# 7. Supervised Learning: Logistic Regression

# Logistic Regression

## Example

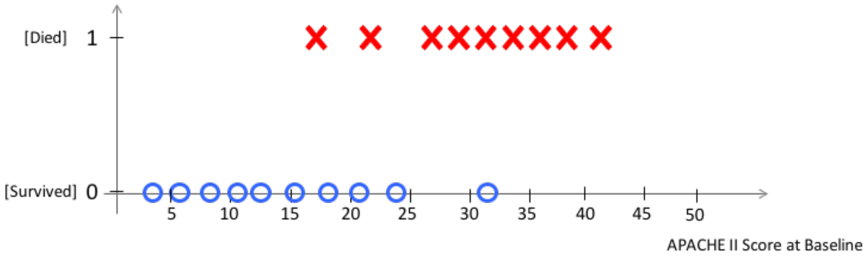
- Prediction of sepsis mortality based on features such as APACHE II score at baseline
- It is a supervised learning problem because one has access to input-output examples
- It is a classification problem because the output variable is discrete-valued
- Input variables also known as features
- Output variable also known as labels

## Training set

| *Input Variable (APACHE II Score)* | *Output Variable (Survived / Died)* |
|---|---|
| *3* | *Survived* |
| *6* | *Survived* |
| *15* | *Survived* |
| *29* | *Died* |
| *38* | *Died* |

