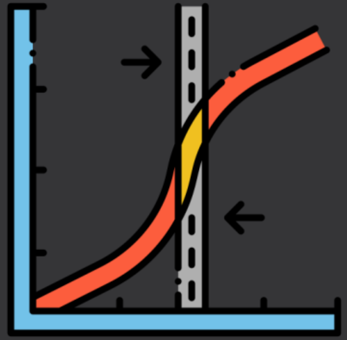


Logistic Regression Classification

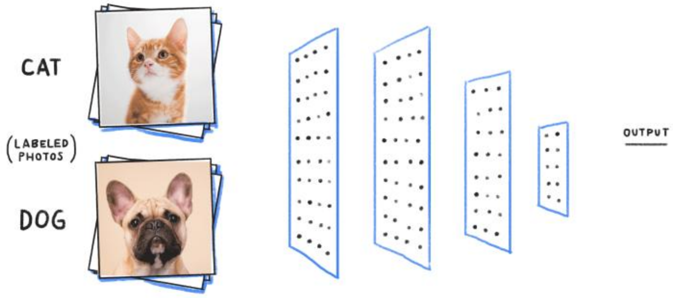
Lecture 10 – HCCDA-AI

Imran Nawar

29 June 2025

Classification

- Predicts discrete-valued output.
- Classification is the task of predicting a discrete label (or class) for an input.
- Output is categorical: each input is assigned to one of the predefined classes.



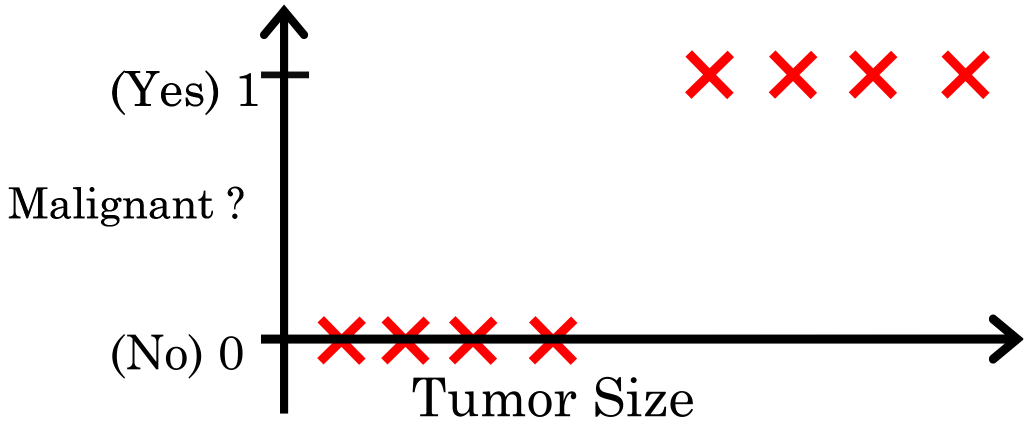
Common Examples:

- **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)

Classifying Tumor as Malignant or Benign:



Steps:

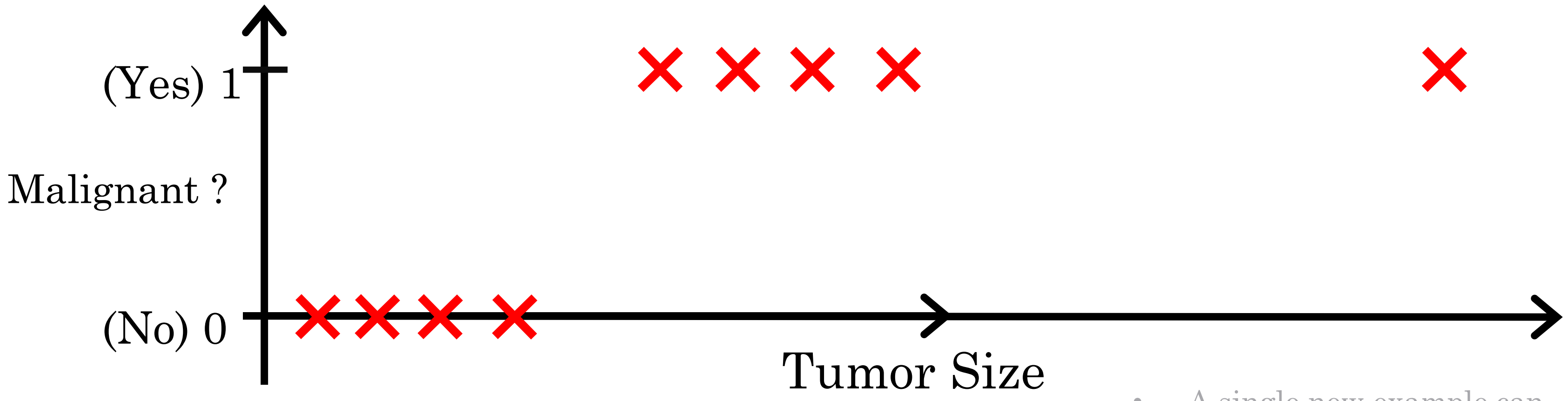
- Apply Linear Regression
- Pick a threshold
- For this dataset, linear regression seems to work reasonably well.

