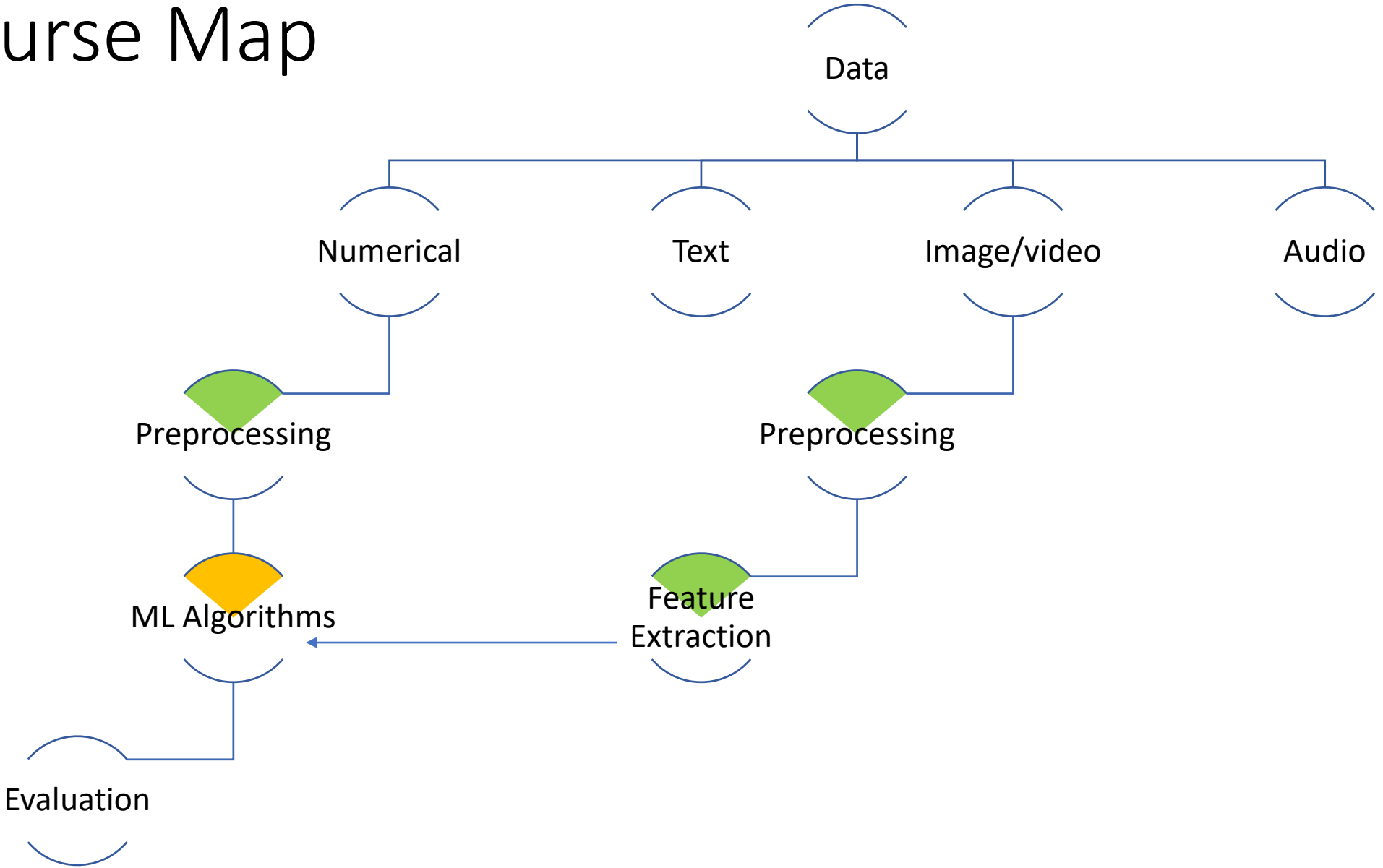
AI 330: Machine Learning

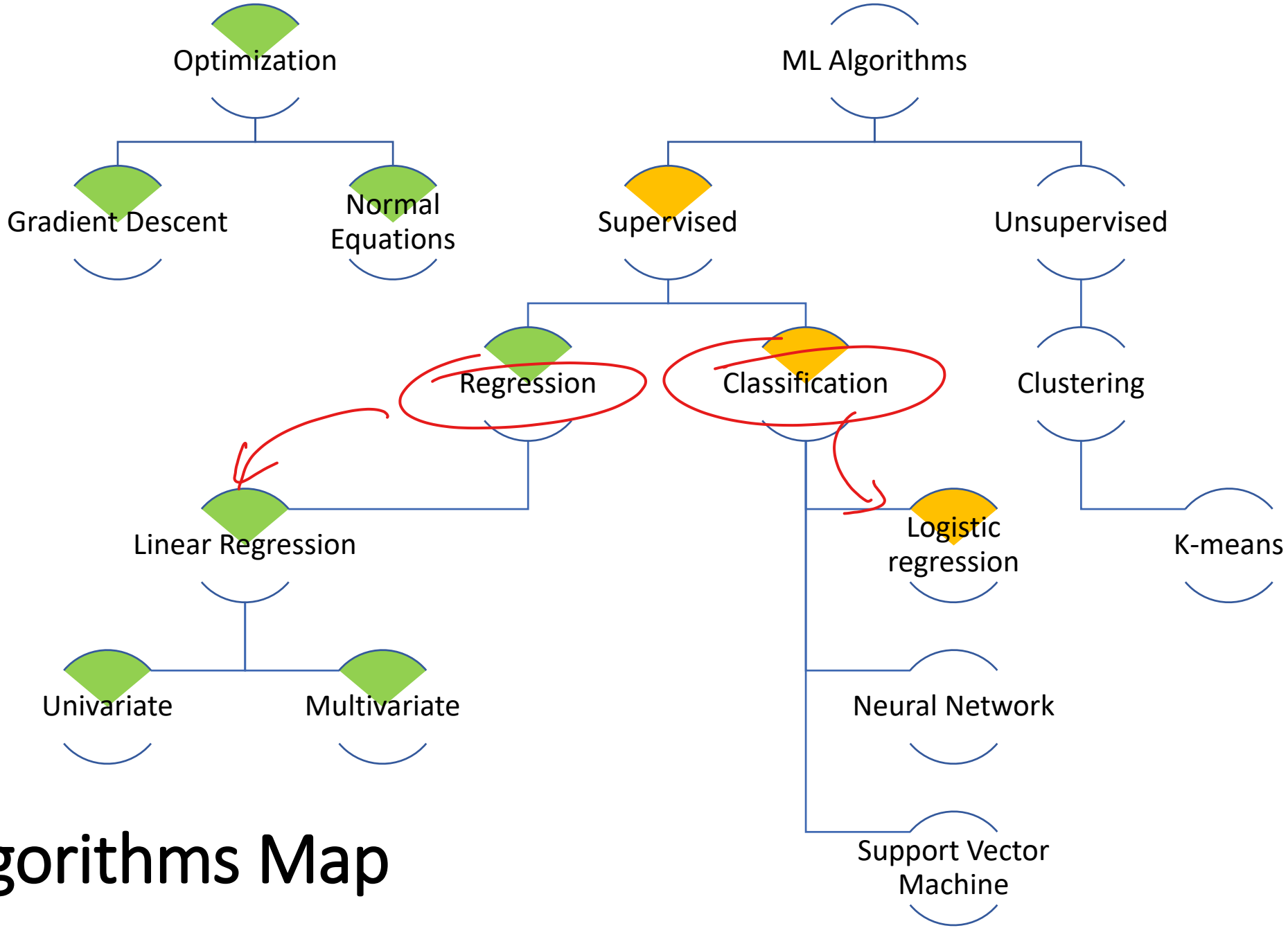
Fall 2023

Dr. Wessam EL-Behaidy
Associate Professor, Computer Science Department,
Faculty of Computers and Artificial Intelligence,
Helwan University.

Dr. Ensaf Hussein
Associate Professor, Computer Science Department,
Faculty of Computers and Artificial Intelligence,
Helwan University.

Course Map


Data
Numerical
Text
Image/video
Audio
Preprocessing
Preprocessing
ML Algorithms
Feature Extraction
Evaluation



ML Algorithms Map

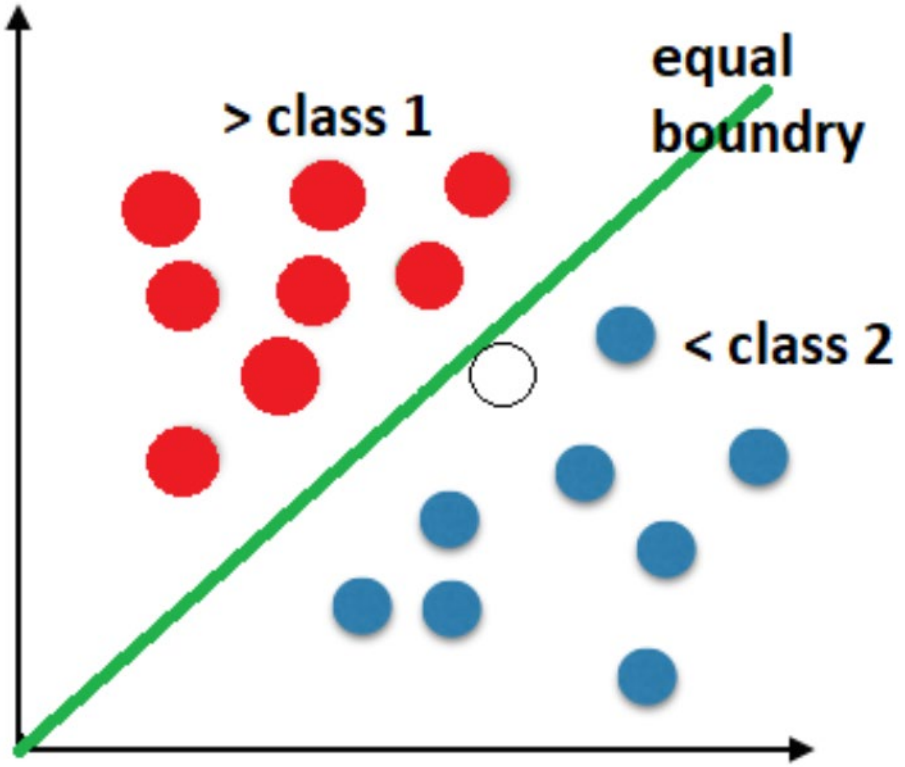
Lecture 6
Logistic Regression
&
Regularization

Slides of:
https://www.coursera.org/learn/machine-learning at Stanford University (Prof. Andrew Ng)

Classification Problem:
Logistic Regression

Classification

- Discrete outcomes.
- Binary $y \in \{0,1\}$, 0 negative , 1 positive (normal / abnormal)
- Multi-class: a telescope that identifies whether an object in the night sky is a galaxy, star, or planet. $y \in \{0,1,2,3\}$



- Logistic regression is a statistical model used for binary classification tasks, where the goal is to predict whether an outcome belongs to one of two possible classes (e.g., yes/no, 1/0). It's called "logistic" because it utilizes the logistic function to model the probability of an event occurring.