*Training sample*

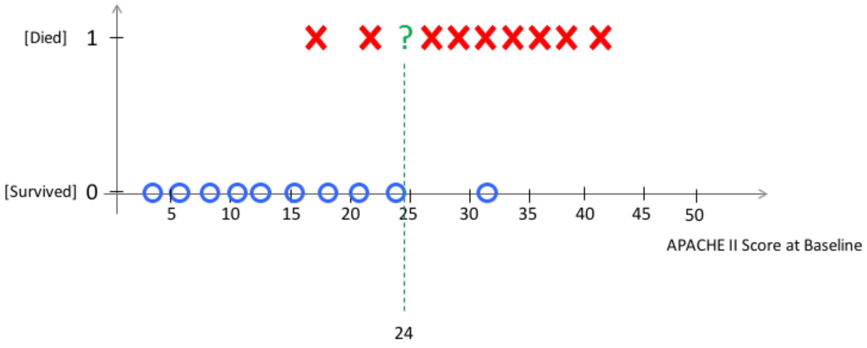
$(x_i, y_i)$




## Testing set

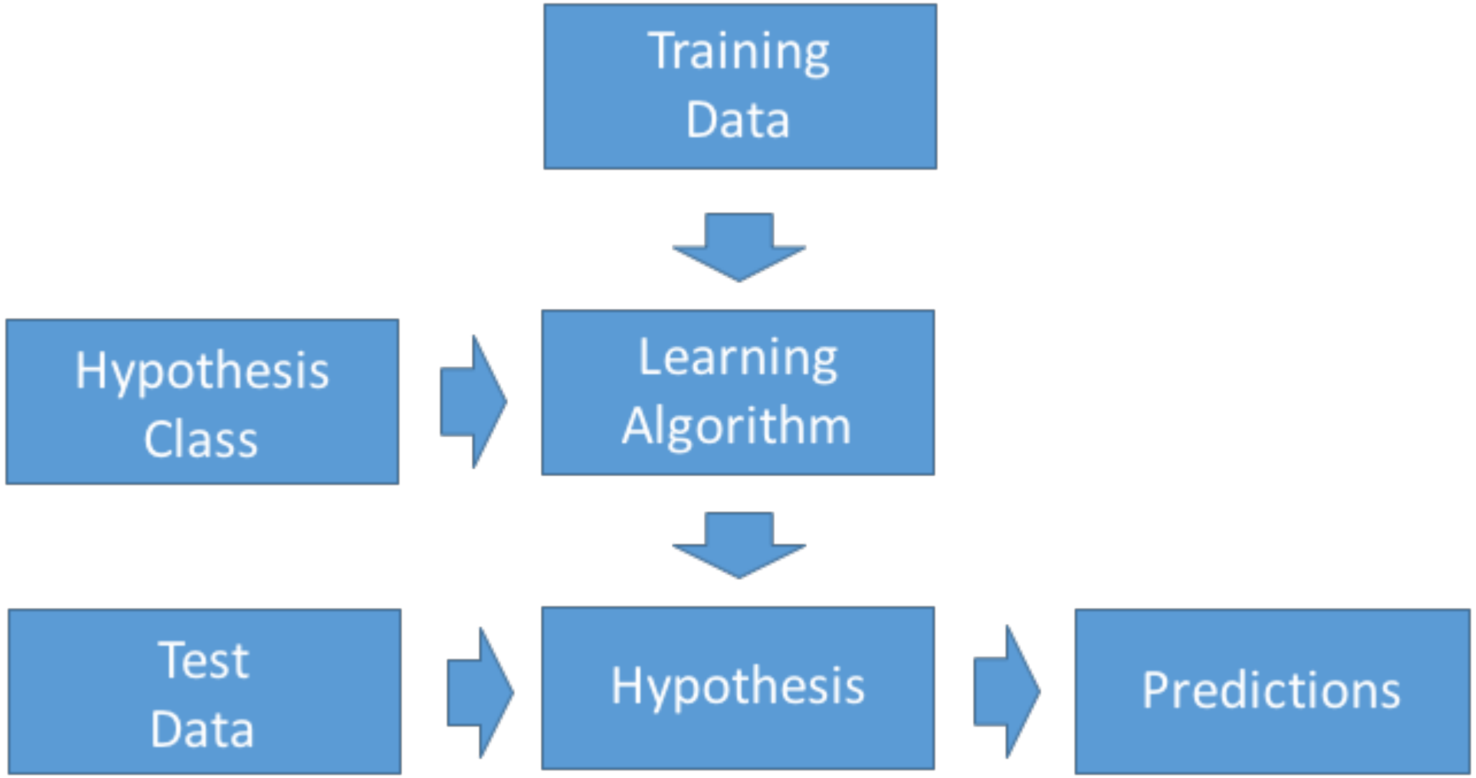
| *Input Variable (APACHE II Score)* | *Output Variable (Survived / Died)* |
|---|---|
| *24* | *?* |

*Testing Sample*



# Logistic Regression

## Process

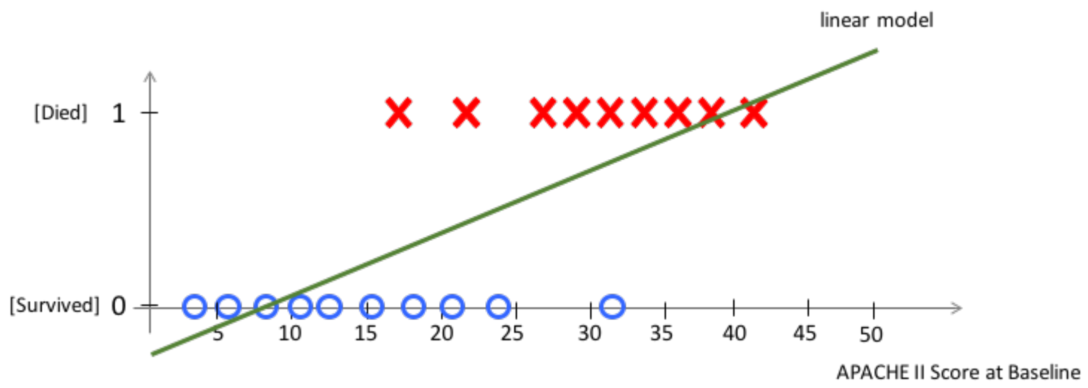



- One is given access to a **training data** – consisting of various **feature-label** pairs – and **testing data** – consisting of features points with unknown **label**.
- One is also given a **hypothesis** (or **model**) **class** containing a series of **hypotheses** (or **models**) that potentially explain the relationship between the **features** and **labels**.
- The **learning algorithm** selects a **hypothesis** (or **model**) from the **hypothesis** (or **model**) **class** that fits the **training data**.
- Such selected **hypothesis** can then be used on the **testing data** to determine the **label** associated with the new **features**.

