SWINBURNE UNIVERSITY OF TECHNOLOGY

COS30082
Applied Machine Learning

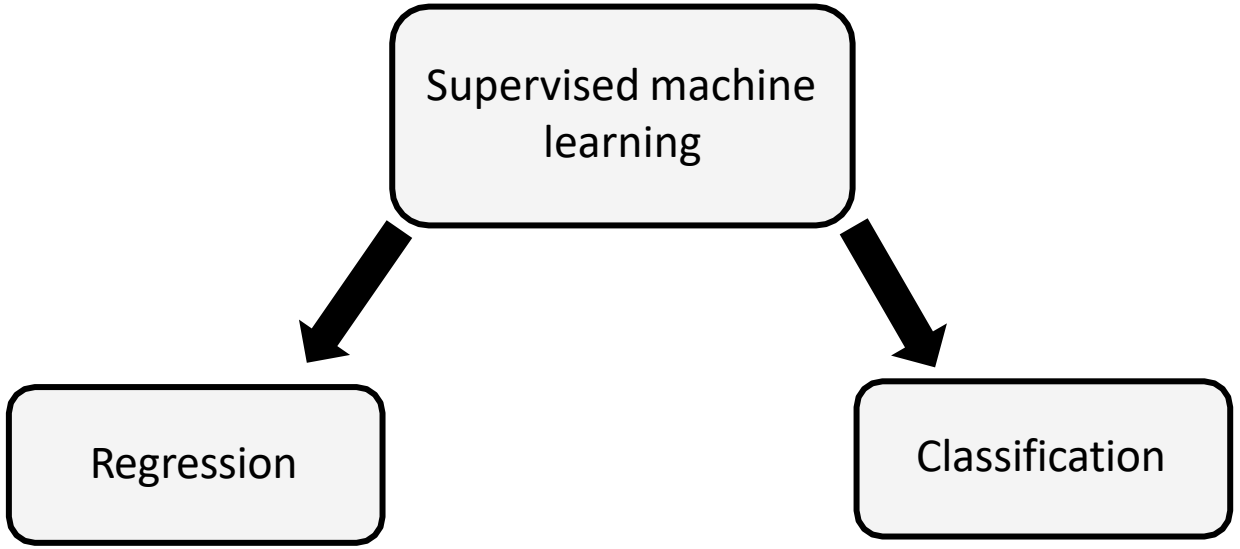
Lecture 3

Logistic Regression

Topics

- Understanding of logistic regression problem
  - What is the difference between linear and logistic regression
- To find out how logistic regression works
- To learn logistic regression optimization techniques
- To solve a machine learning problem using a logistic regression algorithm

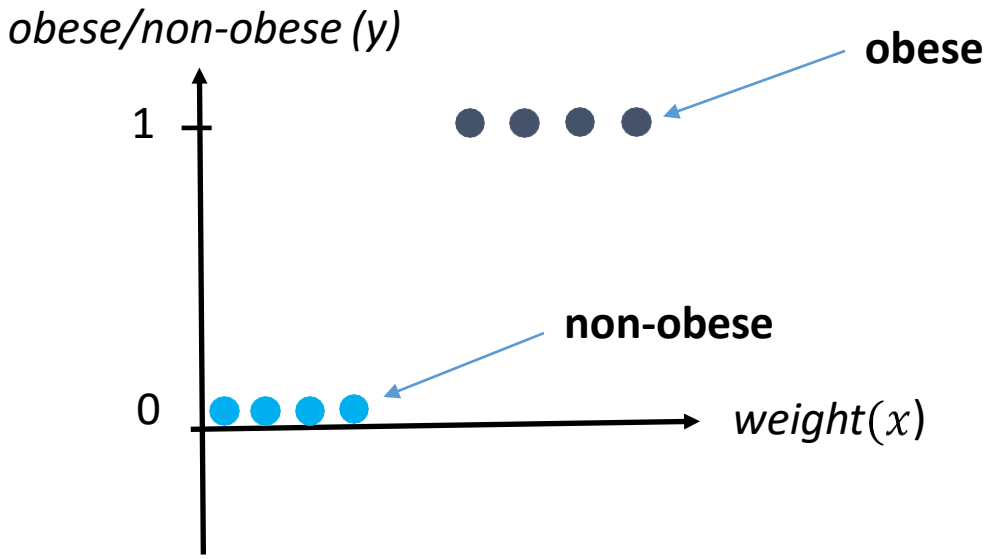
Regression vs Classification



- What is the price of the houses?
- What is the height of the students?

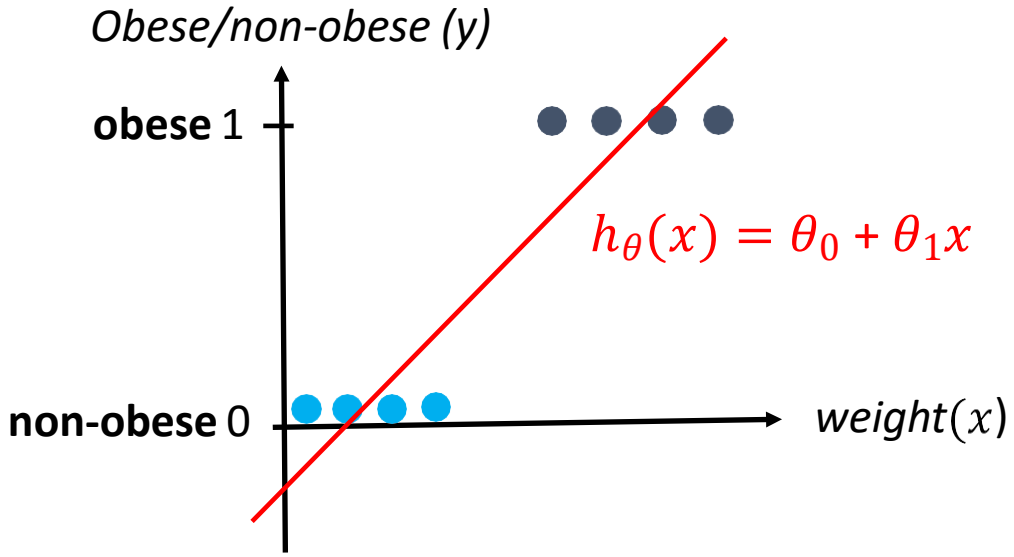
- Is the person boy or girl?
- Is the email spasm or not spasm?
- Is she happy or not?

Linear classification using Hard Threshold



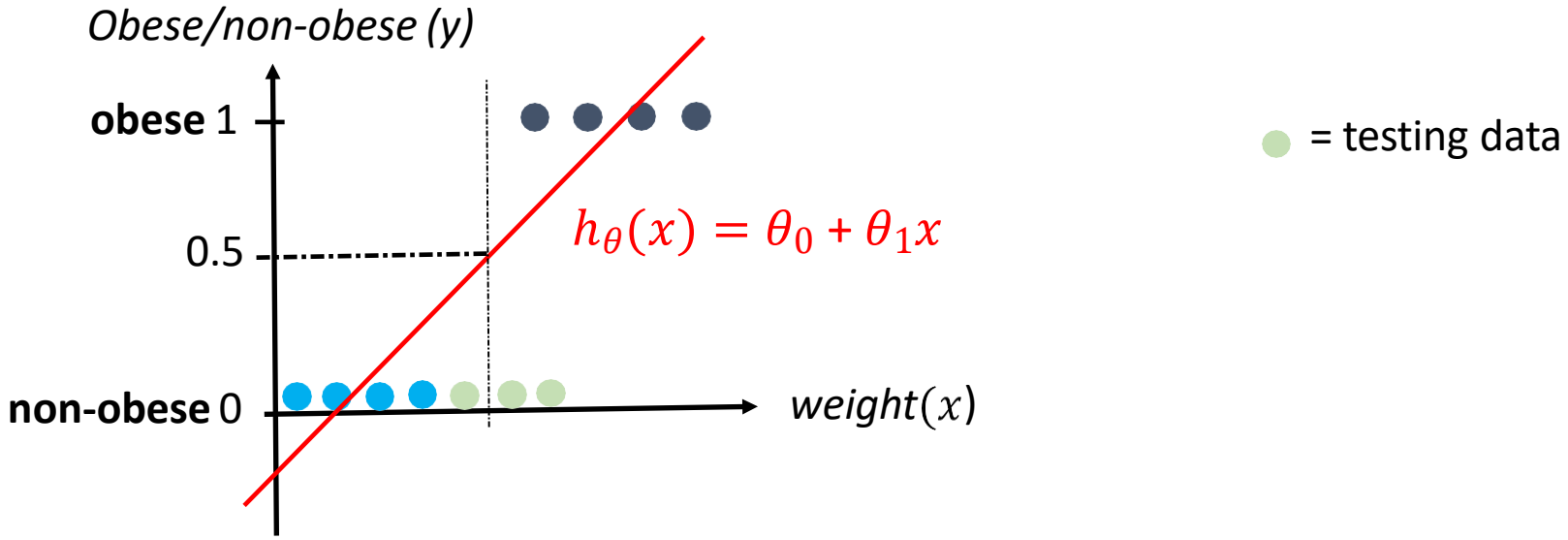
- For example, from a series of N training set, we want to model the relationship between the *obesity level* and *weight.*
- The obese patient is labeled as 1 and the non-obese patient is labeled as 0.

Linear classification using Hard Threshold



- This red line shows the **linear regression model** that maps the independent variables (*weight*) to the dependent variable (*obesity level*).
- It learns the best fit line to minimize the distance between the predicted value $h_\theta(x)$ and actual value y .