- In logistic regression, the input features are linearly combined with weights, and the result is transformed using the logistic (sigmoid) function, which maps the linear combination to a value between 0 and 1. This value represents the probability of the event being in the positive class.

- Mathematically, the logistic regression model can be expressed as:

- P(Y=1|X) = 1 / (1 + e^-(b0 + b1X1 + b2X2 + ... + bn*Xn))

- Here:
  - P(Y=1|X) is the probability of the event Y being in class 1 given the input features X.
  - b0, b1, b2, ..., bn are the model parameters (coefficients) to be learned from the training data.
  - X1, X2, ..., Xn are the input features.

the threshold is a value that is used to classify the predicted probabilities into two classes. Logistic regression predicts the probability that an instance belongs to a particular class, and the threshold is the point at which you decide which class the instance should be assigned to based on its predicted probability

- The model is trained by optimizing these parameters using techniques like maximum likelihood estimation. Once trained, it can be used to make predictions by classifying instances based on the calculated probabilities. Typically, a threshold (e.g., 0.5) is chosen, and if the predicted probability is above the threshold, the instance is classified as the positive class; otherwise, it's classified as the negative class.

- Logistic regression is widely used in various fields, such as healthcare, finance, and marketing, for tasks like spam detection, customer churn prediction, and medical diagnosis.