# Logistic Regression vs. Simple Linear Regression

## Can we solve this classification problem using simple linear regression?

Example 1



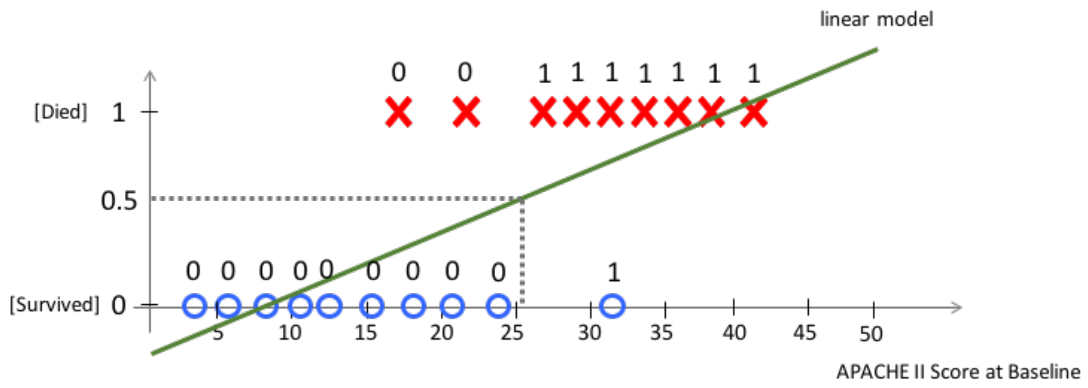

Classification Rule (Classification with Simple Linear Regression)

- If output of linear regressor is less than 0.5 declare SURVIVED
- If output of linear regressor is more than 0.5 declare DIED

# Logistic Regression vs. Simple Linear Regression

## Can we solve this classification problem using simple linear regression?

Example 1




Classification Rule (Classification with Simple Linear Regression)

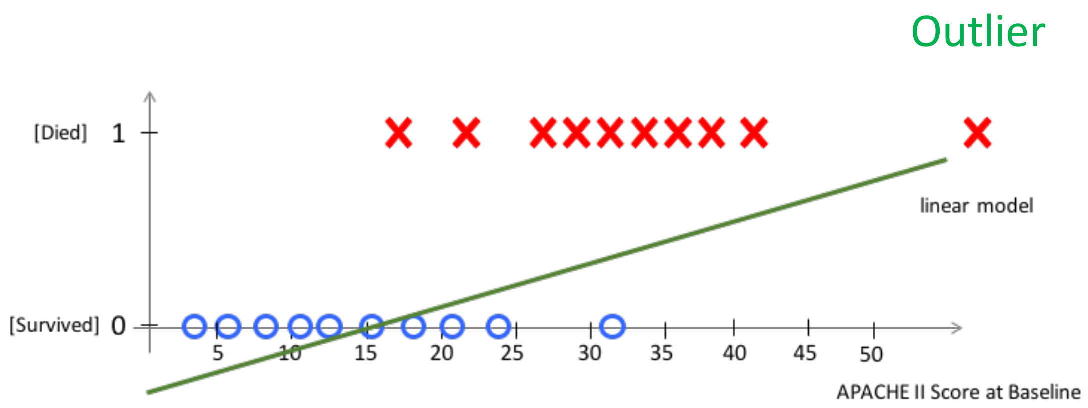
- If output of linear regressor is less than 0.5 declare SURVIVED
- If output of linear regressor is more than 0.5 declare DIED

Classification performance is relatively good!

