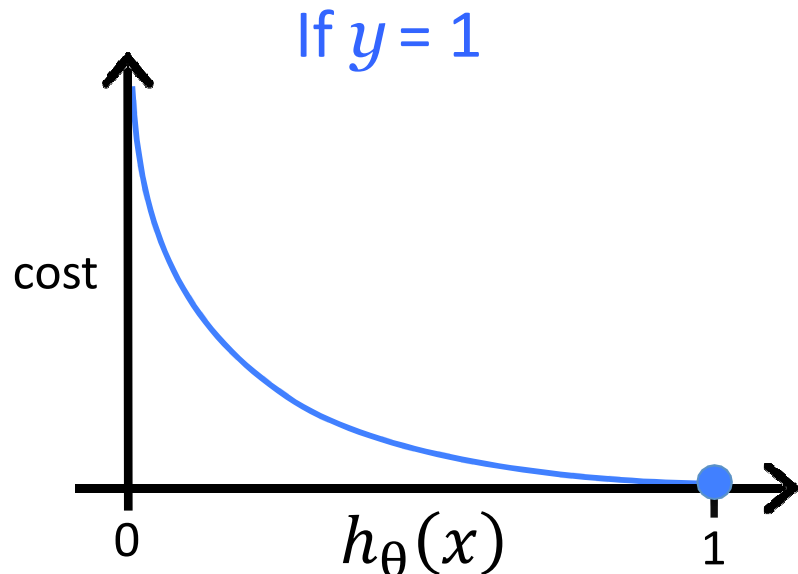


Intuition Behind the Objective

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

If $y = 1$

- Cost = 0 if prediction is correct
- As $h_{\theta}(x) \rightarrow 0$, $\text{cost} \rightarrow \infty$
- Captures intuition that larger mistakes should get larger penalties
 - e.g., predict $h_{\theta}(x) \rightarrow 0$, but $y = 1$

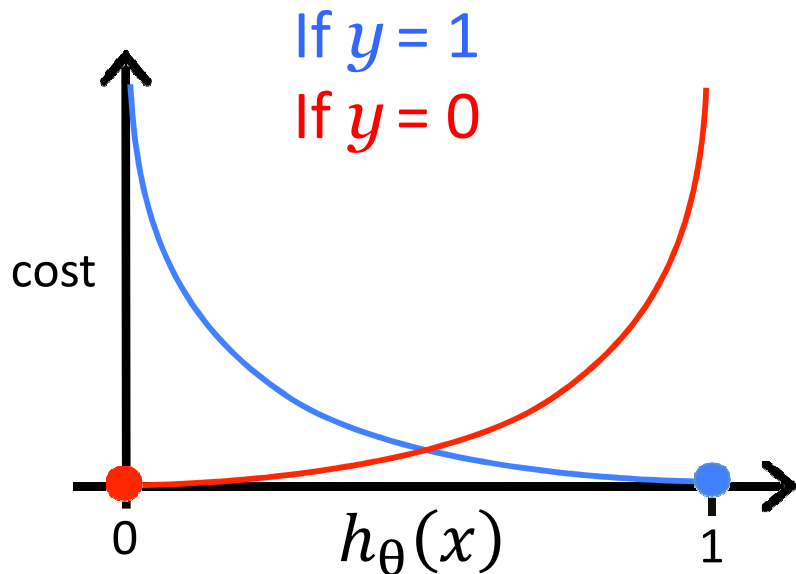


Intuition Behind the Objective

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

If $y = 0$

- Cost = 0 if prediction is correct
- As $(1 - h_{\theta}(x)) \rightarrow 0$, $\text{cost} \rightarrow \infty$
- Captures intuition that larger mistakes should get larger penalties



Gradient Descent for Logistic Regression

$$J(\theta) = - \sum_{i=1}^n (y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})))$$

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

- Initialize θ
- Repeat until convergence

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

simultaneous update
for $j = 0 \dots d$

Use the natural logarithm ($\ln = \log_e$) to cancel with the $\exp()$ in $h_{\theta}(x)$

Gradient Descent for Logistic Regression

$$J(\theta) = - \sum_{i=1}^n (y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})))$$

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

- Initialize θ simultaneous update for
- Repeat until convergence $j = 0 \dots d$

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

This looks almost identical to linear regression, except $1/n$ term!

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

Gradient Descent for Logistic Regression

$$J(\theta) = - \sum_{i=1}^n (y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})))$$

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

- Initialize θ
 - Repeat until convergence
- simultaneous update for $j = 0 \dots d$

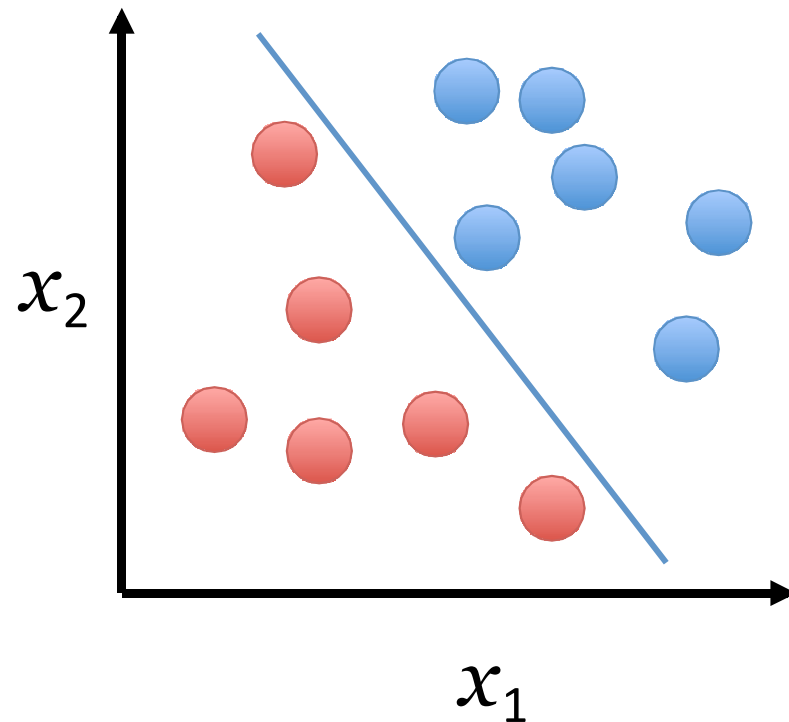
$$\theta_j \leftarrow \theta_j - \alpha \sum_{i=1}^n (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

However, the form of the model is very different!