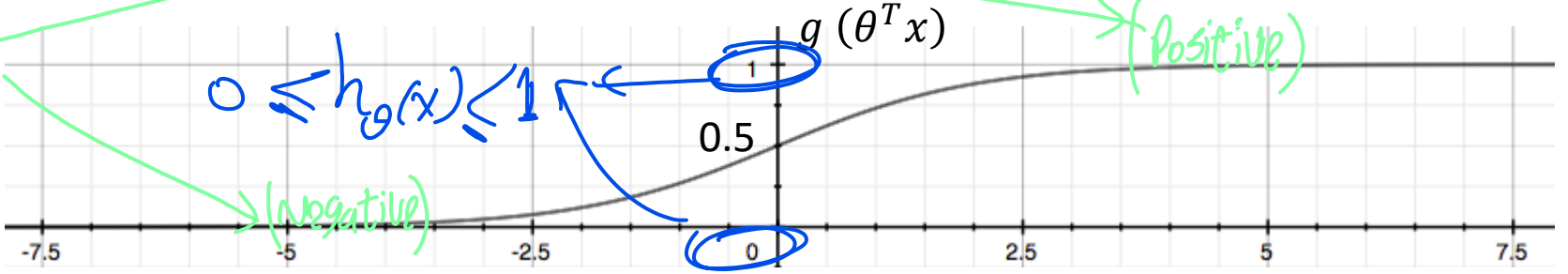
Hypothesis Representation

- Logistic regression model

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

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

Sigmoid Function
Logistic Function



- We want our classifier to output values between 0 and 1
  - When using linear regression we did $h_\theta(x) = (\theta^T x)$
  - For classification hypothesis representation we do $h_\theta(x) = g((\theta^T x))$

Interpretation of hypothesis Output

- $h_\theta(x)$ will give us the **probability** that our output is 1.

For example,

$h_\theta(x) = 0.7$ gives us a probability of 70% that our output is 1.

- Our probability that our prediction is 0 is just the complement of our probability that it is 1

For example,

if probability that it is 1 is 70%, then

the probability that it is 0 is 30%.

Interpretation of hypothesis Output

- $h_\theta(x)$ will give us the **probability** that our output is 1.

For example,

$x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumourSize} \end{bmatrix}$, $h_\theta(x)$=0.7… $y = 1$

➔ 70% chance of a tumor being malignant.

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

Probability that $y = 1$, given x, parameterized by θ

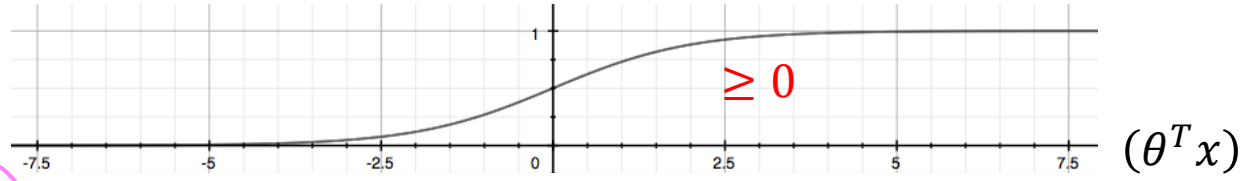
Decision Boundary

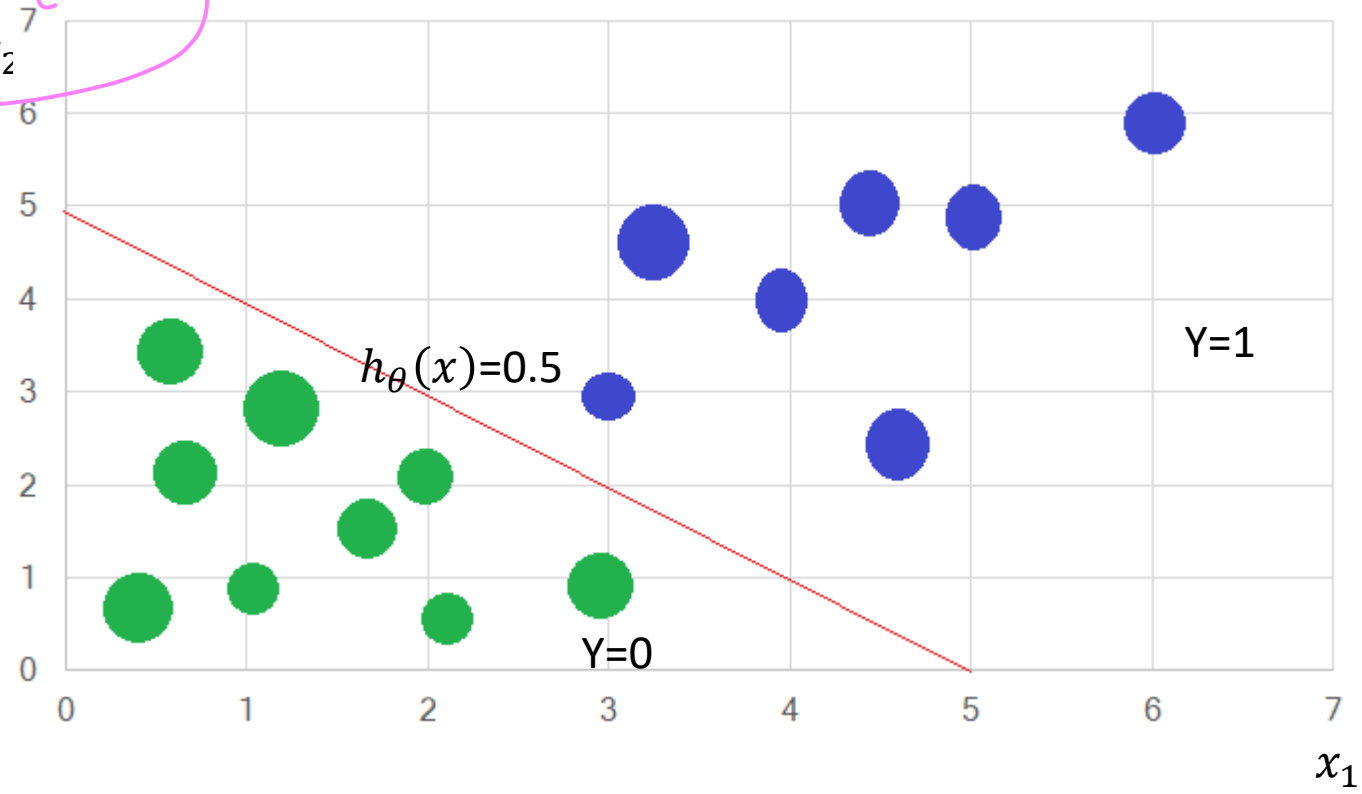
$$h_\theta(x) = g\,(\theta^T x)$$
$$= \quad g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$
$$= g(-5 + 1x_1 + 1x_2)$$