Linear classification using Hard Threshold

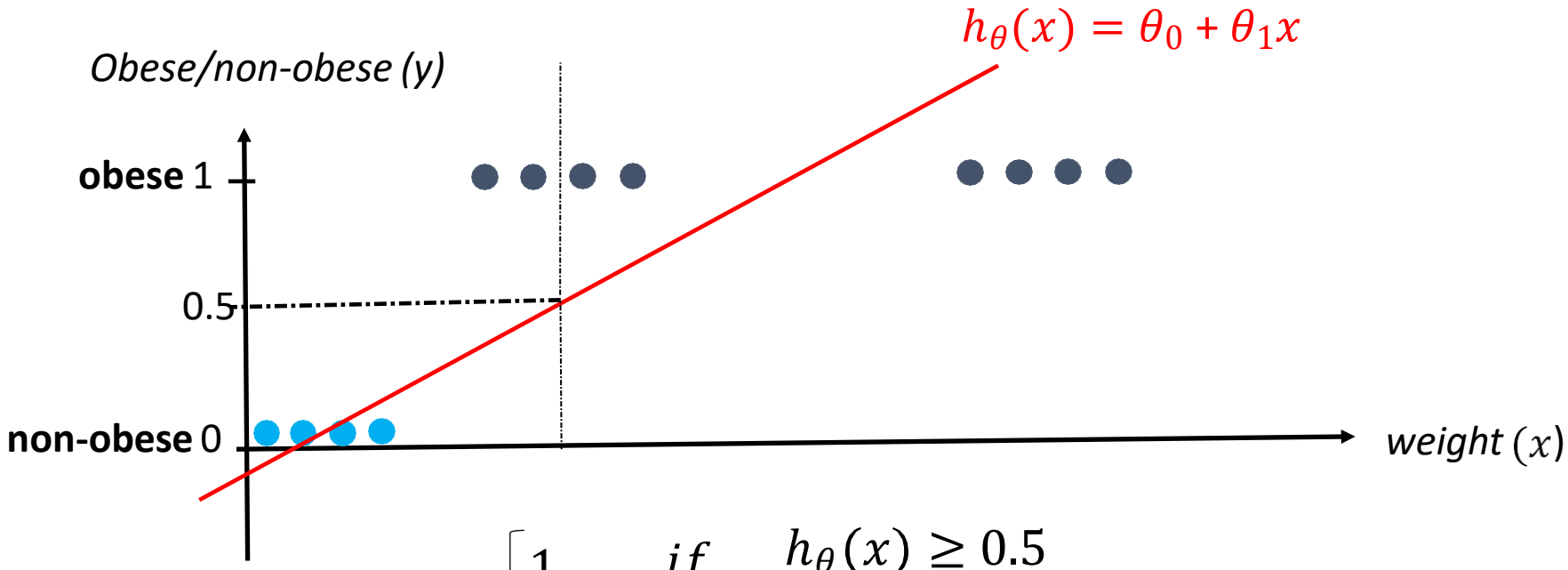


- Using a linear regression model, the predicted value $h_\theta(x)$ can be classified into a real value, y (*0* or *1*) based on **hard threshold**.
- For example: threshold $h_\theta(x)$ at 0.5

$$y = \begin{cases} 1 & if \quad h_\theta(x) \geq 0.5 \\ 0 & else \quad h_\theta(x) < 0.5 \end{cases}$$

Linear classification using Hard Threshold

- However, linear regression model used for classification is sensitive to **imbalance data**.



$$y = \begin{cases} 1 & if \quad h_\theta(x) \geq 0.5 \\ 0 & else \quad h_\theta(x) < 0.5 \end{cases}$$

Linear classification using Hard Threshold

- In addition, the prediction is **continuous but not probabilistic**.
  - It is possible that $h_\theta\ (x) < 0$ and $h_\theta\ (x) > 1$.
- For classification, we want to model the probability of y being **0 or 1** based on a probabilistic model, $h_\theta\ (x) = \mathrm{P}\ (y = 1 | x; \theta)$.
  - Hence, the prediction should fall within $1 \leq h_\theta\ (x) \leq 0$.

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

- This represents the probability that the output y is equal to 1 given the input features x and the model parameters θ.
- It indicates that the hypothesis function is a function of both the input features x and the model parameters θ.

Introduction of Logistic Regression

- **Logistic Regression** is one of the machine learning algorithms used for **classification** problems.
- It predicts the probability of a **categorical** dependent variable y, which could be represented by binary values, 0 or 1, true or false, yes or no.
- In other words, the logistic regression model predicts $\mathrm{P}\,(y = 1\,|x; \theta)$ as a function of x, given parameters of θ.

Logistic regression

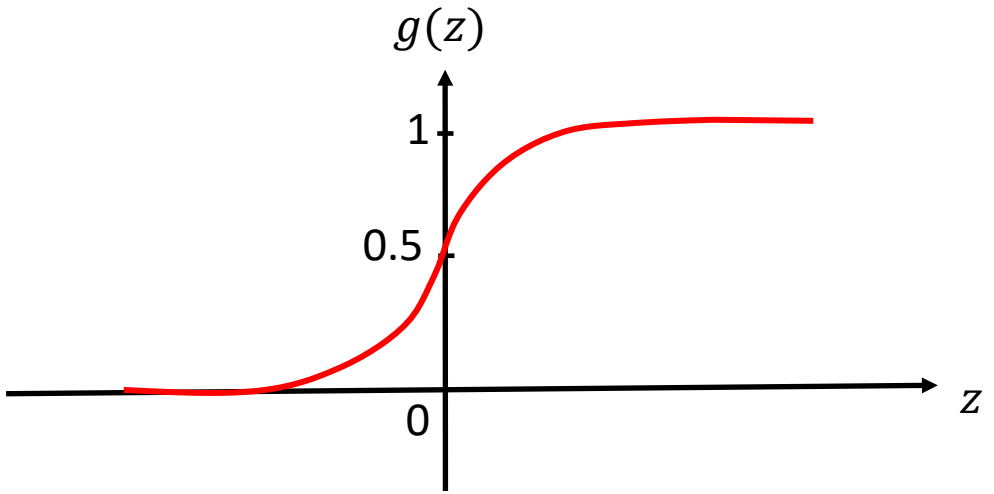
- To ensure hypothesis $h_\theta\ (x)$ lies between 0 and 1:

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

- $z = \theta_0 + \theta_1 x$
  $h_\theta(x)$ is represented as:

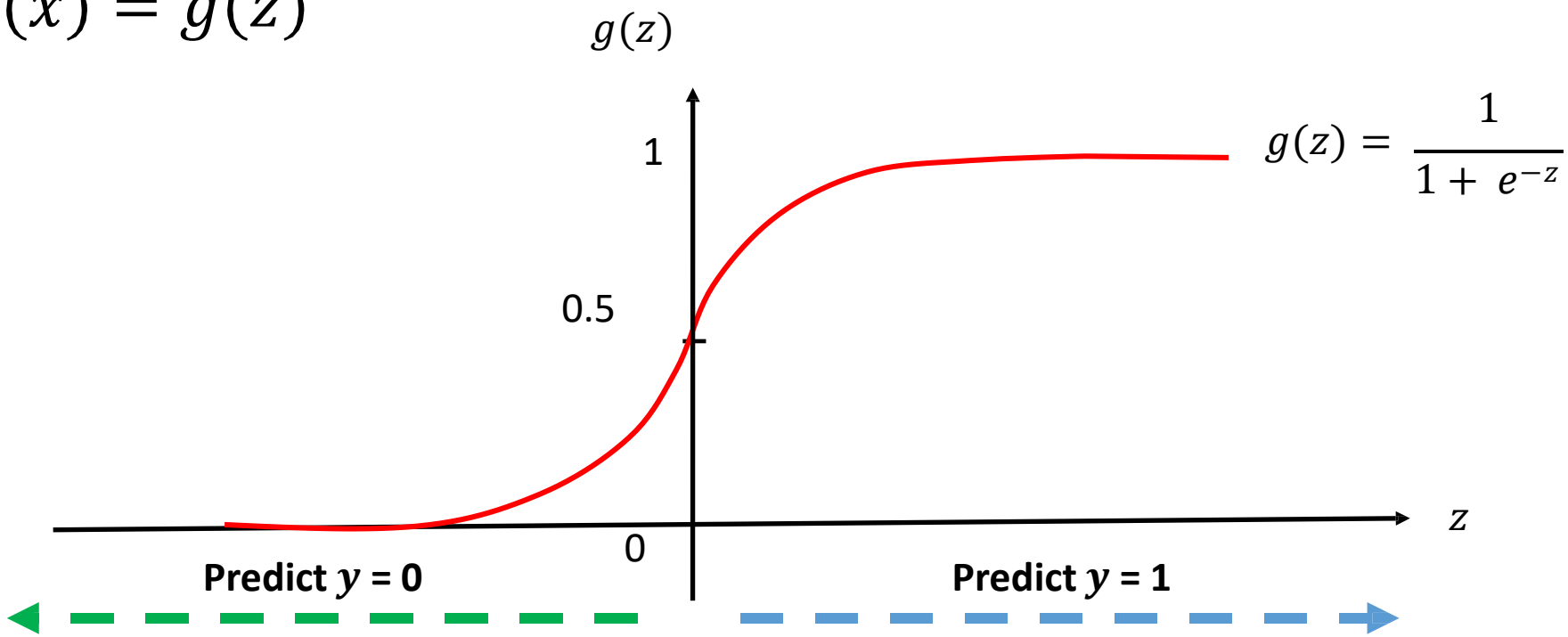
$h_\theta(x) = g(z)$ where g is a sigmoid function $g(z) = \frac{1}{1+e^{-z}}$

$$h\ (x) = \frac{1}{1+ e^{-(\theta_0 + \theta_1 x)}}$$



Logistic regression

$$z = \theta_0 + \theta_1 x$$
$$h_\theta(x) = g(z)$$



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

$z < 0$
$g(z) < 0.5$

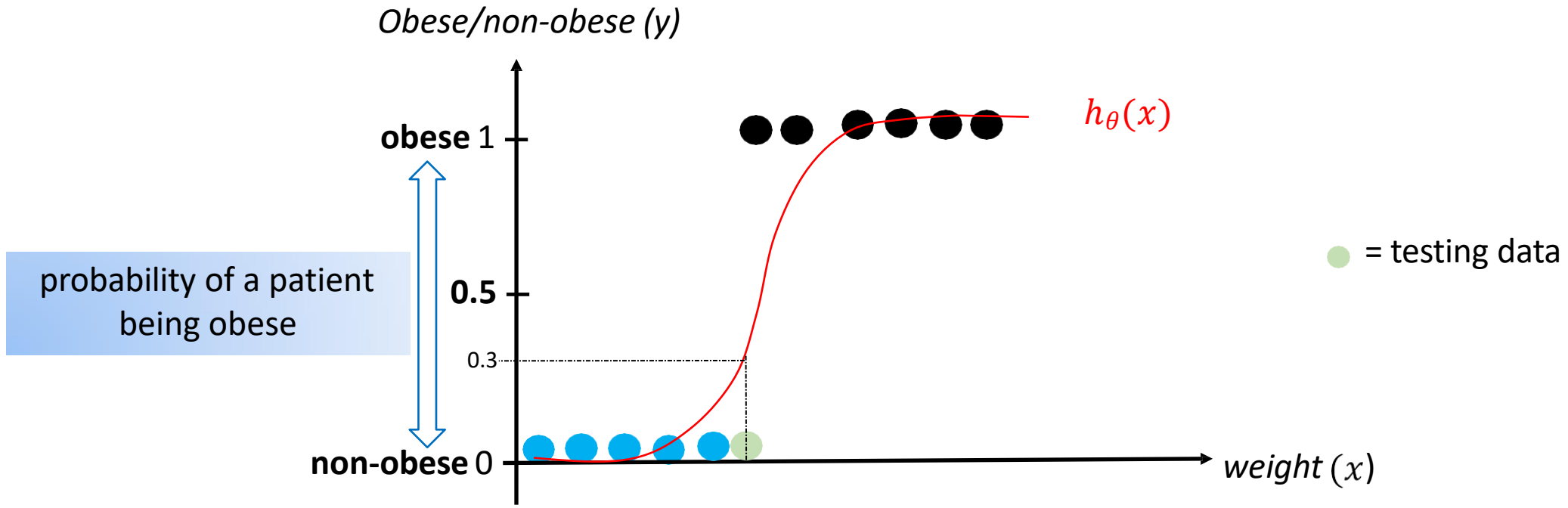
$\theta_0 + \theta_1 x < 0$
$h_\theta(x) < 0.5$