# Logistic Regression vs. Simple Linear Regression

## Can we solve this classification problem using simple linear regression?

Example 2



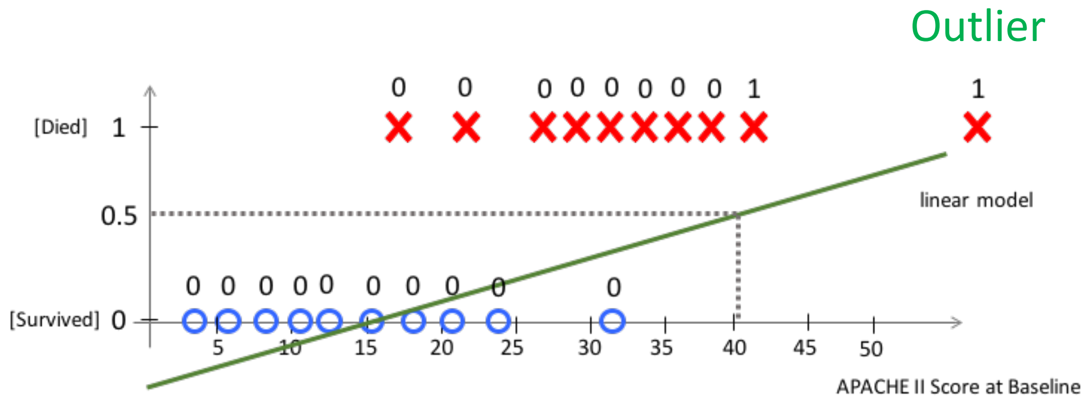

Classification Rule (Classification with Simple Linear Regression)

- If output of linear regressor is less than 0.5 declare SURVIVED
- If output of linear regressor is more than 0.5 declare DIED

# Logistic Regression vs. Simple Linear Regression

## Can we solve this classification problem using simple linear regression?

Example 2




Classification Rule (Classification with Simple Linear Regression)

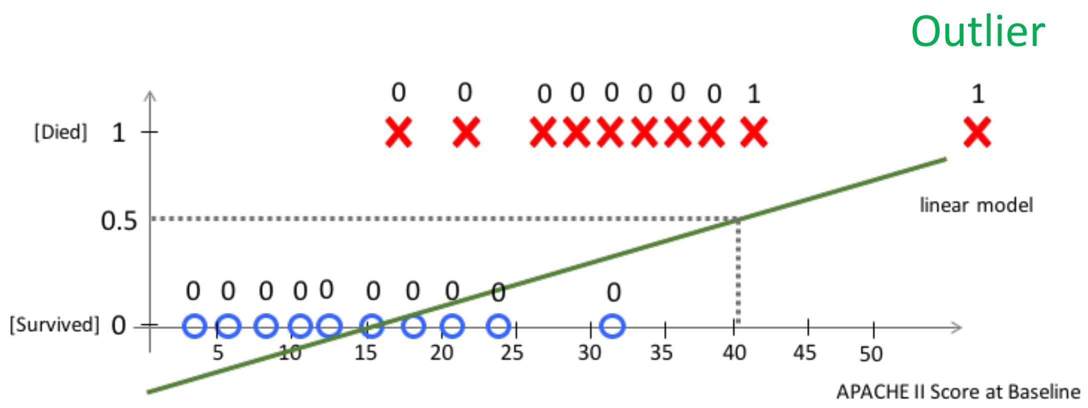
- If output of linear regressor is less than 0.5 declare SURVIVED
- If output of linear regressor is more than 0.5 declare DIED

Classification performance is very poor!

# Logistic Regression vs. Simple Linear Regression

## Can we solve this classification problem using simple linear regression?

Example 2




Classification Rule (Classification with Simple Linear Regression)

- If output of linear regressor is less than 0.5 declare SURVIVED
- If output of linear regressor is more than 0.5 declare DIED

**Classification**

The output should be 0 or 1

**Linear Regression**

The output can be less than 0
and can be greater than 1 .

**Logistic Regression**

The output is between 0 and 1

# Logistic Regression: Model

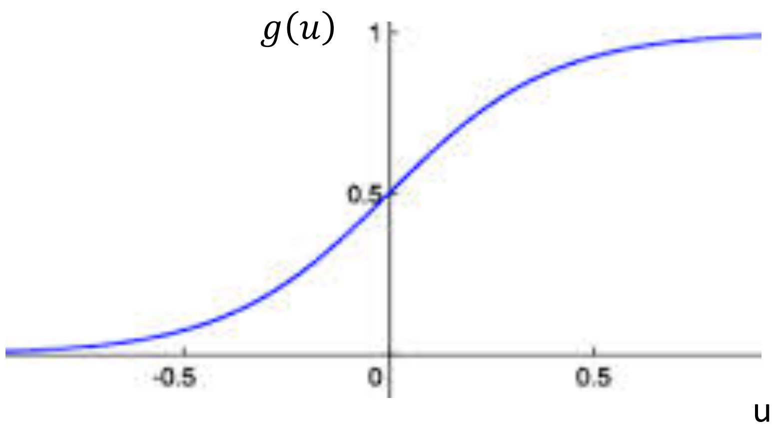
## Model

The hypothesis is such that:

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

where $\boldsymbol{\theta} = (\theta_0, \theta_1, \dots, \theta_m)^t$ is the param. vector
and $\boldsymbol{x} = (1, x_1, \dots, x_m)^t$ is the feature vector

## Logistic or Sigmoid Function




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

## Interpretation

$$\Pr(\hat{y} = 1 | \boldsymbol{x}; \boldsymbol{\theta}) = h_{\boldsymbol{\theta}}(\boldsymbol{x}) \implies \Pr(\hat{y} = 0 | \boldsymbol{x}; \boldsymbol{\theta}) = 1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})$$

$$\Pr(\hat{y} = 1 | \boldsymbol{x}; \boldsymbol{\theta}) = 0.7 \implies \text{A patient has a 70\% chance of a tumor being malignant}$$

# Logistic Regression: Decision Regions

## Classifier

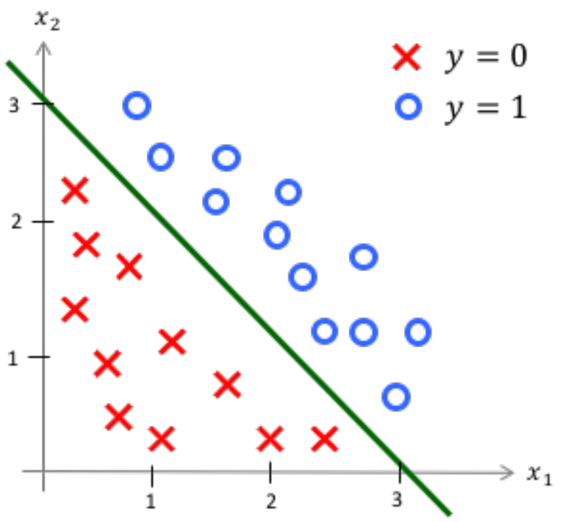
$$\hat{y} = 1 \quad \Leftarrow \quad \Pr(\hat{y} = 1 | \boldsymbol{x}; \boldsymbol{\theta}) \geq 0.5$$

$$\hat{y} = 0 \quad \Leftarrow \quad \Pr(\hat{y} = 1 | \boldsymbol{x}; \boldsymbol{\theta}) < 0.5$$

$$\hat{y} = 1 \quad \Leftarrow \quad \boldsymbol{\theta}^t \boldsymbol{x} \geq 0$$

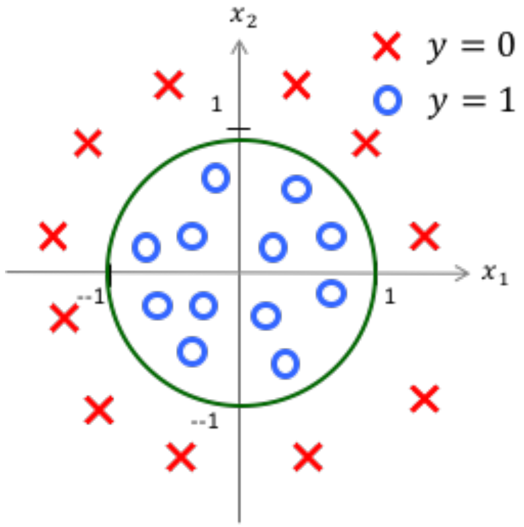
$$\hat{y} = 0 \quad \Leftarrow \quad \boldsymbol{\theta}^t \boldsymbol{x} < 0$$

## Linear Classifiers




The hypothesis is $h_{\boldsymbol{\theta}}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$ with $\theta = (-3,1,1)^t$.