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"

Classifying Tumor as Malignant or Benign:



- A single new example can break it
- Applying linear regression to a classification problem usually is not a great idea!!

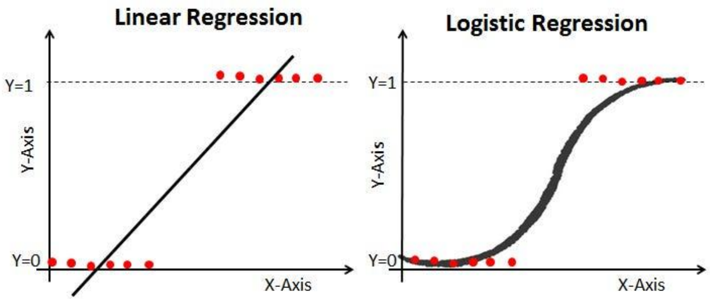
Conclusion:

- We need a model designed to predict probabilities and handle classification boundaries more reliably.

Logistic Regression

Logistic Regression

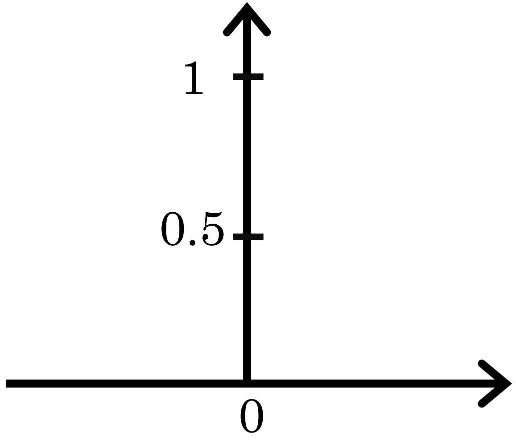
- Logistic regression predicts the probability that the input belongs to **class 1**, the class of interest.
- It uses the **sigmoid** (logistic function) to convert linear output into probabilities in the range [0, 1].



Sigmoid Function:

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

- Converts linear output into probabilities.
- Helps determine class labels based on a threshold (e.g., ≥ 0.5 → class 1, < 0.5 → class 0).