$z \geq 0$
$g(z) \geq 0.5$

$\theta_0 + \theta_1 x \geq 0$
$h_\theta(x) \geq 0.5$

Interpretation of $\boldsymbol{h_\theta(x)}$

- $h_\theta(x) = g(z) = g(\theta_0 + \theta_1 x) = g(\ \theta_0 + \theta_1 . weight\ )$



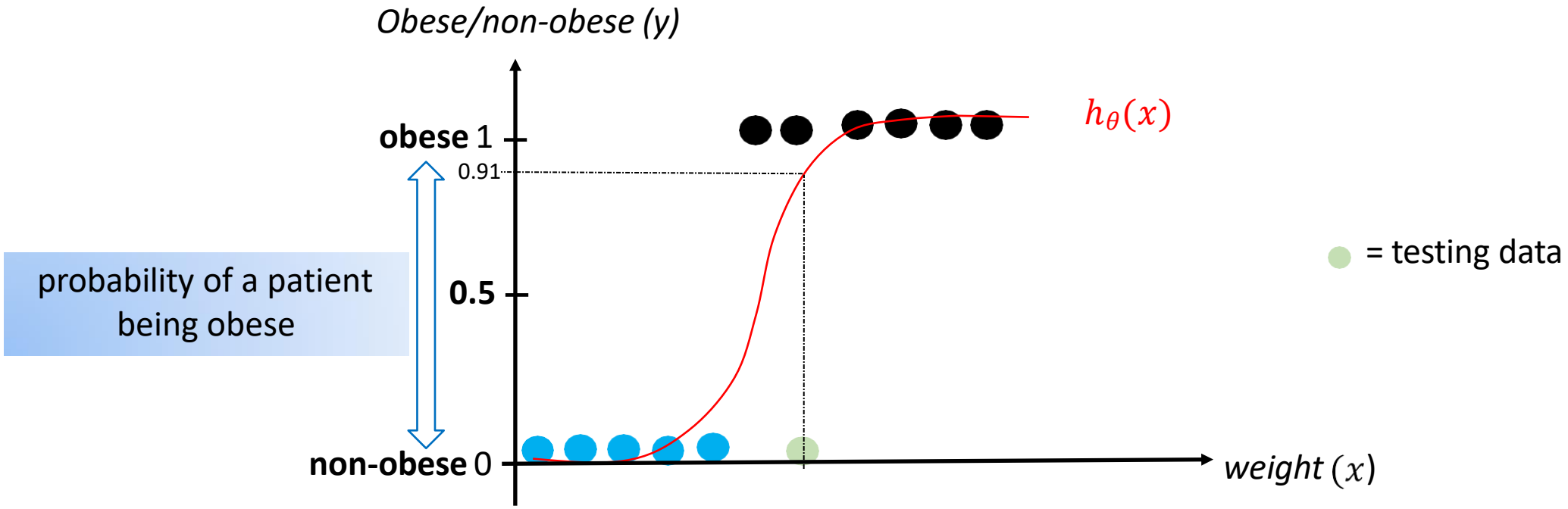
$h_\theta(x) = 0.3 \longrightarrow \mathrm{P}(y = 1\ |x; \theta) = 0.3$

Probability of 0.3 the patient is obese, given x and parameterised by θ

Predict y = 0 (non-obese)

Interpretation of $\boldsymbol{h_\theta(x)}$

- $h_\theta(x) = g(z) = g(\theta_0 + \theta_1 x) = g(\ \theta_0 + \theta_1 . weight\ )$



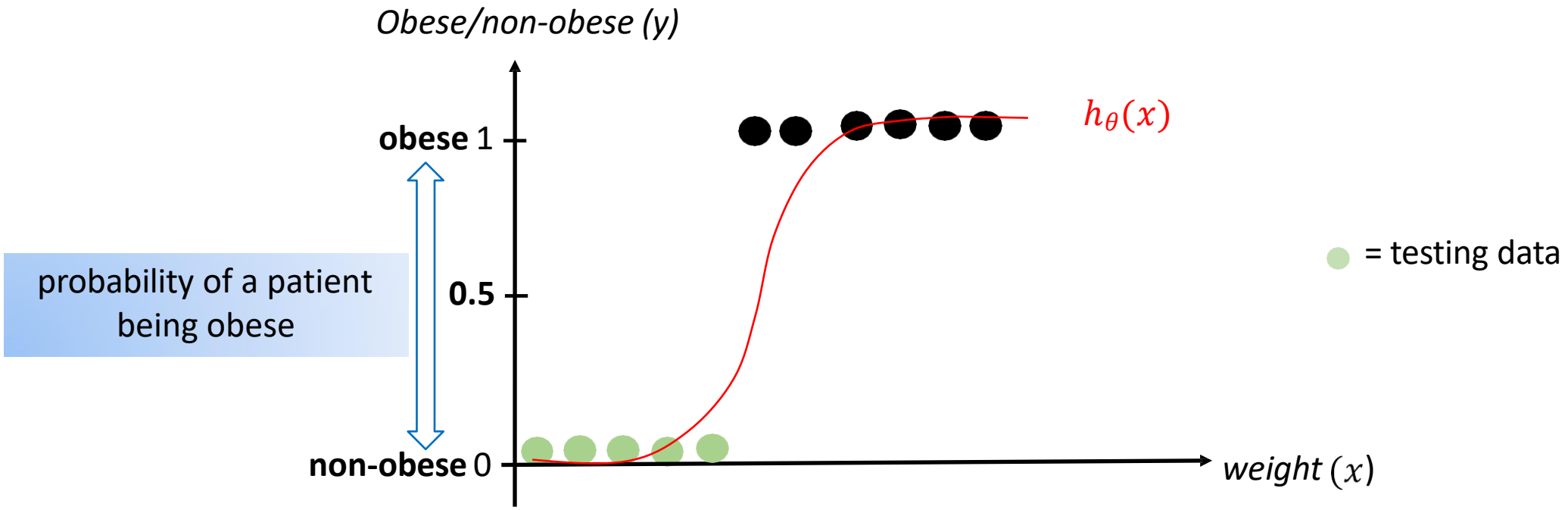
$h_\theta(x)$ = 0.91 ⟶ $\mathrm{P}(y = 1\ |x; \theta) = 0.91$

Probability of 0.91 the patient is obese, given x and parameterised by θ

Predict y = 1 (obese)

Interpretation of $\boldsymbol{h_\theta(x)}$

- $h_\theta(x) = g(z) = g(\theta_0 + \theta_1 x) = g(\ \theta_0 + \theta_1 . weight\ )$



$$h_\theta(x) = \mathrm{P}(y = 1 \mid x; \theta) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

$$\mathrm{P}(y = 0 \mid x; \theta) + \mathrm{P}(y = 1 \mid x; \theta) = 1$$

$$\mathrm{P}(y = 0 \mid x; \theta) = 1 - \mathrm{P}(y = 1 \mid x; \theta)$$

What is decision boundary?

- The **decision boundaries** separates the data-points into **decision regions**, which are actually the classes in which they belong.
- After training a machine learning model using a dataset, it is often necessary to visualize the classification of the data-points in Feature Space.

https://www.cs.princeton.edu/courses/archive/fall08/cos436/Duda/PR_simp/bndrys.htm

Decision boundary

- For example, our task is to classify two different species of *Iris* [1] (*setosa* and *versicolor)* from a series of N training data, based on the features (*sepal width, petal length*…) of the flower organ.

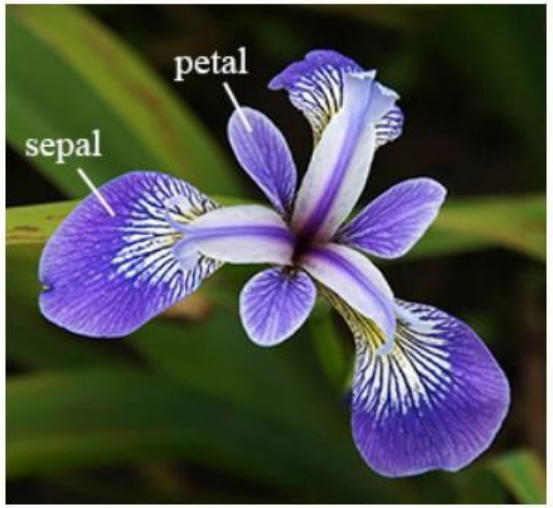
setosa

versicolor

[1] F. R. A., "The use of multiple measurements in taxonomic problems," Annual Eugenics, vol. 7, no. II, pp. 179-188, 1936