## Nonlinear Classifiers




The hypothesis is $h_{\boldsymbol{\theta}}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$ with $\theta = (-1,0,0,1,1)^t$.

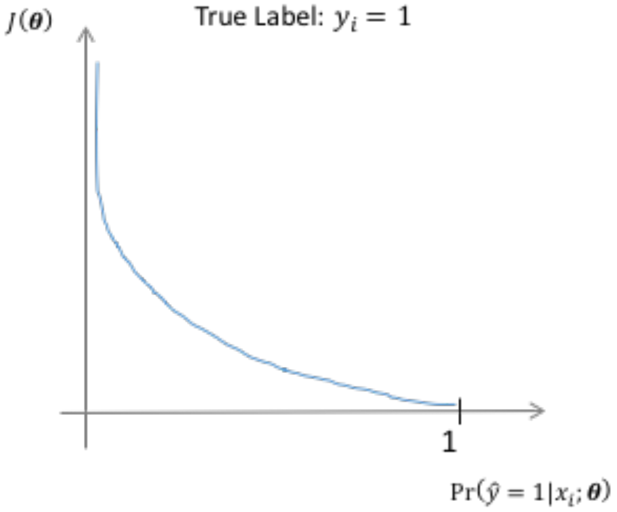
# Logistic Regression: Cost Function

How does the learning algorithm select the linear hypothesis / model? We need a cost function…

## Cost Function

$$J(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n} \mathrm{y_i} \cdot \log_2\big(\Pr(\hat{y} = 1|x_i;\boldsymbol{\theta})\big) + (1 - y_i) \cdot \log_2\big(1 - \Pr(\hat{y} = 1|x_i;\boldsymbol{\theta})\big)$$

## Why this cost function?




$y^{(i)} = 1 \wedge \Pr\big(\hat{y} = 1|x^{(i)};\boldsymbol{\theta}\big) \approx 1 \Longrightarrow$ cost is small

$y^{(i)} = 1 \wedge \Pr\big(\hat{y} = 1|x^{(i)};\boldsymbol{\theta}\big) \approx 0 \Longrightarrow$ cost is large

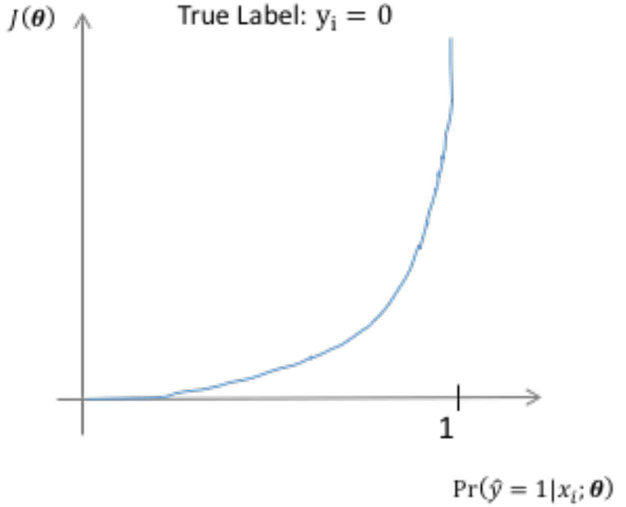
# Logistic Regression: Cost Function

How does the learning algorithm select the linear hypothesis / model? We need a cost function…

## Cost Function

$$J(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n} \mathrm{y_i} \cdot \log_2\big(\Pr(\hat{y} = 1|x_i;\boldsymbol{\theta})\big) + (1-y_i)\cdot \log_2\big(1-\Pr(\hat{y} = 1|x_i;\boldsymbol{\theta})\big)$$

## Why this cost function?




$y^{(i)} = 0 \wedge \Pr\big(\hat{y} = 1|x^{(i)};\boldsymbol{\theta}\big) \approx 1 \Longrightarrow$ cost is large

$y^{(i)} = 0 \wedge \Pr\big(\hat{y} = 1|x^{(i)};\boldsymbol{\theta}\big) \approx 0 \Longrightarrow$ cost is small

# Logistic Regression: Learning Algorithm

## Learning Algorithm

We should select the logistic regression model parameters that minimize the cost function as follows:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta})$$