Logistic Regression

Hypothesis Representation

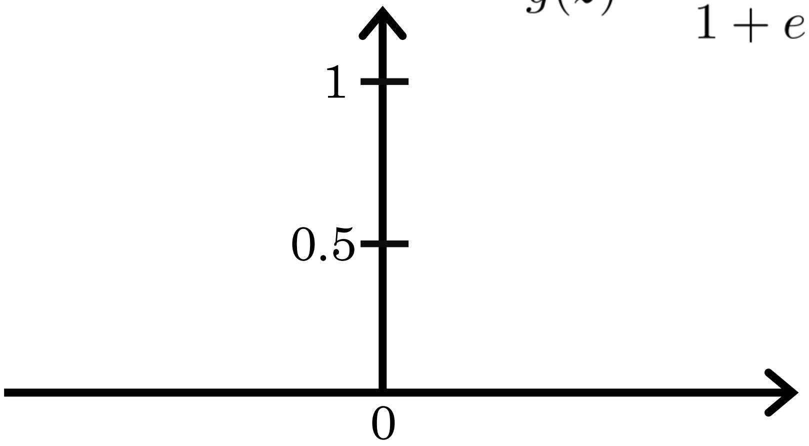
Logistic Regression Model

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

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

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

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



Sigmoid function
Logistic function

Interpretation of Hypothesis Output

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

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

$$h_\theta(x) = 0.7$$

Tell patient that 70% chance of tumor being malignant

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

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

Logistic Regression

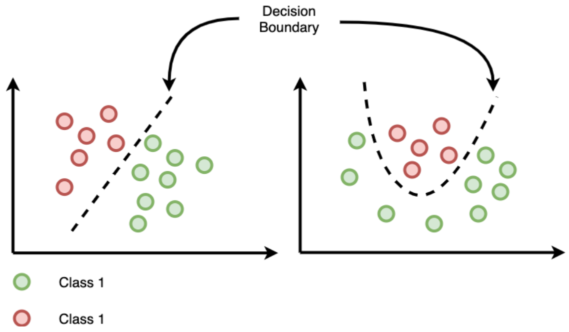
Decision Boundary

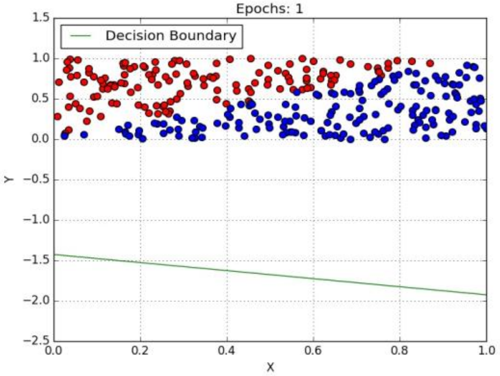
Decision Boundary

- A decision boundary is a line, surface, or hypersurface that separates different classes in a feature space.
- It guides how data points are classified in a supervised learning algorithm.

Decision Boundary in Logistic Regression:

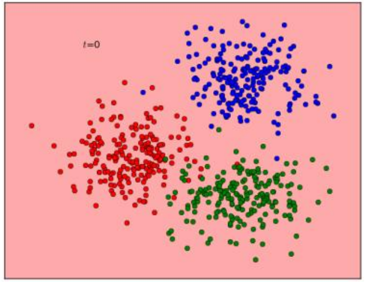
- Logistic regression aims to find a proper fit for the decision boundary to classify new data accurately.
- It is linear boundary in two dimensional space.



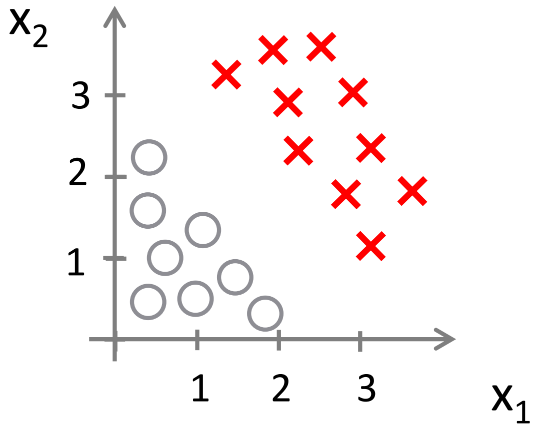


Key Insights:

- **Linear Decision Boundary:**
  - Suitable for problems where classes can be separated by a straight line.
- **Non Linear Boundaries:**
  - More complex models like decision trees, SVMs, or neural networks handle nonlinear separations.



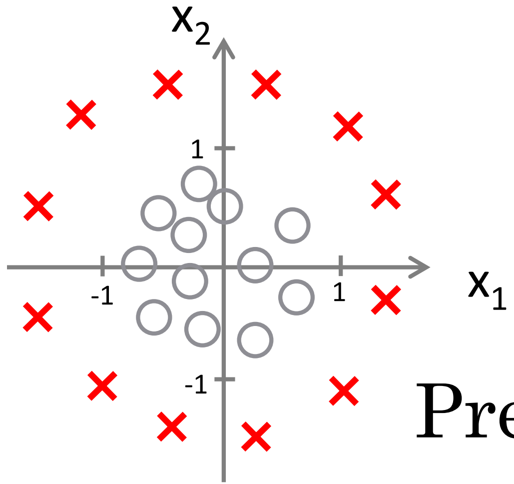
Decision Boundary



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

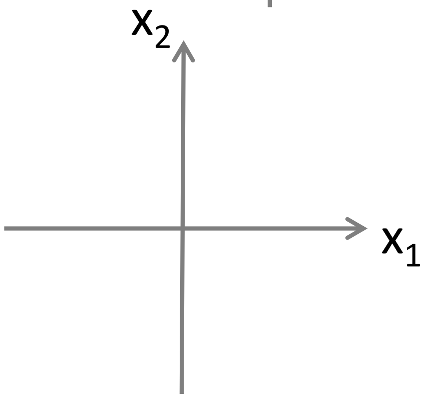
Predict " $y = 1$ " if $-3 + x_1 + x_2 \geq 0$

Non-linear decision boundaries



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

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



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

Logistic Regression

Cost Function

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$

m examples $x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix}$ $x_0 = 1, y \in \{0, 1\}$

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

How to choose parameters θ?