- $\theta = \begin{bmatrix} -5 \\ 1 \\ 1 \end{bmatrix}$

$[-5\ 1\ 1] \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$

$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = [\theta_0 \theta_1 \theta_2] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$





$Predict\ y = 1\ if\ -5 + x_1 + x_2 \geq 0$

$x_1 + x_2 \geq 5$

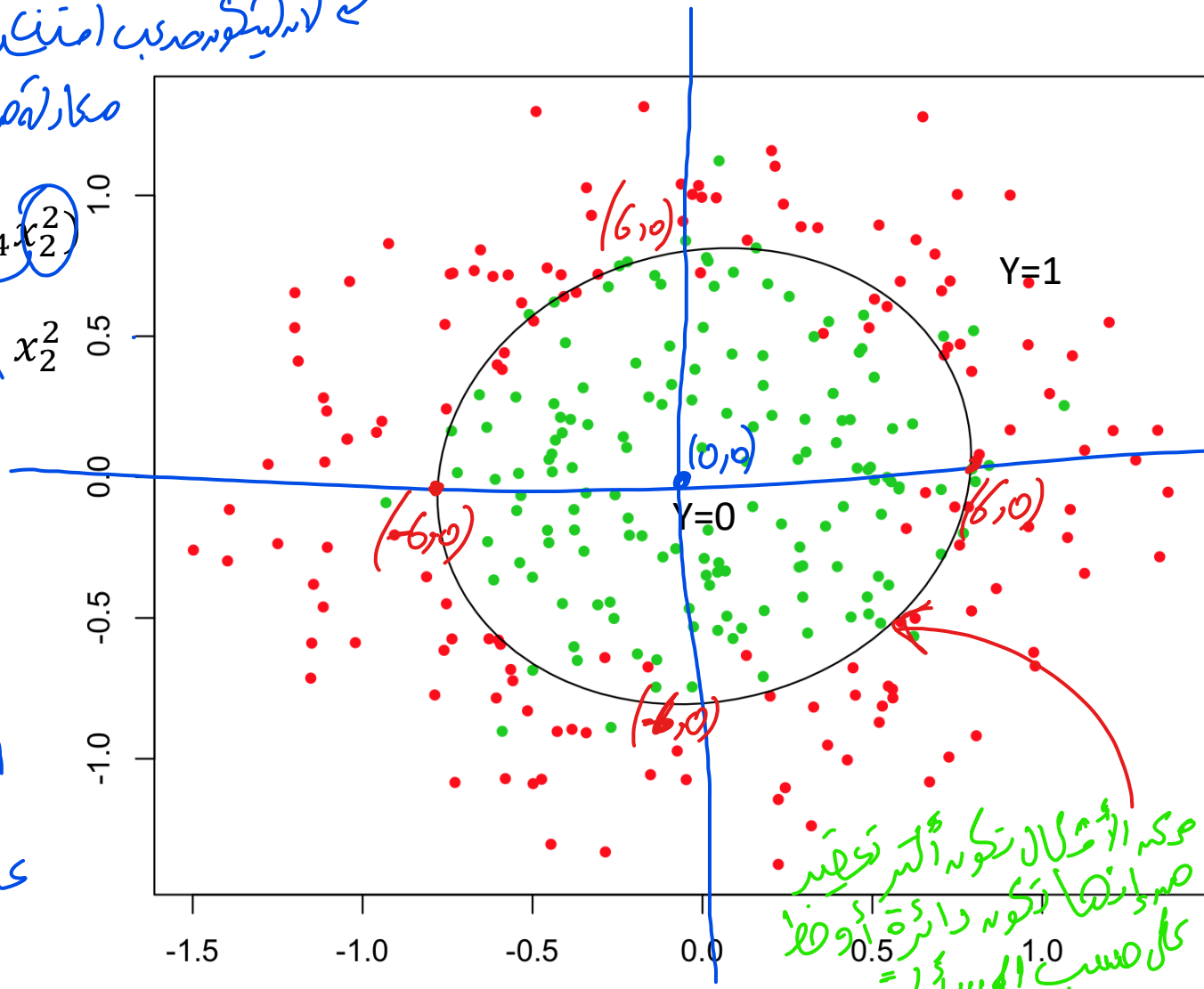
Non-linear decision boundaries

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

$$= -0.6 + 0\,x_1 + 0\,x_2 + 1\,x_1^2 + 1\,x_2^2$$

$$y = 1, if \; -0.6 + x_1^2 + x_2^2 \geq 0$$

$$x_1^2 + x_2^2 \geq 0.6$$



loss → J(θ) ← for Prediction (linear regression)

Not good for classification

- The cost function for logistic regression, often referred to as the "log loss" or "cross-entropy loss," measures how well a logistic regression model is performing in terms of its ability to predict the probability of an event belonging to a particular class. The goal is to minimize this cost function during the training of the model.

Y=0 و Y=1 بستخدم معادلتين واحدة classification لل

- he cost function is defined as follows:

- Cost(b0, b1, b2, ..., bn) = -1/m * Σ [y * log(P) + (1 - y) * log(1 - P)]

- Here:
  - "Cost" is the cost function.
  - "b0, b1, b2, ..., bn" are the model parameters (coefficients).
  - "m" is the number of training examples.
  - "Σ" represents the summation over all training examples.
  - "y" is the actual class label (0 or 1) for a given example.
  - "P" is the predicted probability that the example belongs to class 1, as calculated by the logistic regression model.

- The cost function penalizes the model more when its predicted probability is far from the actual class label. If the model's prediction is close to the true class label, the log loss is small; if it's far off, the log loss becomes significantly larger.

cost function

بتحاول تقلل القيمة قدر الإمكان

- During the training process, the model's parameters (b0, b1, b2, ..., bn) are adjusted to minimize this cost function. Techniques like gradient descent or other optimization methods are used to find the values of the parameters that lead to the lowest cost.