Decision boundary

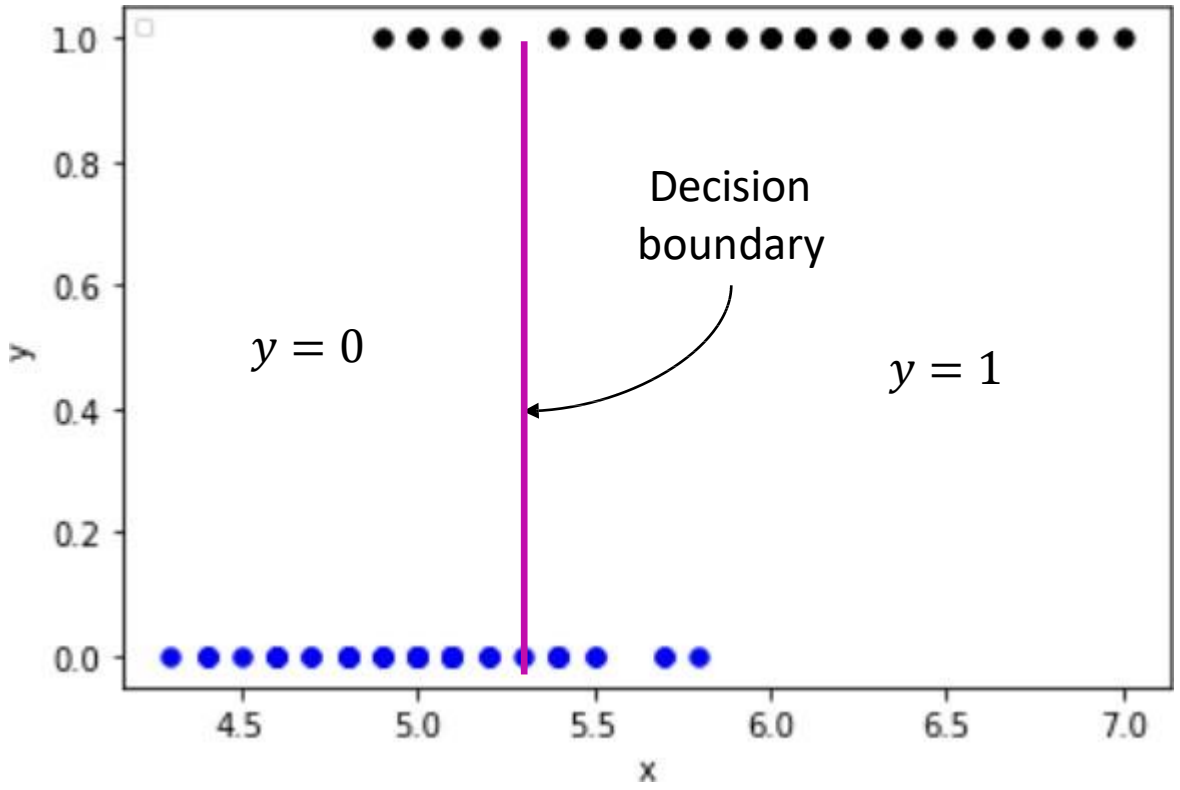
- To classify the two different species based on the *length* of the sepal*:*

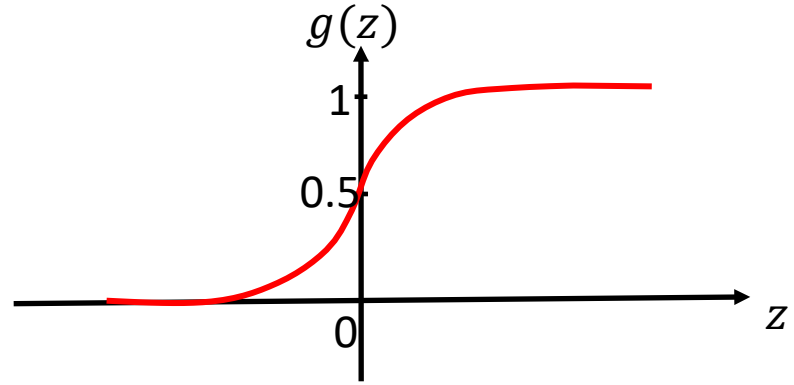


sepal (length), x	species	y
5.1	setosa	0
4.9	setosa	0
6.7	versicolor	1
5.4	setosa	0
6	versicolor	1
⋮	⋮	⋮
⋮	⋮	⋮

Decision boundary

$$z = \theta_0 + \theta_1 x$$
$$h_\theta(x) = g(z)$$





$$y = 1 \quad \text{if } h_\theta(x) \geq 0.5$$
$$z \geq 0$$

$$\theta_0 + \theta_1 x \geq 0$$
$$\theta_0 = -1.033 \quad \theta_1 = 0.192$$

$$-1.033 + 0.192x \geq 0$$
$$x \geq \frac{1.033}{0.192}$$

$$x \geq 5.380$$

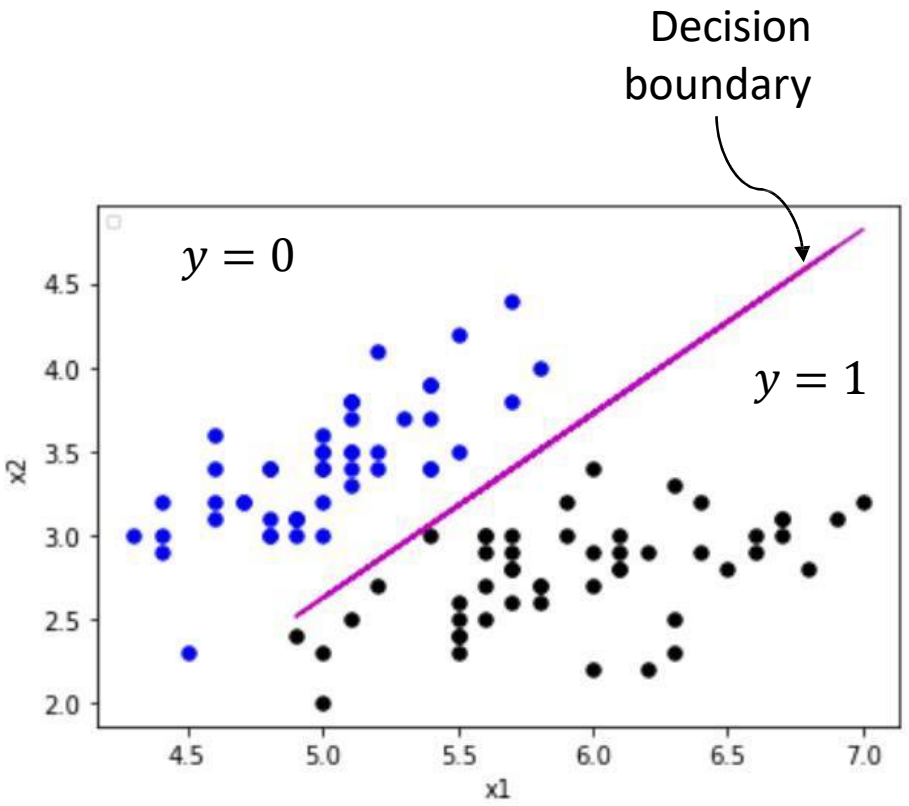
Decision boundary

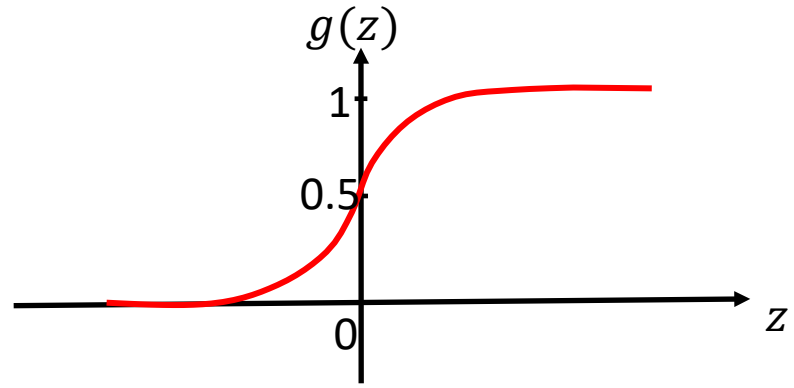
- To classify the two different species based on the *length* and *width* of the sepal*:*

sepal (length), x_1	sepal (width), x_2	species	y
5.1	3.5	setosa	0
4.9	3	setosa	0
6.7	3.1	versicolor	1
5.4	3.7	setosa	0
6	2.2	versicolor	1
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮

Decision boundary

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





$$y = 1 \quad \text{if } h_\theta(x) \geq 0.5$$
$$z \geq 0$$

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 \geq 0$$

$$\theta_0 = -0.588 \quad \theta_1 = 0.225 \quad \theta_2 = -0.205$$

$$-0.588 + 0.225 x_1 - 0.205 x_2 \geq 0$$

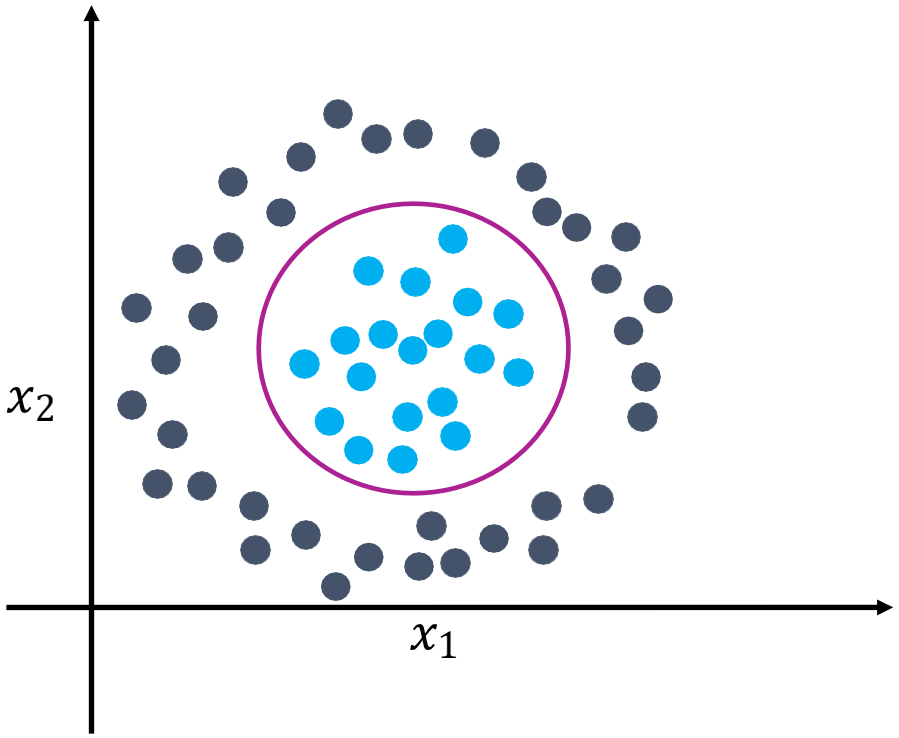
$$\frac{1}{0.205}[-0.588 + 0.225 x_1] \geq x_2$$