Cost Function

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

$$\mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

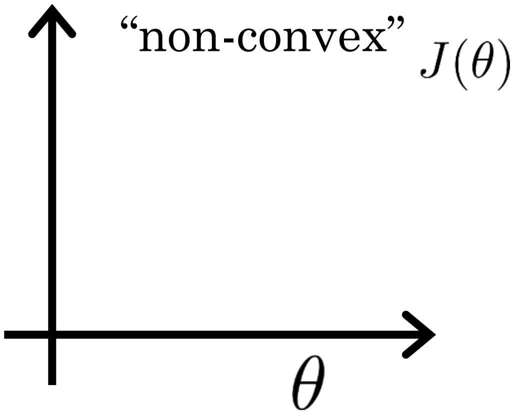
$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left(h_\theta(x^{(i)})_, \, y^{(i)} \right)^2$$

- Squared error leads to a **non-convex** cost function in logistic regression.
- Gradient descent may get stuck in local minima.

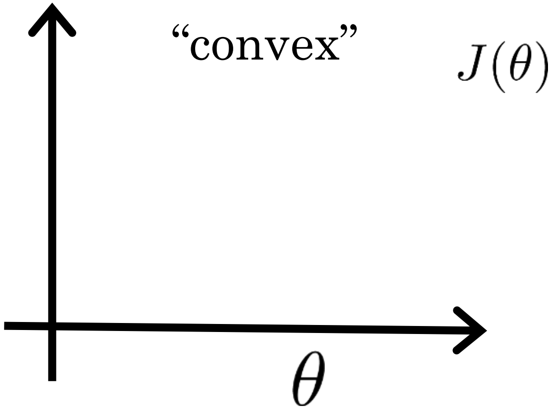
Cost Function

- The squared error cost function work will for linear regression, but if we use this particular cost function for logistic regression, this will be a non-convex function of parameter θ.
- For logistic regression: $h_\theta(x) = \dfrac{1}{1 + e^{-\theta^T x}}$
- If put this sigmoid function in cost function. The $J(\theta)$ will look like figure a) with many local optimum.
- Figure a) is a non-convex function, if gradient descent is used, it is not guaranteed to converge to global optimum.

a)



b)

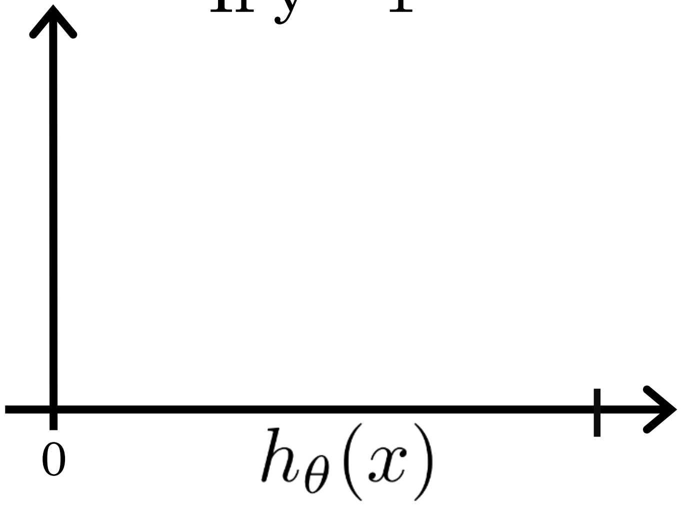


Logistic Regression Cost Function

Binary Cross Entropy Loss

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

If y = 1



$\mathrm{Cost} = 0$ if $y = 1, h_\theta(x) = 1$

But as $h_\theta(x) \to 0$

$Cost \to \infty$

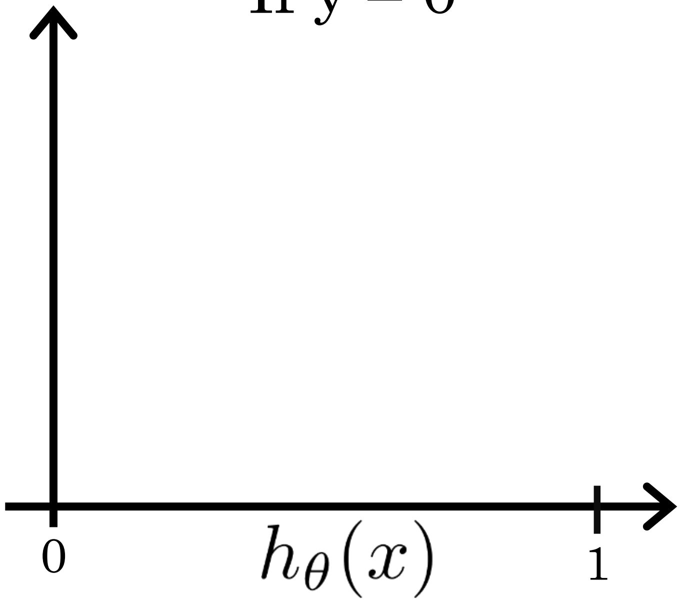
Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x; \theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.