- In logistic regression, the ultimate goal is to find the parameter values that result in the lowest cost, indicating that the model's predicted probabilities are as close as possible to the true class labels for the training data. This ensures that the model makes accurate predictions on new, unseen data.

How to choose parameter θ- Cost Function

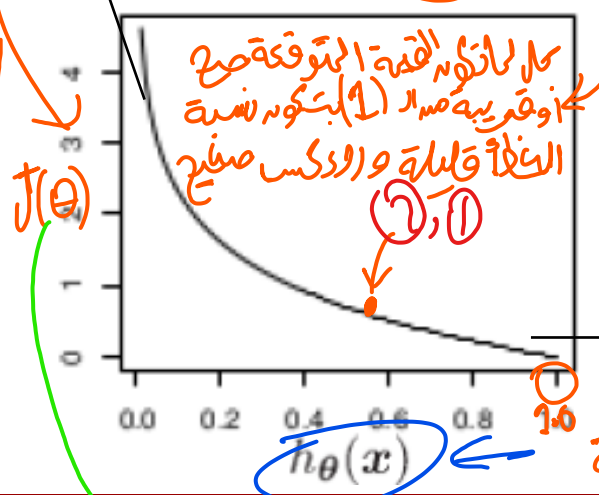
- $Training\ set: \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$

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

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \cdot \\ \cdot \\ x_n \end{bmatrix} \qquad x_0 = 1, y \in \{0,1\}$$

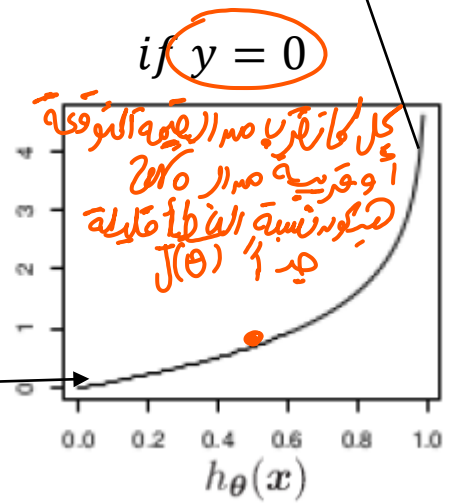
$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & if\ y = 1 \\ -\log(1 - h_\theta(x)) & if\ y = 0 \end{cases}$$

$Cost \to \infty, y \to 0$ if $y = 1$



$Cost = 0, y = 1$

$Cost \to \infty, y \to 1$ if $y = 0$



$Cost = 0, y = 0$

Simplified Cost Function

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

when y is equal to 1, then the second term will be zero and will not affect the result.

If y is equal to 0, then the first term will be zero and will not affect the result.

- Cost Function of Logistic regression using training set:

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

Gradient Descent for Logistic Regression

- **Cost Function of Logistic regression using training set:**

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

- **Using Gradient Descent (GD) to $min_\theta\ J(\theta)$:**

Repeat {

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

}

Repeat until convergence : {

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

, simultaneously updates θ_j

for every $j = 0,1,2, \ldots, n$

}

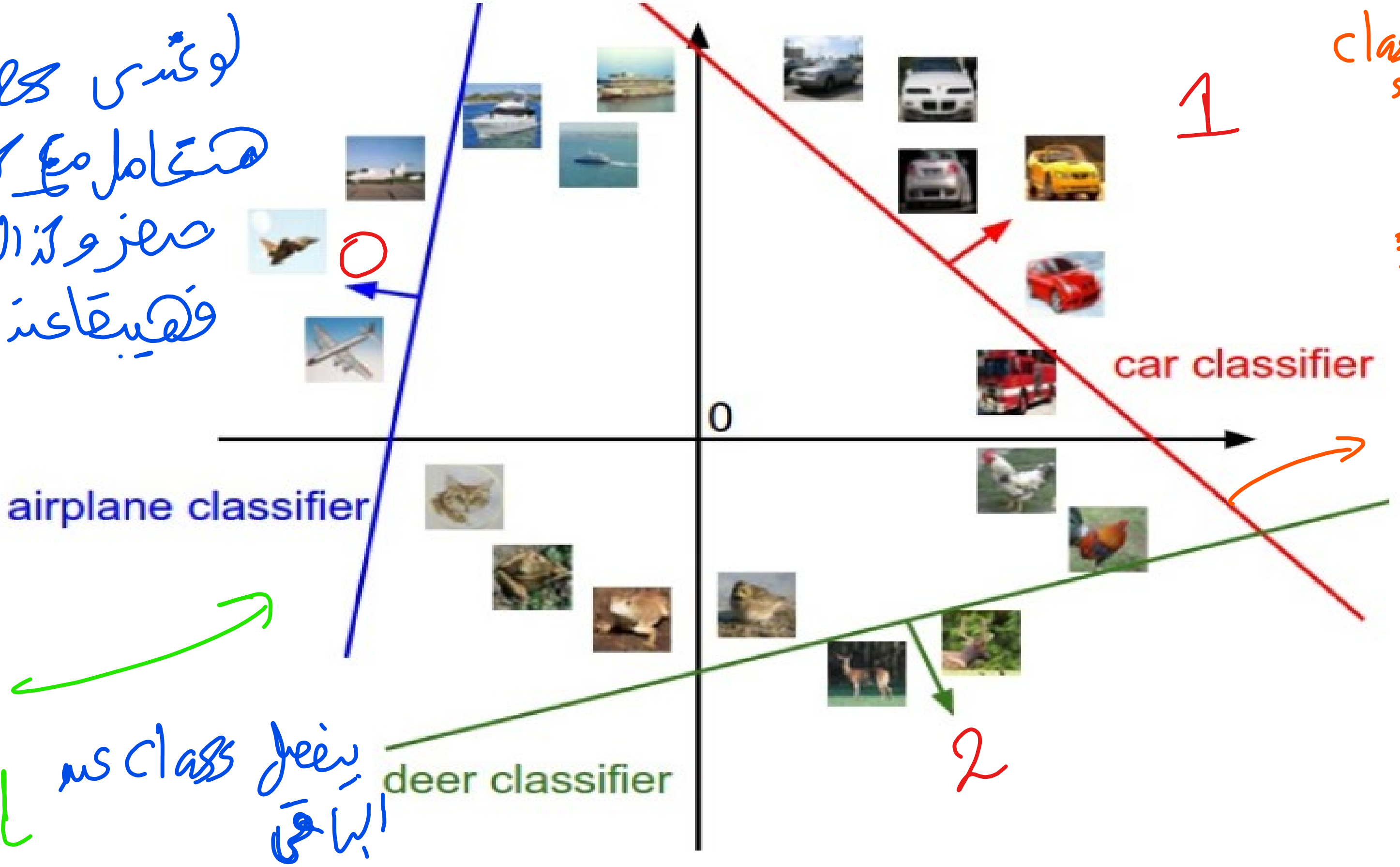
$$\textbf{where}, h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Multiclass Classification

- Train a logistic regression classifier $h_\theta(x)$ for each class

مش هنستخدم 0 و 1 بس هيكون 2, 3, 4, ...