$$x_2 \leq \frac{0.225}{0.205} x_1 - \frac{0.588}{0.205}$$

Examples of non-linear decision boundary

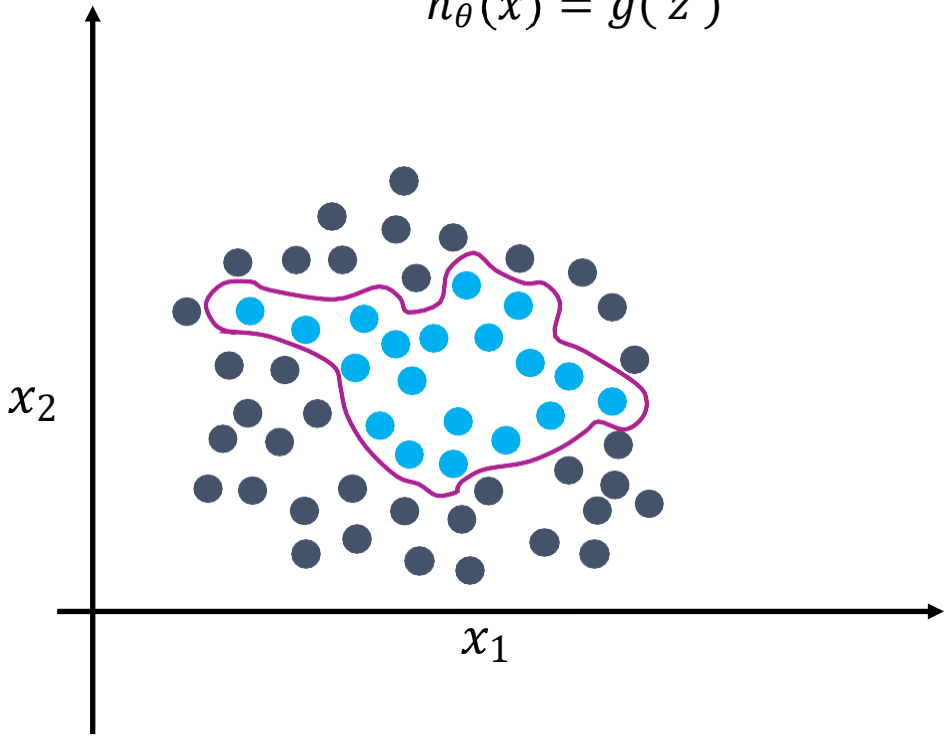
$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2$$

$$h_\theta(x) = g(z)$$



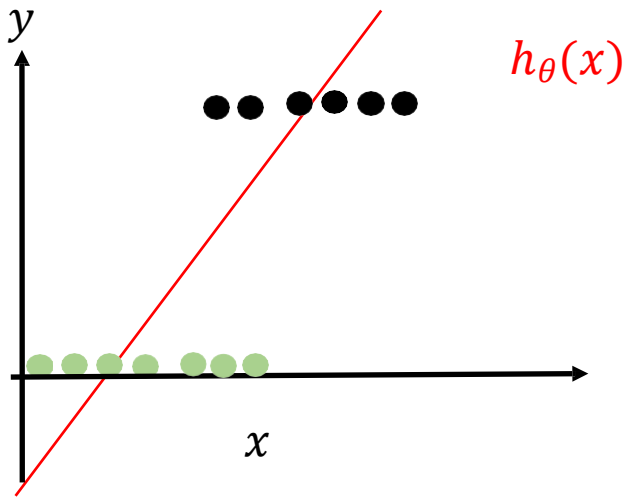
$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \ldots$$

$$h_\theta(x) = g(z)$$



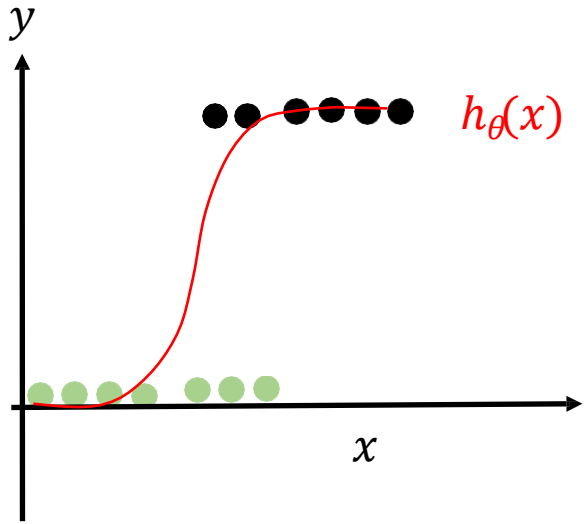
Linear regression vs Logistic regression

Linear regression



- Data is modelled by a **linear** function
- It is **necessary** that a linear relationship be established among dependent and independent variable
- The dependent target output, y is **continuous**

Logistic regression

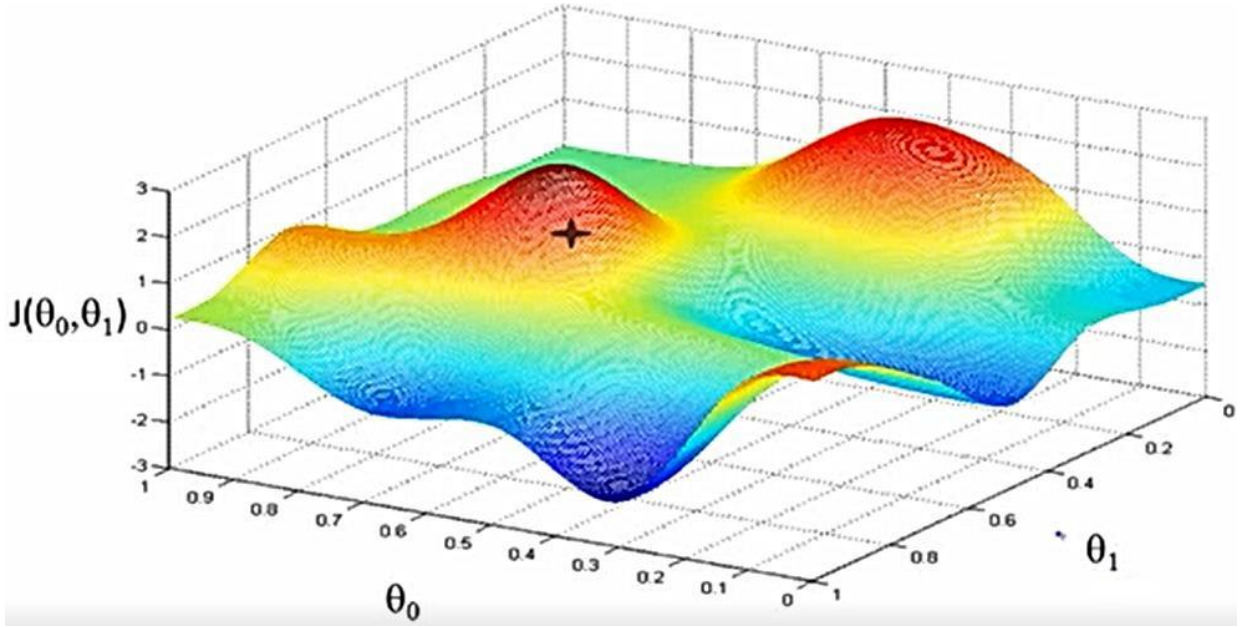


- Data is modelled by **S-shaped** logistic function
- It is **not necessary** that a linear relationship be established among dependent and independent variable
- The dependent target output, y is **discrete**

Cost function

$$J(\theta) = \frac{1}{2N} \sigma_{n=1}^{N} \; cost\left(h\ (x^{(n)}), y^{(n)}\right)$$

- Using linear regression cost function $J(\theta) = \frac{1}{2N} \sigma_{n=1}^{N} \left(h_\theta(x^{(n)}) - y^{(n)}\right)^2$ for logistic regression may result in a **non-convex** function.
- This is due to the sigmoid function of $h_\theta(x^{(n)})$ is a **non-linear** function.



[1] Machine Learning, accessed 12 May 2020, <https://www.coursera.org/learn/machine-learning>

Cost function

- In logistic regression, the cost function measures the discrepancy between the predicted probabilities and the actual labels in the training data. Specifically, it quantifies how well the model's predictions align with the true outcomes.
- The binary cross-entropy loss function is commonly used as the cost function. It penalizes the model more severely for misclassifications that have higher confidence.
- By minimizing the cost function, the model adjusts its parameters to improve its predictive accuracy on the training data.

Cost function

- The formula for the binary cross-entropy cost function is as follows:

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

Where:

- $J(\vartheta)$ represents the cost function.
- N is the number of training examples.
- $y^{(i)}$ is the actual label (0 or 1) of the *i*th training example.
- $h_\vartheta(x^{(i)})$ is the predicted probability that the *i*-th training example belongs to class 1, given by the sigmoid function applied to the input $x^{(i)}$ and the model parameters ϑ.

Gradient Descent

- Hypothesis $h_\theta(x)$ is represented as:

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

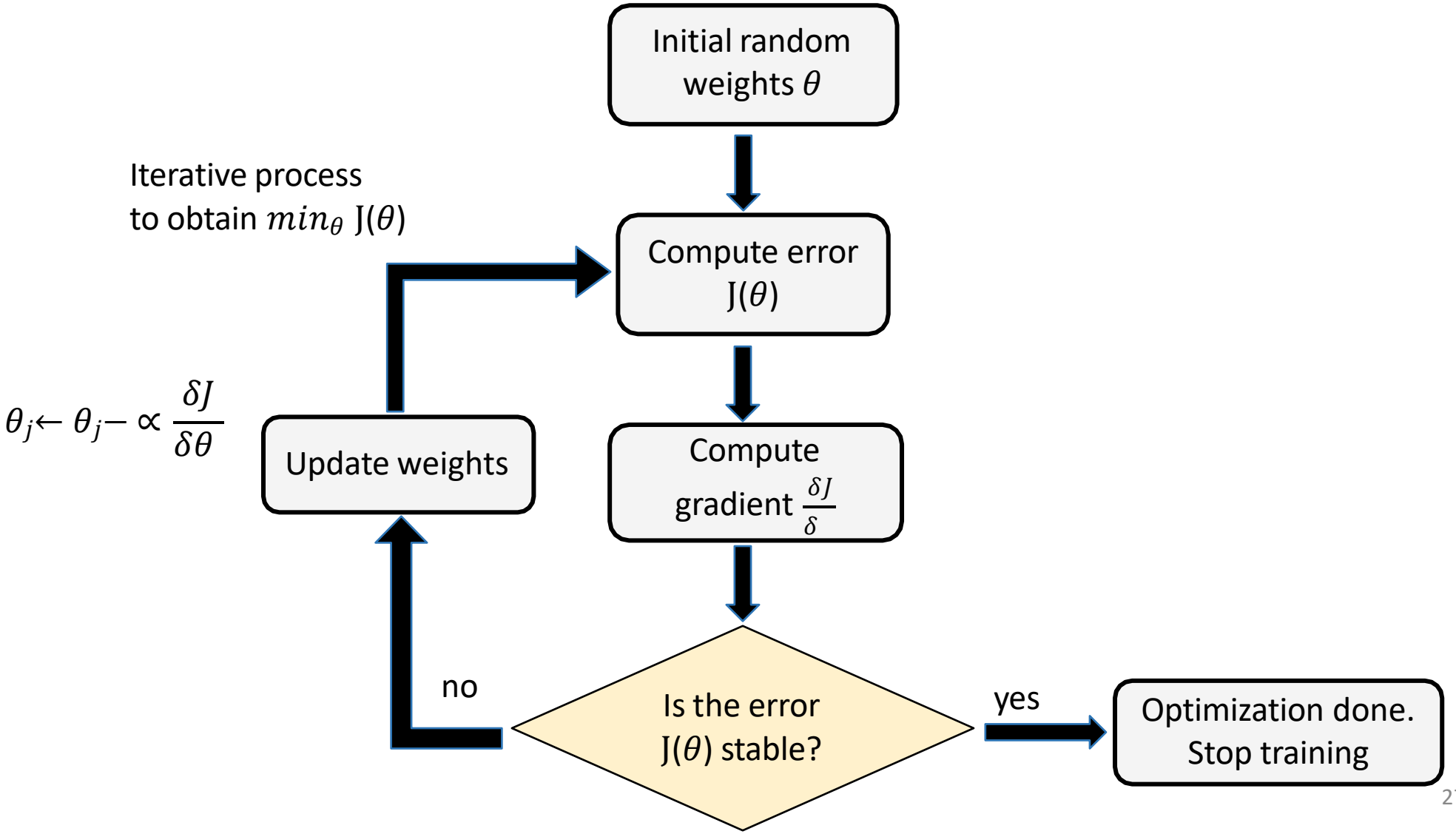
- Parameters:θ_0 ,…, θ_M
- Cost function:

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

- Goal:

$$minimize_\theta \ \mathrm{J}(\theta)$$

How Gradient Descent works (revisited)



Multiclass classification

setosa

versicolor

verginica

Multiclass classification

- Multiclass classification with logistic regression can be done through:
  - one-vs-all binary logistic regression
  - multinomial logistic regression.

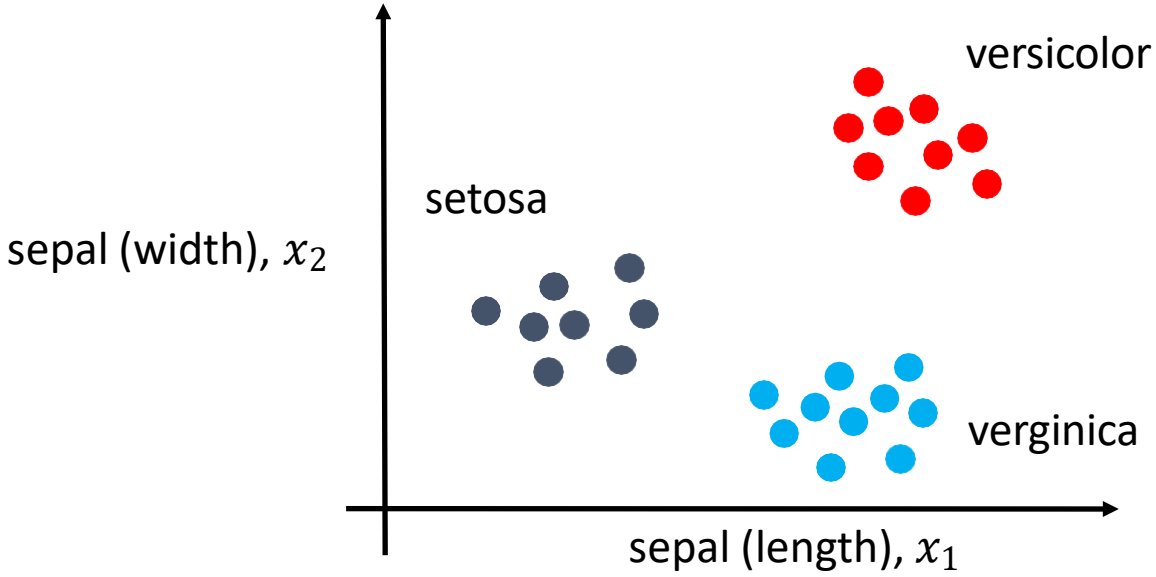
One-vs-all binary logistic regression

setosa

versicolor

verginica

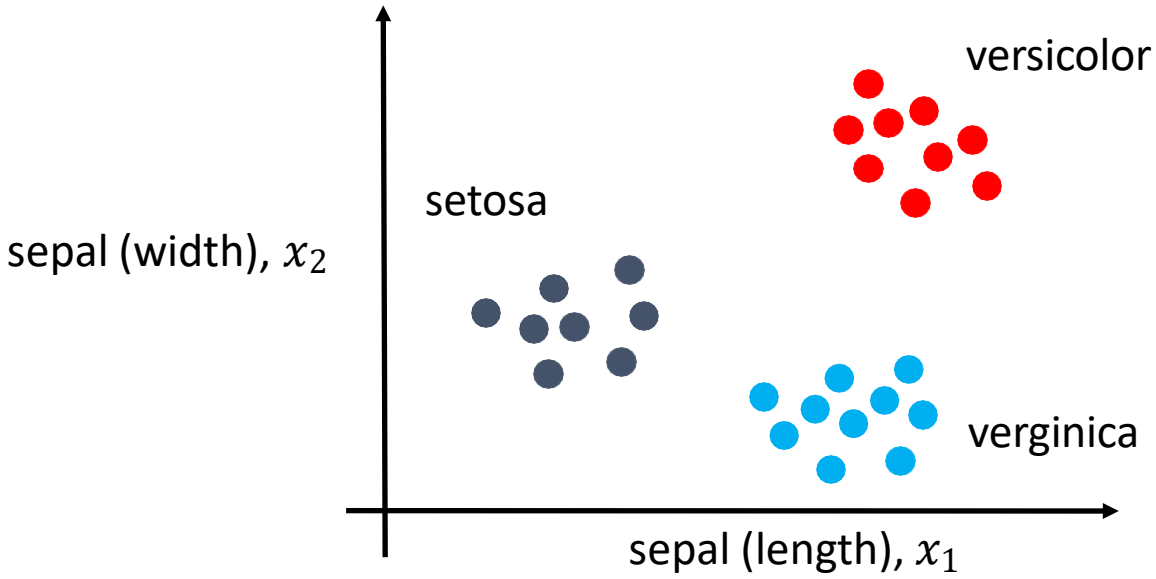
$C = [\text{setosa} \quad \text{versicolor} \quad \text{verginica}]$



One-vs-all binary logistic regression

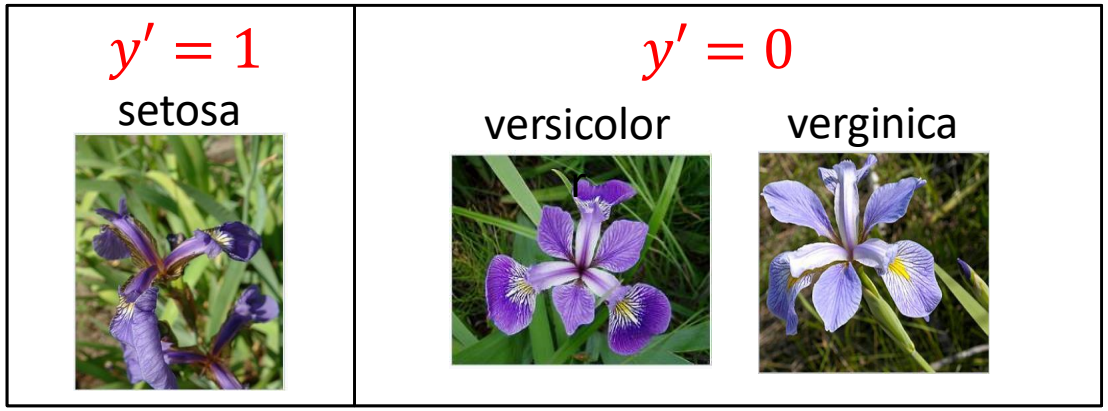
- Train a logistic regression classifier $h^{(c)}(x)$ for each class c to predict $\mathrm{P}(y = c|x;\theta)$:

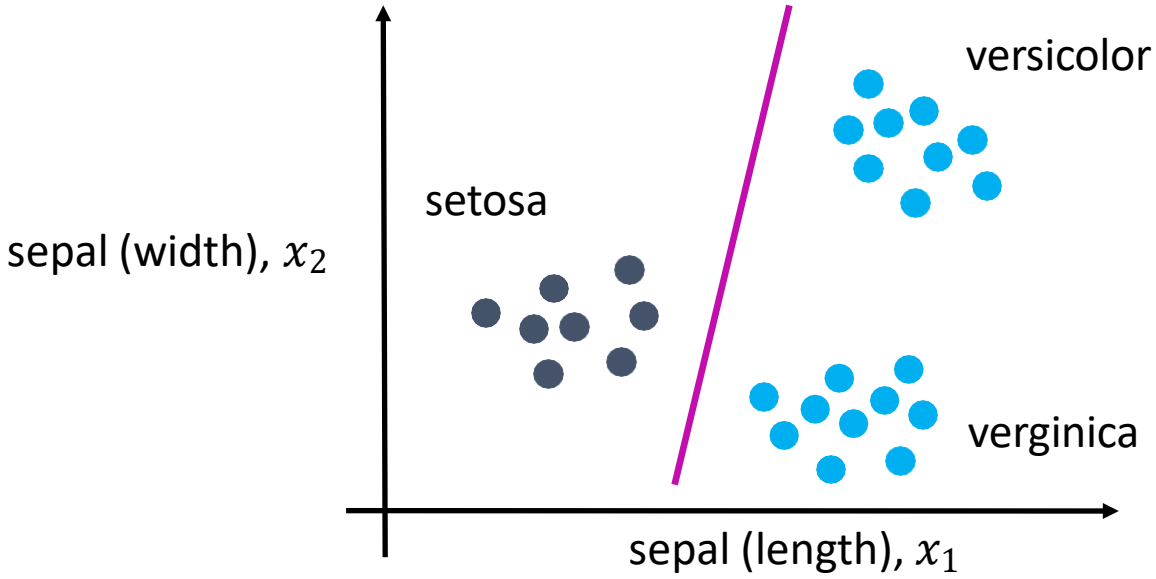
$$h_\theta^{(c)}(x) = \mathrm{P}(y = c|x;\theta) \quad c \in C$$