- Loss is lowest when $h_\theta(x^{(i)})$ predicts close to true label $y^{(i)}$

Logistic Regression Cost Function

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

If y = 0



Logistic Regression

Simplifies Cost Function and Gradient Descent

Logistic regression cost function

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

❖ Our overall loss function.

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

❖ For single training example

This equation is completely equivalent to the above more complex formula.

Note: $y = 0$ or 1 always

if $y^{(i)} = 1$:

$$L(h_{\vec{\theta}}(\vec{x}^{(i)}), y^{(i)}) = -1 \log(h_\theta(\vec{x}^{(i)})) - 0$$
$$= -\log(h_\theta(\vec{x}^{(i)}))$$

if $y^{(i)} = 0$:

$$L(h_{\vec{\theta}}(\vec{x}^{(i)}), y^{(i)}) = 0 - \log(1 - h_\theta(\vec{x}^{(i)})$$
$$= -\log(1 - h_\theta(\vec{x}^{(i)}))$$

Logistic regression cost function

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

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

To fit parameters : θ

$$\min_\theta J(\theta)$$

To make a prediction given new x:

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

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

Repeat $\{$

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

$\}$

(simultaneously update all θ_j)

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

Repeat $\{$

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

$\}$ (simultaneously update all θ_j)

Algorithm looks identical to linear regression!

Gradient of Binary Cross Entropy Loss Function

Derivative of Cost Function in Logistic Regression:

While implementing logistic regression gradient descent we need to compute:
Derivative of cost function with respect to θ_0 and $\vec{\theta}$.

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

$$\rightarrow h_{\vec{\theta}}(\vec{x}) = \frac{1}{1+e^{-(\theta_0+\vec{\theta}\cdot x)}} = \sigma(\theta_0+\vec{\theta}\cdot\vec{x})$$

$$= \sigma(z)$$

$$\rightarrow z = \theta_0 + \vec{\theta}\cdot\vec{x}$$

$\frac{\partial Cost}{\partial \vec{\theta}}$? $\qquad \frac{\partial Cost}{\partial \theta_0}$?

Loss Function: $L = -\left[y^{(i)}\log(h_{\vec{\theta}}(\vec{x}^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$

$$\vec{\theta} = \begin{bmatrix}\theta_1\\ \theta_2\\ \cdots\\ \theta_n\end{bmatrix} \qquad \frac{\partial L}{\partial\theta_1}, \frac{\partial L}{\partial\theta_2}, \cdots \rightarrow \frac{\partial L}{\partial\vec{\theta}}$$

$\Rightarrow \frac{\partial L}{\partial\vec{\theta}}$ By applying chain rule

→ For simplicity I write θ.

$$\frac{\partial L}{\partial\theta} = \frac{\partial L}{\partial h_\theta(x^{(i)})} \times \frac{\partial h_\theta(x^{(i)})}{\partial\theta}$$

$$= \frac{\partial L}{\partial h_\theta(x^{(i)})} \times \frac{\partial h_\theta(x^{(i)})}{\partial z} \times \frac{\partial z}{\partial\theta}$$

Let's find all the three derivatives one by one and then multiply it to get $\frac{\partial L}{\partial\theta}$

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

$$= -\left[y^{(i)}\times\frac{1}{h_\theta(x^{(i)})} + (1-y^{(i)})\times\frac{1}{1-h_\theta(x^{(i)})}\right] \qquad \because \frac{\partial}{\partial}\log x = \frac{1}{x}$$

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

→ y is our prediction, it will be constant

→ Also find derivative of $1-h_\theta(x^{(i)})$ which is $0-1=-1$

Gradient of Binary Cross Entropy Loss Function

So - will get cancelled.