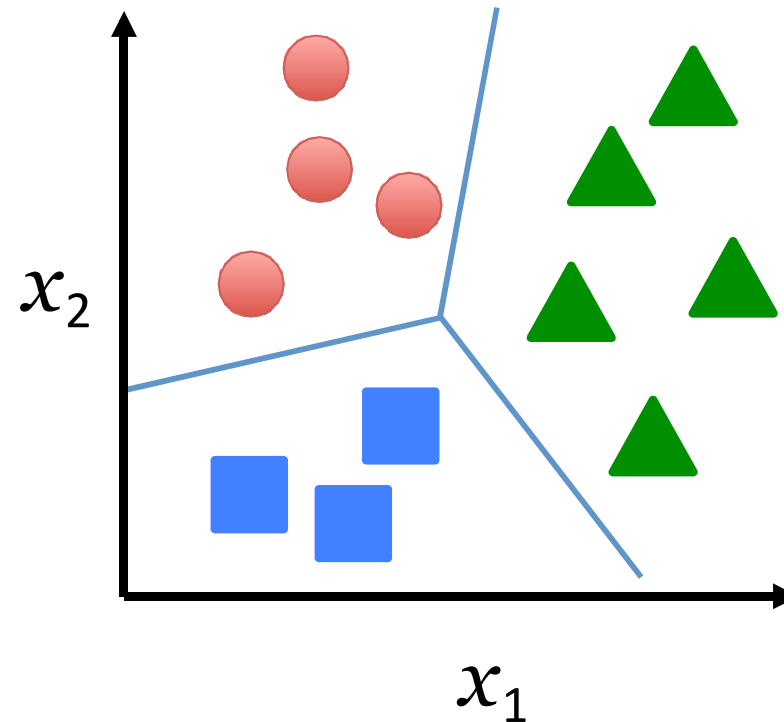
$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Multi-Class Classification

Binary classification:



Multi-class classification:

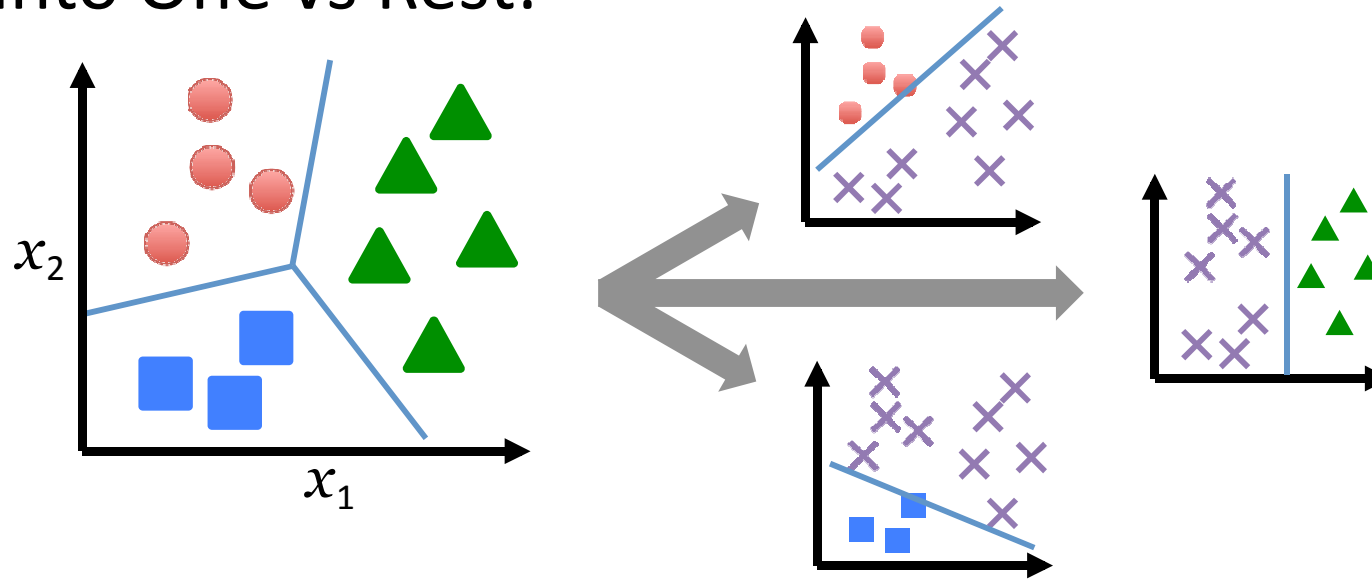


Disease diagnosis: healthy / cold / flu / pneumonia

Object classification: desk / chair / monitor / bookcase

Multi-Class Logistic Regression

Split into One vs Rest:



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

Implementing Multi-Class Logistic Regression

- Use $h_{\theta}^{(c)}(x)$ as the model for class c
- Gradient descent simultaneously updates all parameters for all models
 - Same derivative as before
- On a new input x , predict class label by picking the class i that maximizes

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