## New Predictions

The new predictions are then given by:

$$Pr(\hat{y} = 1|\boldsymbol{x};\boldsymbol{\theta}^*) = \mathrm{g}(\boldsymbol{\theta}^{*t}\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{\theta}^{*t}\boldsymbol{x}}}$$

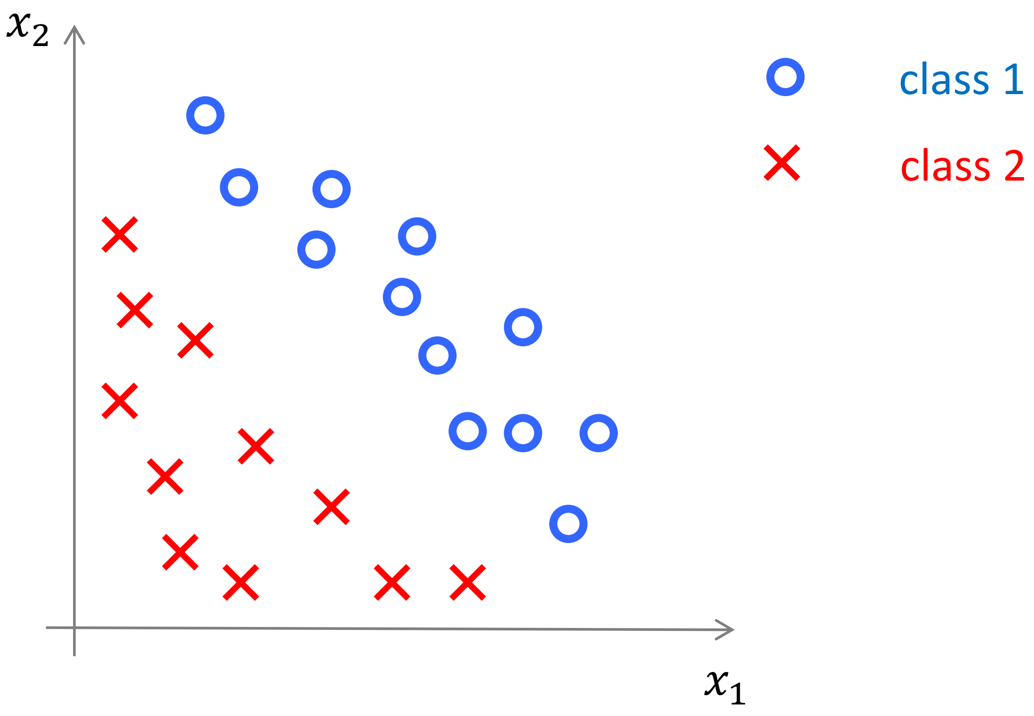
## There is no closed-form solution for the optimal parameters