لو عندي 3 classes
هتعامل مع كل كلاس على إنه 1 والباقي
صفر وكذا لكل class
فهيبقا عندي 3 lines بيفرقوا ما بينهم



هنشوف ال item ذا class 1
بال classifier الأول لو لأ
هنستخدم ال classifier التاني
عشان اشوف هو ال 2 ولا 3

كل line هو عبارة عن
equation وكل line
بيقولك هو ده ال class
بتاعها ولا لا

OVA

one Verses all

بيفصل ال class عن الباقي

يعني ال class ناحية (1) والباقي كلهم ناحية تانية (0)

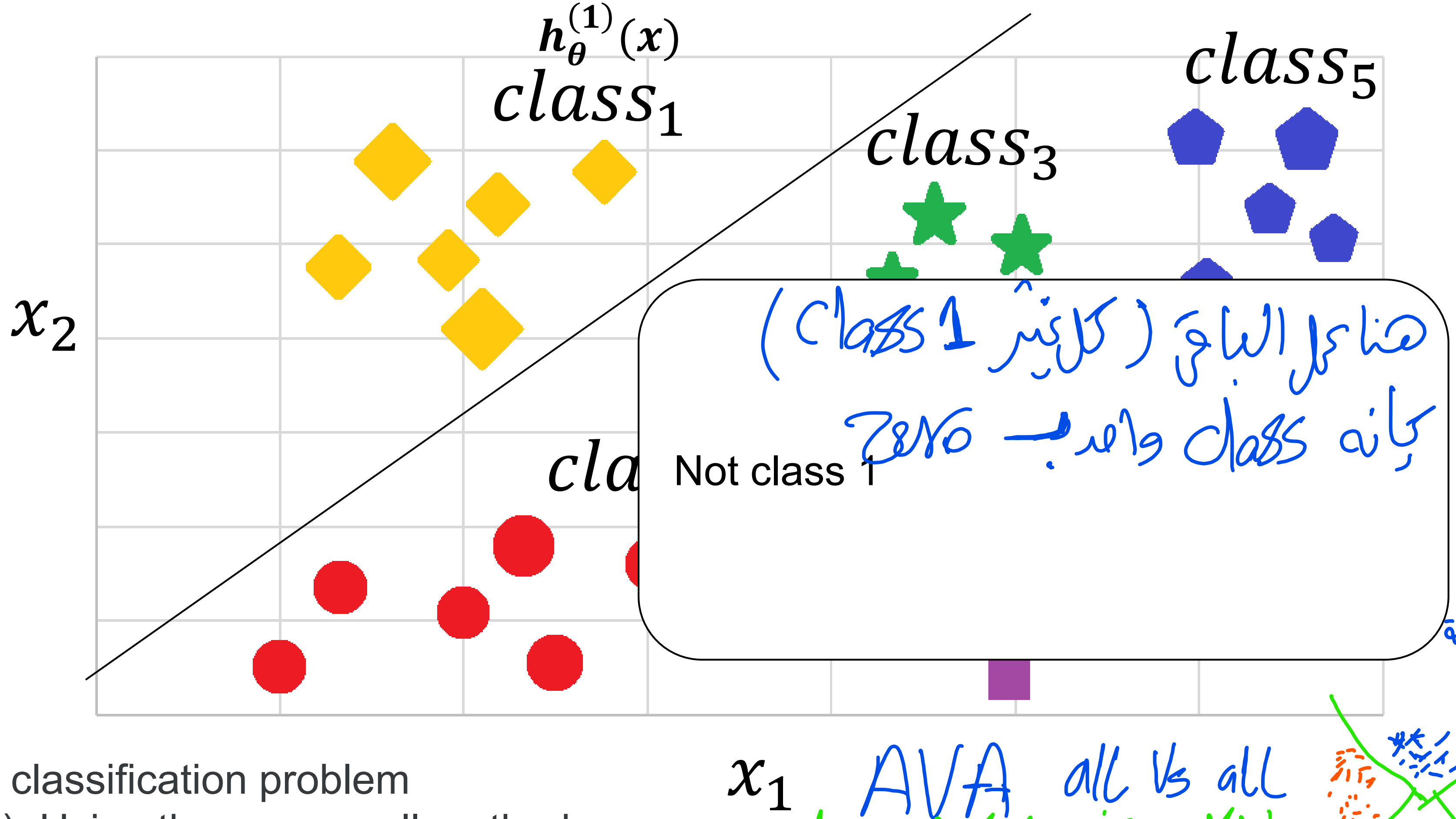
Multiclass Classification (one-vs-all)