One-vs-all binary logistic regression

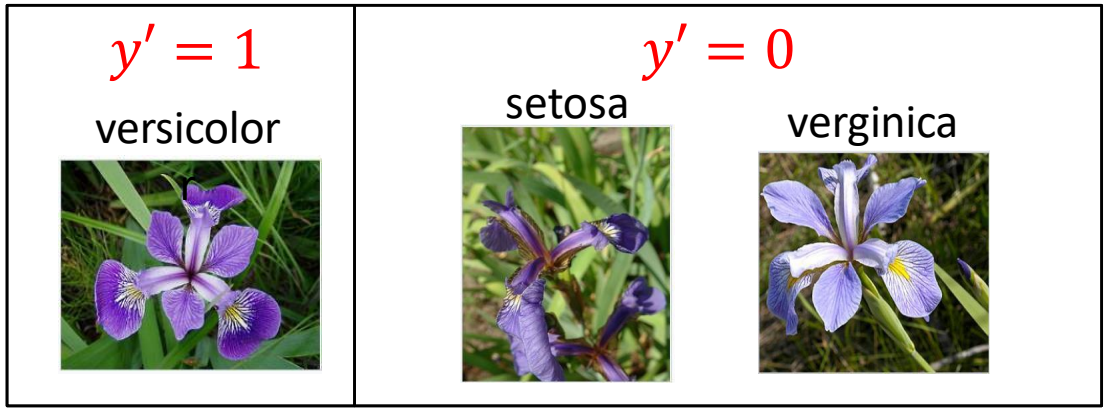
$$h_\theta^{(set.)}(x) = \mathrm{P}(y = \text{setosa} \mid x; \theta)$$

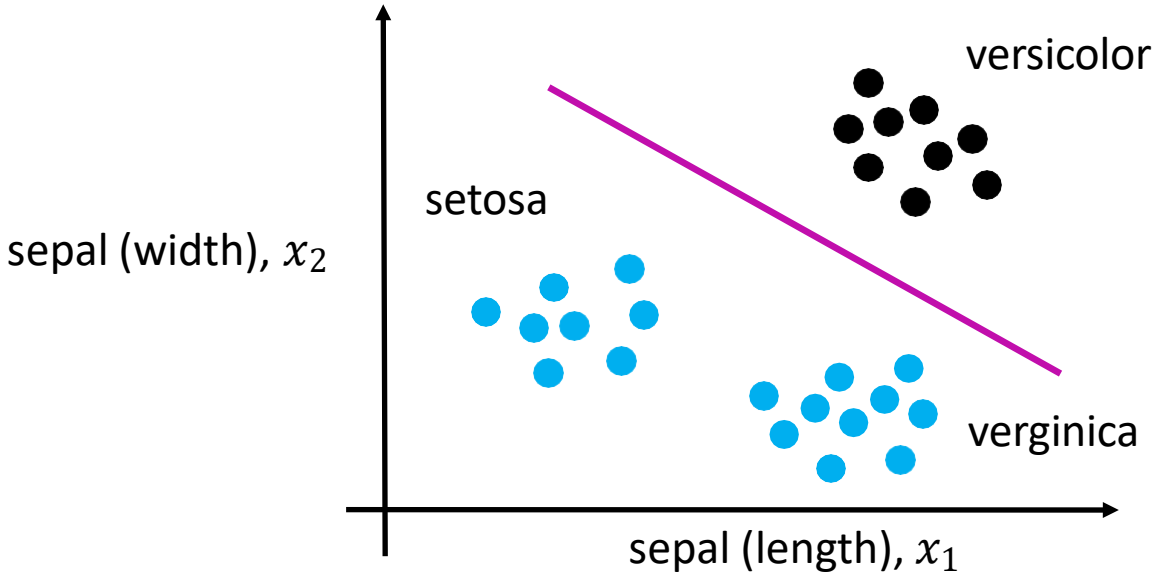




One-vs-all binary logistic regression

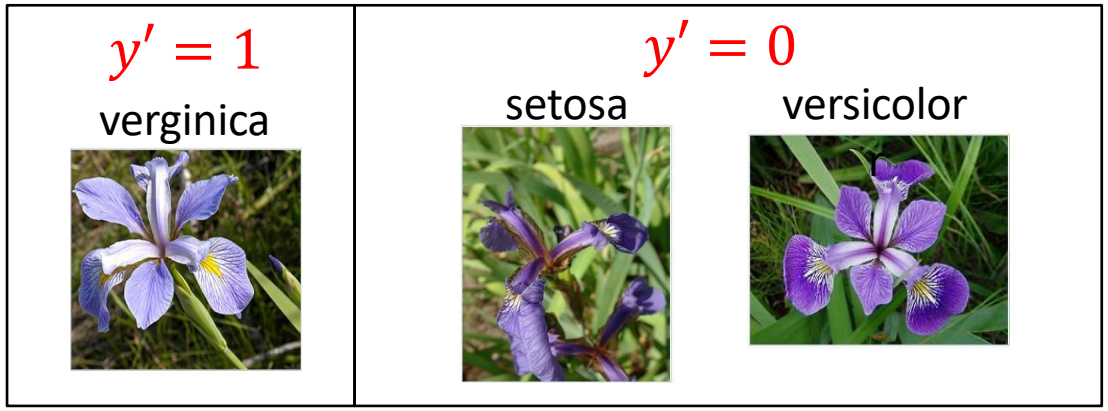
$$h_\theta^{(vers.)}(x) = \mathrm{P}(y = \text{versiclor} \mid x; \theta)$$

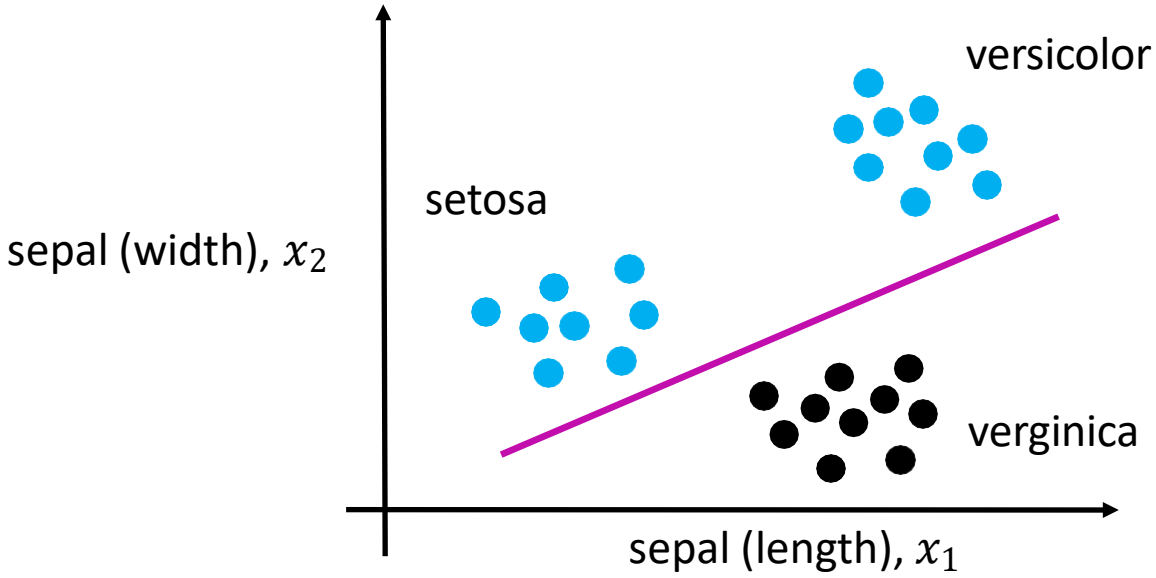




One-vs-all binary logistic regression

$h_\theta^{(verg.)}(x) = \mathrm{P}(y = \text{verginica} | x; \theta)$



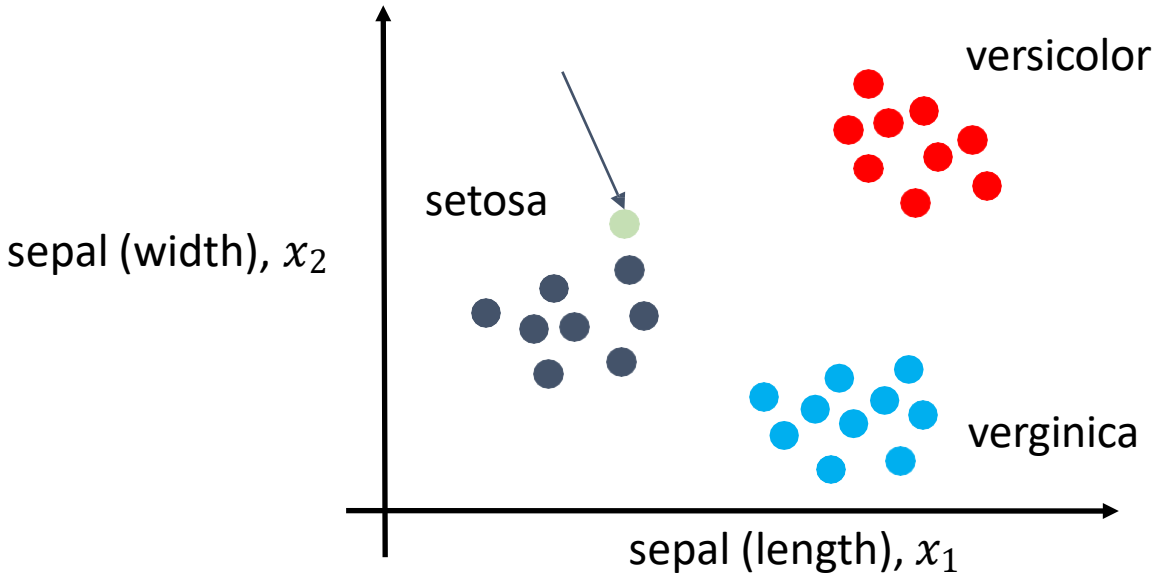


One-vs-all binary logistic regression

testing data, x

$$h_\theta^{(set.)}(x) = \mathrm{P}(y = \text{setosa} \mid x; \theta)$$
$$h_\theta^{(vers.)}(x) = \mathrm{P}(y = \text{versicolor} \mid x; \theta)$$
$$h_\theta^{(verg.)}(x) = \mathrm{P}(y = \text{verginica} \mid x; \theta)$$

$$\Rightarrow \max_c h_\theta^{(c)}(x)$$



Multiclass classification

- Multiclass classification with logistic regression can be done through:
  - one-vs-all binary logistic regression
  - multinomial logistic regression.