$$\frac{\partial L}{\partial h_\theta(x^{(i)})} = \frac{-y}{h_\theta(x^{(i)})} + \frac{1-y^{(i)}}{1-h_\theta(x^{(i)})}$$

$$\Rightarrow \frac{\partial h_\theta(x^{(i)})}{\partial z} = \frac{\partial}{\partial z}\left(\frac{1}{1+e^{-z}}\right)$$

$$= \frac{\partial}{\partial z}(1+e^{-z})^{-1} \qquad \therefore \text{Chain rule}$$

$$= \frac{-1}{(1+e^{-z})^2} \cdot (1+e^{-z})$$

$$= \frac{-1}{(1+e^{-z})^2} \cdot (-e^{-z}) \qquad \therefore \frac{\partial}{\partial x} e^{-x} = -e^{-x}$$

$$= \frac{e^{-z}}{(1+e^{-z})^2}$$

$$\Rightarrow h_\theta(x^{(i)})^2 = \frac{1}{(1+e^{-z})^2}$$

$$\Rightarrow e^{-z} = \frac{1-h_\theta(x^{(i)})}{h_\theta(x^{(i)})}$$

$$\Downarrow$$

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

So

$$= \frac{1-h_\theta(x^{(i)})}{h_\theta(x^{(i)})} \times (h_\theta(x^{(i)}))^2$$

$$= \frac{1-h_\theta(x^{(i)})}{\cancel{h_\theta(x^{(i)})}} \times \cancel{h_\theta(x^{(i)})} \cdot h_\theta(x^{(i)})$$

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

$$e^{-z} = \frac{1-h_\theta(x^{(i)})}{h_\theta(x^{(i)})}$$

$$= h_\theta(x^{(i)})\left(1-h_\theta(x^{(i)})\right)$$

$$\Rightarrow \frac{\partial}{\partial \theta} z = \frac{\partial}{\partial \theta}(\theta_0 + \theta_1 x)$$

$$= 0 + x$$

$$= x$$

Now

$$\frac{\partial}{\partial \theta} L = \left[\frac{-y}{h_\theta(x^{(i)})} + \frac{(1-y)}{(1-h_\theta(x^{(i)}))}\right] \times (h_\theta(x^{(i)}))(1-h_\theta(x^{(i)})) \times x$$

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

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

Gradient of Binary Cross Entropy Loss Function

$$= \left(-y + h_\theta(x^{(i)})\right)\cdot x$$

$$\frac{\partial}{\partial\theta}L = \left(h_\theta(x^{(i)}) - y\right)\cdot x$$

$$\frac{\partial L}{\partial\vec{\theta}} = \left(h_{\vec{\theta}}(x^{(i)}) - y\right)\cdot\vec{x} \quad \begin{cases} \dfrac{\partial L}{\partial\theta_1} = \left(h_\theta(x^{(i)}) - y\right)\cdot x_1 \\ \dfrac{\partial L}{\partial\theta_2} = \left(h_\theta(x^{(i)}) - y\right)\cdot x_2 \\ \vdots \\ \dfrac{\partial L}{\partial\theta_n} = \left(h_\theta(x^{(i)}) - y\right)\cdot x_n \end{cases}$$

For θ_0:

$$\frac{\partial L}{\partial\theta_0} = \frac{\partial L}{\partial h_\theta(x^{(i)})} \times \frac{\partial h_\theta(x^{(i)})}{\partial z} \times \frac{\partial z}{\partial\theta_0}$$

$$z = \theta_0 + \theta_1 x$$

$$\frac{\partial z}{\partial\theta_0} = 1$$

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

$$= \left(h_\theta(x^{(i)}) - y\right)\cdot 1$$

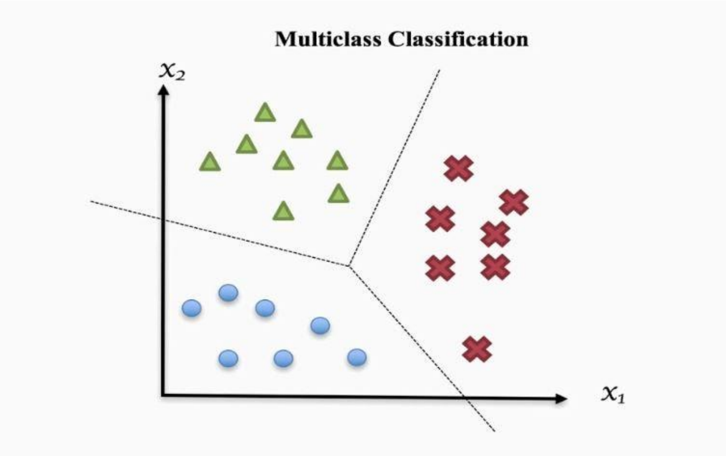
$$\frac{\partial L}{\partial\theta_0} = h_\theta(x^{(i)}) - y$$

Logistic Regression

Multi-class classification: One-vs-all

Multiclass classification

- Multi-class classification is the process of assigning each data instance to **one** class among **three or more** possible categories. Unlike binary classification, where there are only two classes, multi-class classification deals with **multiple** distinct labels.