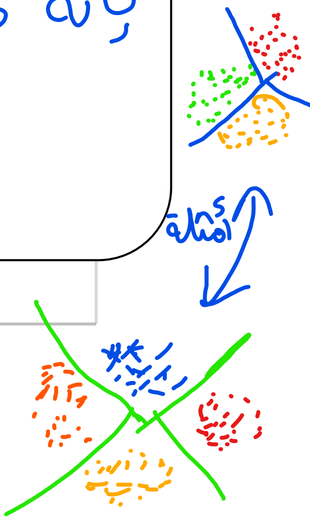
$$h_\theta^{(i)} = P(y = i|x;\theta)$$

$$i = 1, 2, 3 \ldots$$

Pick the class i that maximize

$$prediction = max_i\ (h_\theta^{(i)})$$

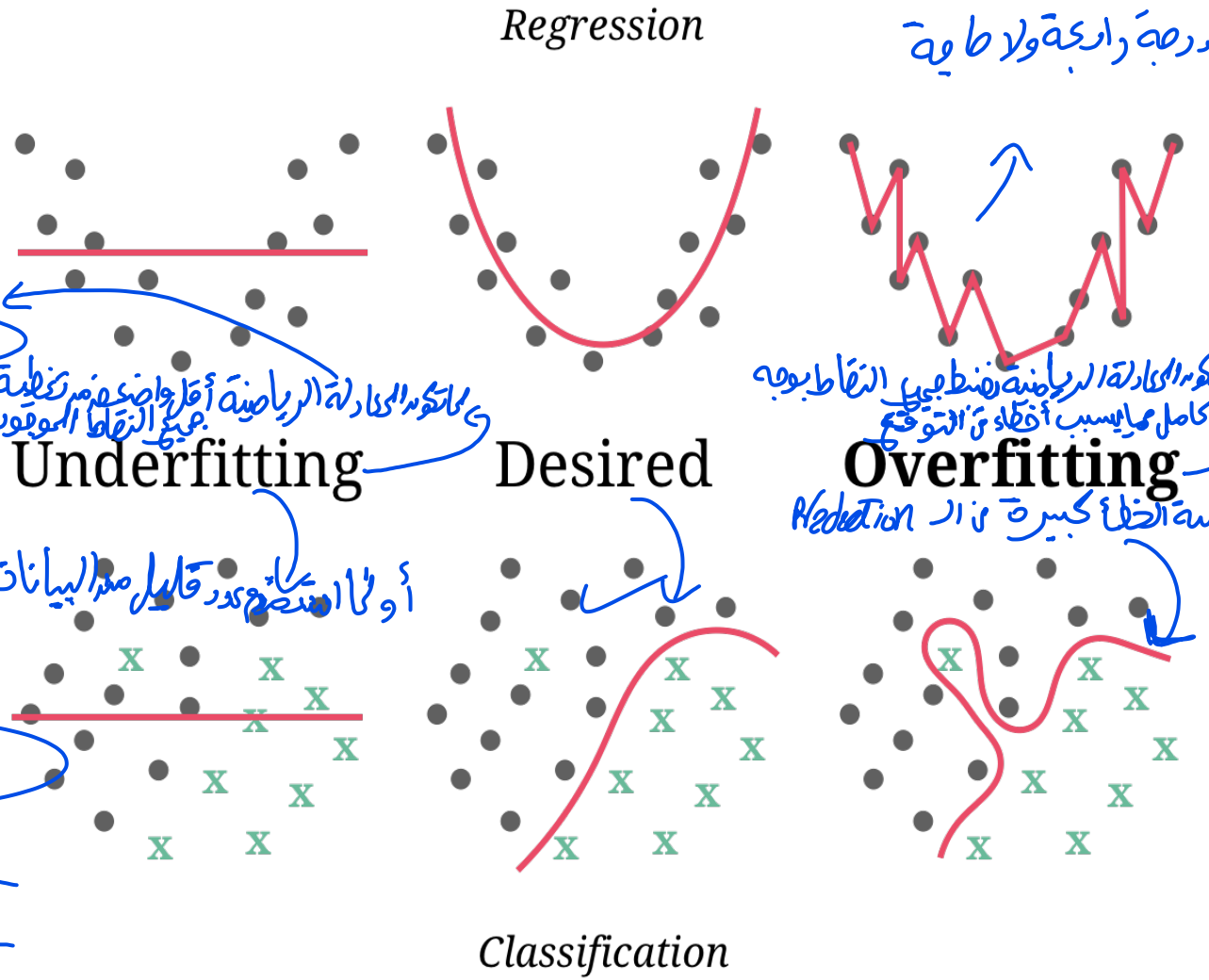




Suppose you have a multi-class classification problem with k classes (so $y \in \{1,2 \cdots, k\}$). Using the one-vs.-all method, how many different logistic regression classifiers will you end up training? K

The Problem of Overfitting

- **Underfitting**, or **high bias**, is when the form of our hypothesis function h maps poorly to the trend of the data. It is usually caused by a function that is too simple or uses too few features.
- At the other extreme, **overfitting**, or **high variance**, is caused by a hypothesis function that fits the available data but **does not generalize** well to predict new data. It is usually caused by a complicated function that creates a lot of unnecessary curves and angles unrelated to the data.



Addressing overfitting

- There are two main options to address the issue of overfitting:

1) Reduce the number of features:

- Manually select which features to keep.
- Use a model selection algorithm.

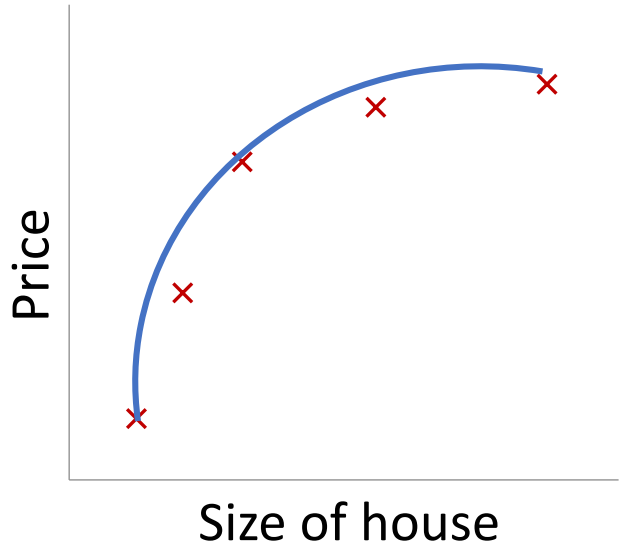
2) Regularization

- Keep all the features, but reduce the magnitude of parameters θ_j.
- Regularization works well when we have a lot of slightly useful features.