Multinomial logistic regression

- Multiple logistic regression, also known as multinomial logistic regression, is an extension of binary logistic regression that allows for more than two categories in the dependent variable. It's commonly used when the dependent variable has three or more categories that are unordered.

- In multiple logistic regression, the model estimates the probabilities of each category of the dependent variable and assigns observations to the category with the highest predicted probability.

Multinomial logistic regression

- To classify the three different species based on the *length* and *width* of the sepal:

$$C = [\text{setosa} \quad \text{versicolor} \quad \text{verginica}]$$

$$y^{(n)} \in \{1,2,3\}$$

sepal (length), x_1	sepal (width), x_2	species	y
5.1	3.5	setosa	1
4.9	3	setosa	1
6.7	3.1	versicolor	2
6.7	3.3	virginica	3
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮

Multinomial logistic regression

$$z = [z_{set.}, z_{vers.}, z_{verg.}] = [z_1, z_2, z_3]$$

where $z_c = \theta_0^{(c)} + \theta_1^{(c)} x_1 + \theta_2^{(c)} x_2 = \theta^{(c)T} x$

$$h_\theta(x) = \text{softmax}(z)$$

The softmax function is used to convert raw scores into probabilities while ensuring that the probabilities sum up to 1 across all categories. It's defined as:

$$\text{softmax}(z_c) = \frac{e^{z_c}}{\sum_{k=1}^{K} e^{z_k}}$$

Where:

- **z** is a vector of raw scores. K is the total number of classes
- softmax($\mathbf{z_c}$) is the probability of category c given the raw scores.

Multinomial logistic regression

$$z = [z_{set.}, z_{vers.}, z_{verg.}] = [z_1, z_2, z_3]$$

where $z_c = \theta_0^{(c)} + \theta_1^{(c)} x_1 + \theta_2^{(c)} x_2 = \theta^{(c)T} x$

$$h_\theta(x) = \text{softmax}(z) = \begin{bmatrix} \frac{e^{z_1}}{\sigma_{i=1}^{3} e^{z_i}} \\ \frac{e^{z_2}}{\sigma_{i=1}^{3} e^{z_i}} \\ \frac{e^{z_3}}{\sigma_{i=1}^{3} e^{z_i}} \end{bmatrix} = \begin{bmatrix} \text{P}(y = \text{setosa}|x) \\ \text{P}(y = \text{versicolor}|x) \\ \text{P}(y = \text{verginica}|x) \end{bmatrix}$$

Multinomial logistic regression

$C = [\textbf{setosa} \quad \text{versicolor} \quad \text{verginica}]$

$$h_\theta(x) = \text{softmax}(z) = \begin{bmatrix} 0.8 \\ 0.15 \\ 0.05 \end{bmatrix}$$

Then $y = 1$

Cost function

- Multinomial logistic regression has a slightly different loss function than binary logistic regression because it uses the **softmax** rather than the sigmoid classifier.
- The cost function used in multiple logistic regression is typically the cross-entropy loss function. It measures the difference between the predicted probabilities and the actual labels. The cross-entropy loss function for multiple logistic regression is given by:

$$J(\boldsymbol{\Theta}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\log(\hat{p}_{ik})$$

Where:

- **Θ** represents all the parameter vectors.
- *N* is the number of observations.
- y_{ik} is an indicator variable that equals 1 if observation *i* belongs to category *k*, and 0 otherwise.
- $\hat{p}_{ik}$ is the predicted probability that observation *i* belongs to category *k*.

Gradient Descent variation

- Gradient descent update a set of parameters in an **iterative** manner to **minimize** an error function.
- The behavior of gradient descent varies depending on the quantity of training data utilized during each iteration.
- There are three main types of gradient descent: **batch**, **stochastic**, and **mini-batch**.

Batch Gradient Descent

- Batch gradient descent calculates the error for each example in the training dataset, but only updates the model after **all** training examples have been evaluated.
- One **training epoch** means one cycle through the **entire** training dataset.

Advantage

- Fewer updates to the model leading to computationally efficient.
- Produces a stable error gradient and a **stable convergence**.

Disadvantage

- A stable error gradient can sometimes result in **premature convergence** of the model to a less optimal set of parameters.
- **Resource exhausted** as it requires the entire training set resides in memory.
- Model updates, and therefore the speed of training, can become **very slow for large datasets**.

Mini-batch Gradient Descent

- Mini-batch gradient descent splits the training dataset into **mini batches,** and calculates the error and updates the model for each mini batch.

Advantage

- Update frequency that is higher than batch gradient descent allows for a more **robust convergence**, avoiding **local minima.**
- Mini-batch size < total training set = adding noise to the learning process, which can help to **improve the generalization** error.
- **Resource efficiency**

Disadvantage

- Additional **hyperparameter** which is the mini-batch size for the learning algorithm.

Stochastic Gradient Descent

- Stochastic gradient descent, often abbreviated SGD, calculates the error and updates the model for **each** example in the training dataset.

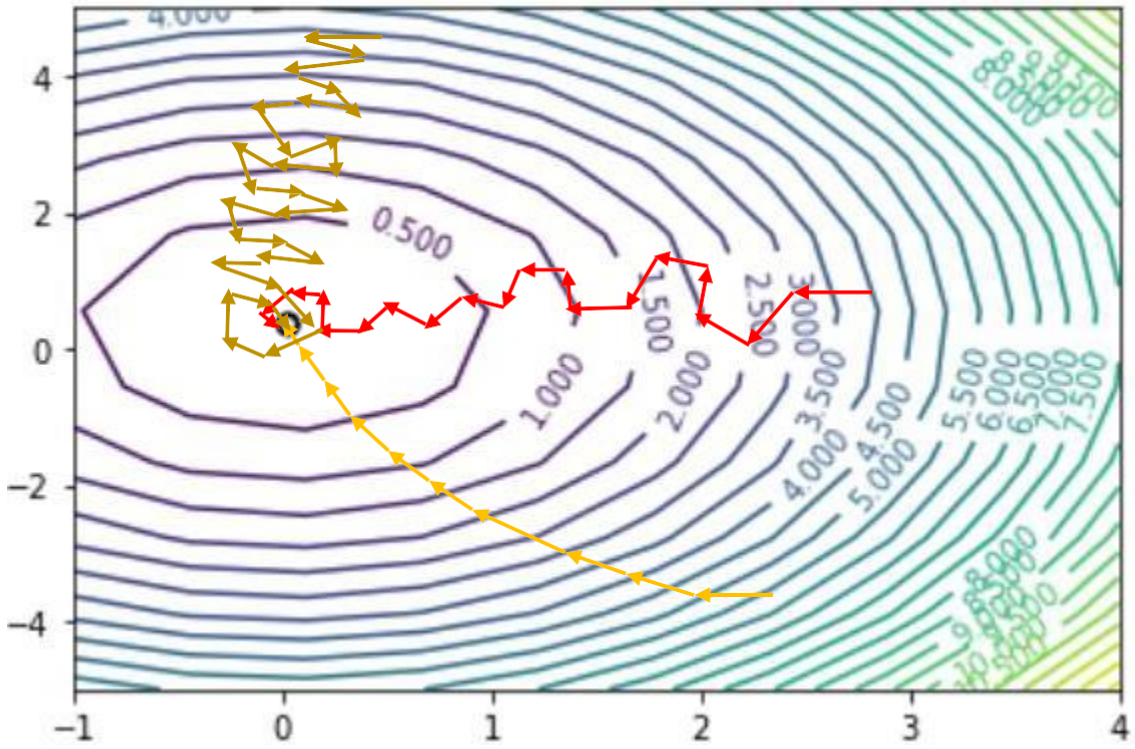
Advantage

- Adding more noise to the learning process than mini- batch helps **to improve generalization error.**
- Faster updates can result in faster learning on some problems.
- **Resource efficiency**

Disadvantage

- Frequent updates is more **computationally expensive** than the other gradient descent configurations.
- The noisy learning process causes the model **harder to converge**.

Gradient Descent variation



Batch Gradient Descent

Mini-batch Gradient Descent

Stochastic Gradient Descent

- Mini-batch Gradient Descent which uses a combination of Stochastic Gradient Descent and Batch Gradient Descent is often a preferred method to be used.

Logistic Regression in Python

Python libraries

- Numpy: library for efficient numerical operations.
- Pandas: library for data manipulation.
- Scikit-learn: library of machine learning tools.
  - linear_model: Implements logistic regression and other linear models.
  - vm: Module for support vector machine algorithms.
  - tree: Module for decision tree-based algorithms.

Import libraries

```
import numpy as np
import pandas as pd
from sklearn import linear_model
```

Logistic Regression in Python

Split training and testing data

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.2,
random_state=0)
```

test_size: specifies the proportion of the dataset that should be allocated for testing.
random_state: controls the randomness of the data splitting process. When random_state is set to an integer, it acts as a seed, ensuring that the data splitting process is reproducible.

Logistic Regression in Python

Fit (train) the Logistic Regression classifier

```
clf = linear_model.LogisticRegression(C=1e40, solver='newton-cg')
fitted_model = clf.fit(X_train, Y_train)
```

- **C**: The C parameter represents the inverse of regularization strength. Regularization is a technique used to prevent overfitting by penalizing large coefficients. A smaller value of C implies stronger regularization, where the model is penalized more for large coefficients. Default 1.0.
- **solver**: The solver parameter specifies the optimization algorithm to use in the logistic regression model.

Logistic Regression in Python

Evaluate model:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score
# Predict on the test data
y_pred = clf.predict(X_test)
# Calculate evaluation scores
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
# Print the evaluation scores
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
....
```

Logistic Regression in Python

Make predictions:

```
data = pd.DataFrame([[1, 2, 48],[2, 0, 45]],
      columns=['feature1', 'feature2', 'feature3’])

# Make predictions on new data
predictions = clf.predict(data)

# Print the predicted class labels
print(predictions)
```

The prediction is an array containing all the predicted class labels for the new data.

Next Lecture

- Support vector machine (SVM)
- Artificial neural networks (ANN)