Examples

- Classifying emails as Spam, Promotions, or Primary.
- Identifying handwritten digits (0-9)
- Categorizing images into Dogs, Cats, and Birds.

Common Algorithms:

- Logistic Regression (One-vs-Rest, Softmax Regression)
- Decision Trees & Random Forest
- Support Vector Machines (SVM, One-vs-One strategy)
- Neural Networks (Deep Learning models like CNNs for image classification)

Loss Function

- Cross Entropy Loss (Softmax Loss) is commonly used to measure prediction accuracy.

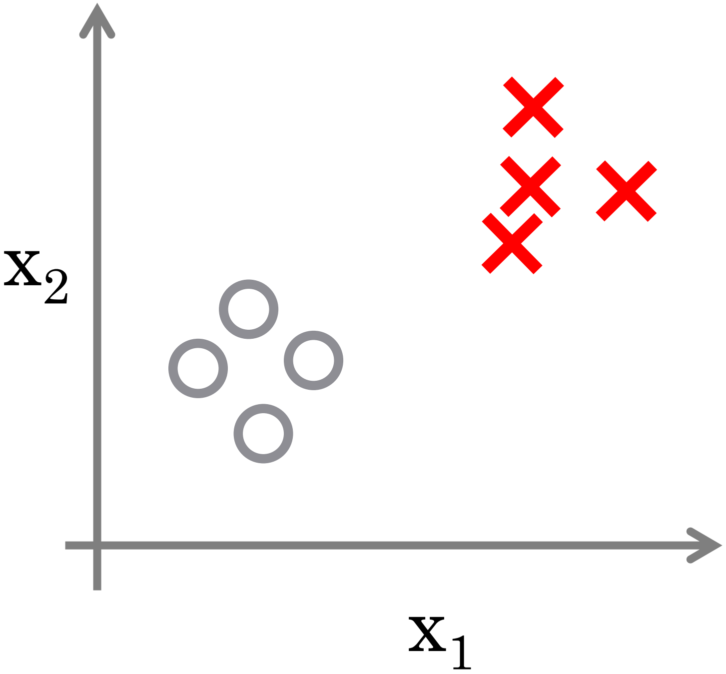
Multiclass classification

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

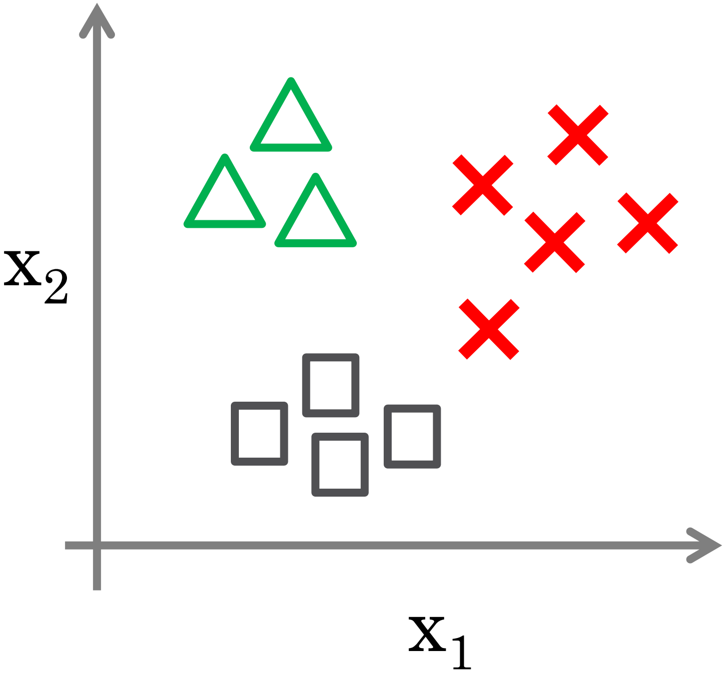
Medical diagrams: Not ill, Cold, Flu

Weather: Sunny, Cloudy, Rain, Snow

Binary classification:

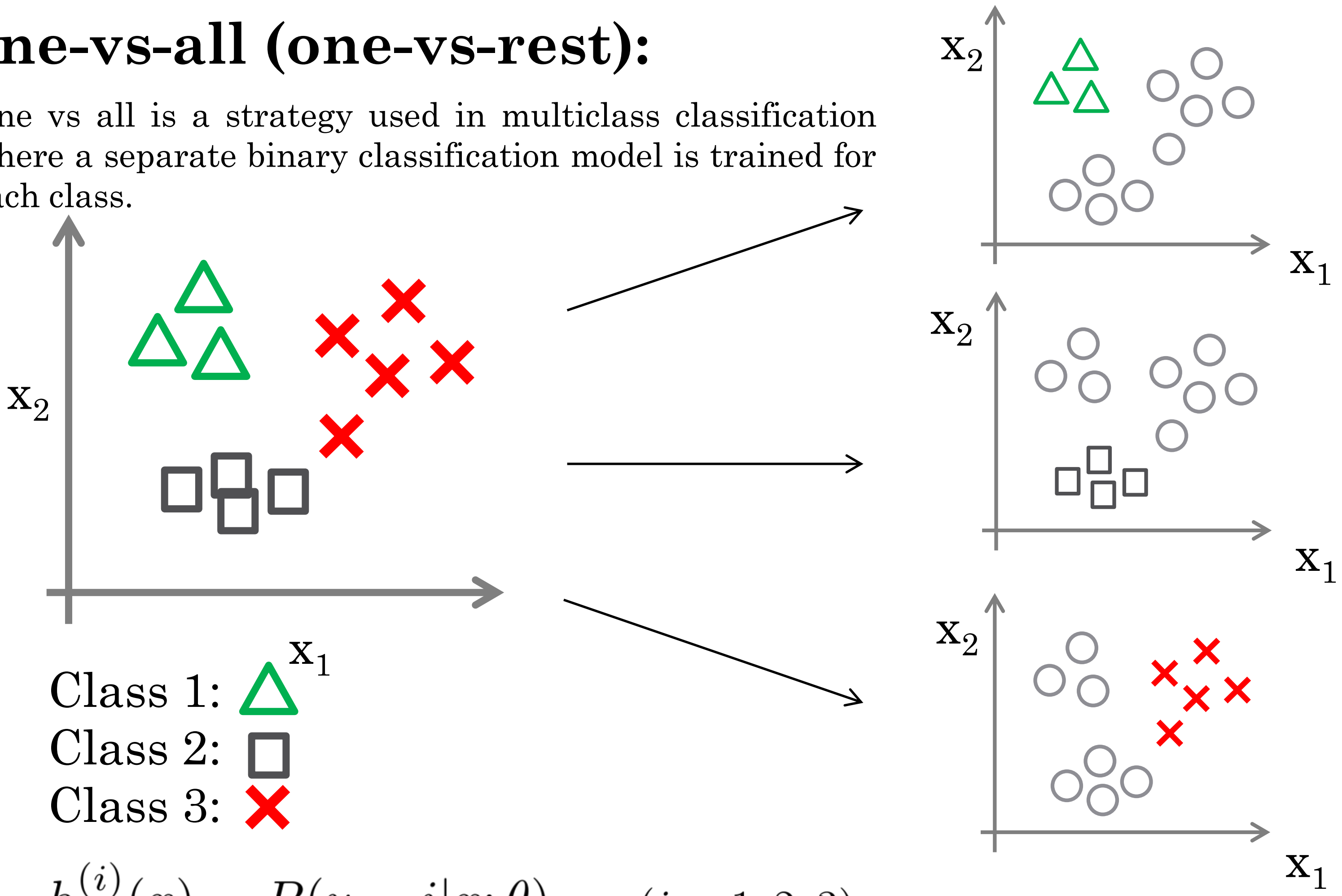


Multi-class classification:



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

One vs all is a strategy used in multiclass classification where a separate binary classification model is trained for each class.



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

Newton’s Method

Newton’s Method

Newton’s Method is a second-order optimization algorithm that uses both the gradient and the Hessian matrix to find the optimal solution.

Key Points:

- Uses **second-order derivatives** (Hessian matrix) for optimization.
- Faster convergence than **Gradient Descent** but computationally expensive.
- Used in logistic regression, SVMs, neural networks, and nonlinear optimization.

Formula:

$$w = w - H^{-1} \nabla J(w)$$

where:

- H = Hessian matrix (second derivatives of the cost function)
- $J(w)$= gradient of the cost function

Newton’s Method vs. Gradient Descent

Feature	Newton’s Method	Gradient Descent
Uses	Second-order derivatives	First-order derivatives
Convergence	Faster	Slower
Computational Cost	Higher (Hessian inversion)	Lower
Step Size Adjustment	Not required	Requires tuning
Suitable for Large Datasets	No	Yes

Applications:

- **Logistic Regression** (Newton-Raphson method)
- **Support Vector Machines (SVMs)**
- **Neural Network Training (L-BFGS method)**

Key Takeaway:

- Newton’s Method is **faster but computationally expensive** compared to Gradient Descent. Suitable for small to medium-sized problems where quick convergence is needed.

Thank You