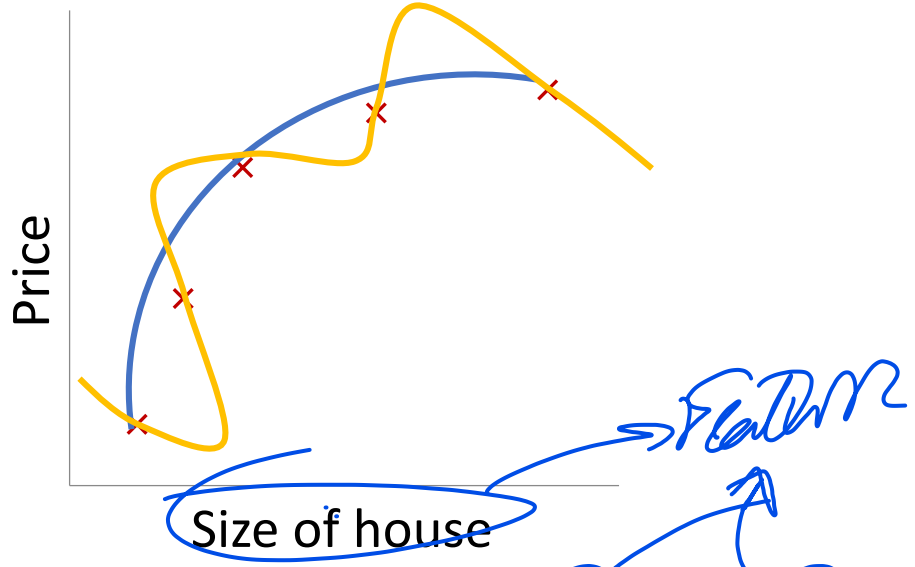
Regularization (To avoid overfitting)

Regularization for Linear Regression

Regularization Intuition



$$\theta_0 + \theta_1 x + \theta_2 x^2$$



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

- Suppose we penalize and make θ_3, θ_4 **really small**.
- **Small values for parameters** $\theta_0, \theta_1, \ldots, \theta_n$:
  - simpler hypothesis
  - less prone to overfitting

overfitting

Regularization for linear Regression

- **Simpler hypothesis ➔ small values of $\theta_1, \theta_2, \ldots \theta_n$.**

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

choose small θ to min $J(\theta)$

- The λ, or lambda, is the **regularization parameter**. $0 \le \lambda \le large\ number$
  It determines how much the costs of our theta parameters are inflated.

- **Example:** we have 2 sets of parameters θ=[1.35 3.5] and θ=[45.2 75.6]
  - **If λ is chosen to be 0**, the cost function act as usual with no penalty on θ
    ➔ choose the large ones θ=[45.2 75.6]
  - **If λ is chosen to be large**, small values of θ are chosen instead of large one.
    ➔ choose the large ones θ=[1.35 3.5]

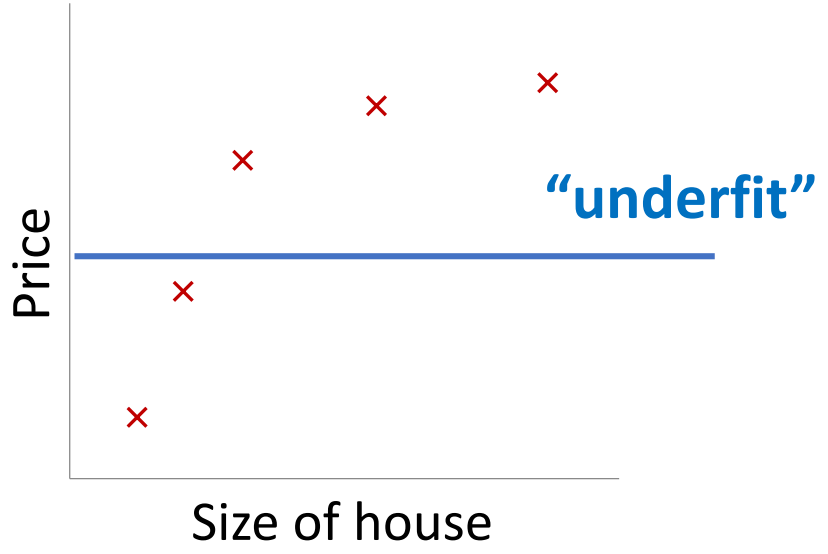
Regularization for linear Regression

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

- What if λ is set to an **extremely large value** (perhaps too large for our problem, say **λ= 10^{10}**)?

➔ it may smooth out the function too much and cause ***underfitting.***

Why?



$$\theta_1 \approx 0, \theta_2 \approx 0, \theta_3 \approx 0, \theta_4 \approx 0$$

➔ $h_\theta(x) = \theta_0$

$$\theta_0 + \cancel{\theta_1 x} + \cancel{\theta_2 x^2} + \cancel{\theta_3 x^3} + \cancel{\theta_4 x^4}$$

Regularization for linear Regression

- **Gradient descent**

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)} \qquad j \notin \{0\}$$

$$\theta_j := \theta_j - \alpha \left[\left(\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \right] \qquad j \in \{1, 2 \ldots n\}$$

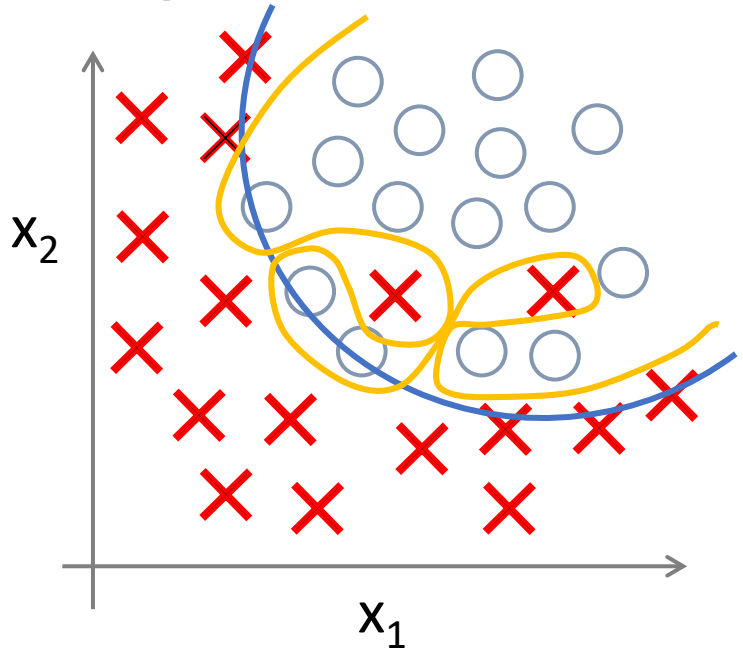
}

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

Regularization (To avoid overfitting)

Regularization for Logistic Regression

Regularization for Logistic Regression



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

Cost function:

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

Small values of parameters

Regularization for Logistic Regression

- **Gradient descent**

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \left[\left(\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \right] \qquad j \in \{1, 2 \ldots n\}$$

}

$$j \notin \{0\}$$

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

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

Thanks