⇩

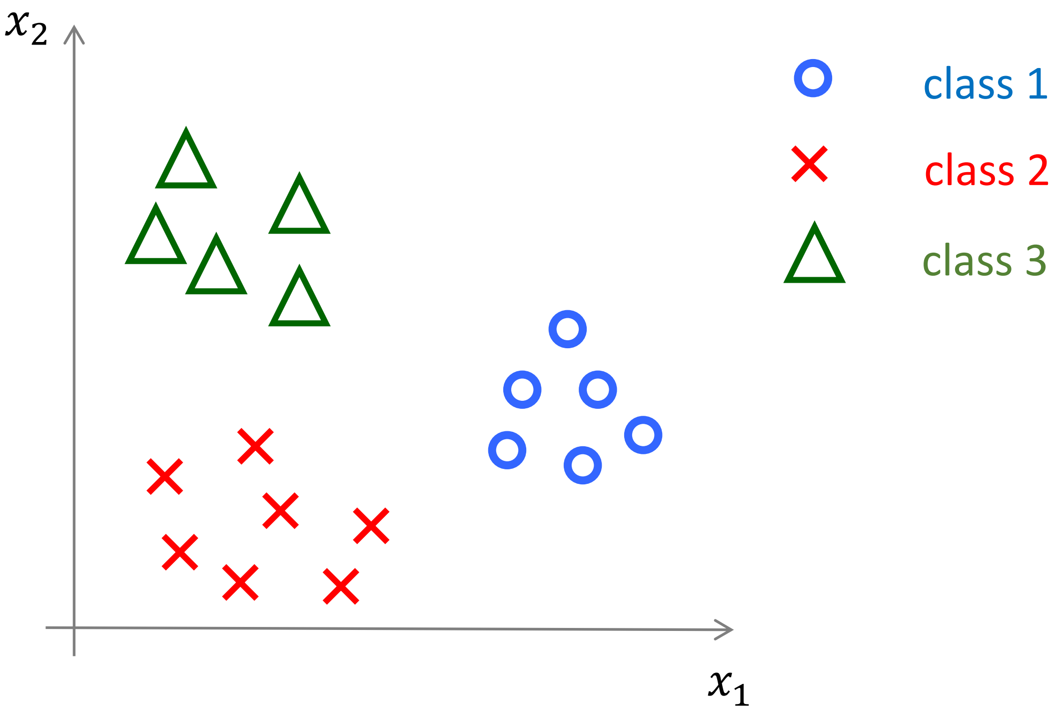
**Gradient descent algorithm**

# Multiclass Classification

## Binary Classification Problem




## Multiclass Classification Problem



# Multiclass Classification: One-vs-All

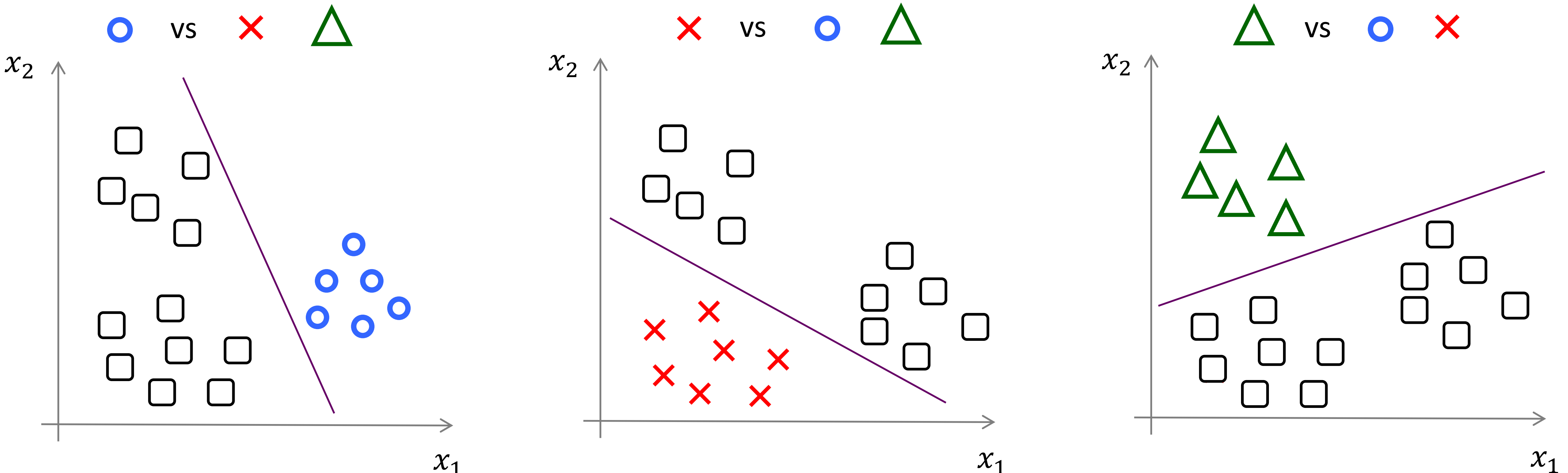
## Procedure

(1) Convert a *M*-class classification problem onto *M* binary classification problems.

(2) Learn a binary classifier – e.g. a logistic regressor – for each binary classification problem delivering a confidence score.

(3) Apply the *M* binary classifiers on new data to obtain *M* confidence scores.

(4) Report the class with the highest confidence score.



# Multiclass Classification: One-vs-One

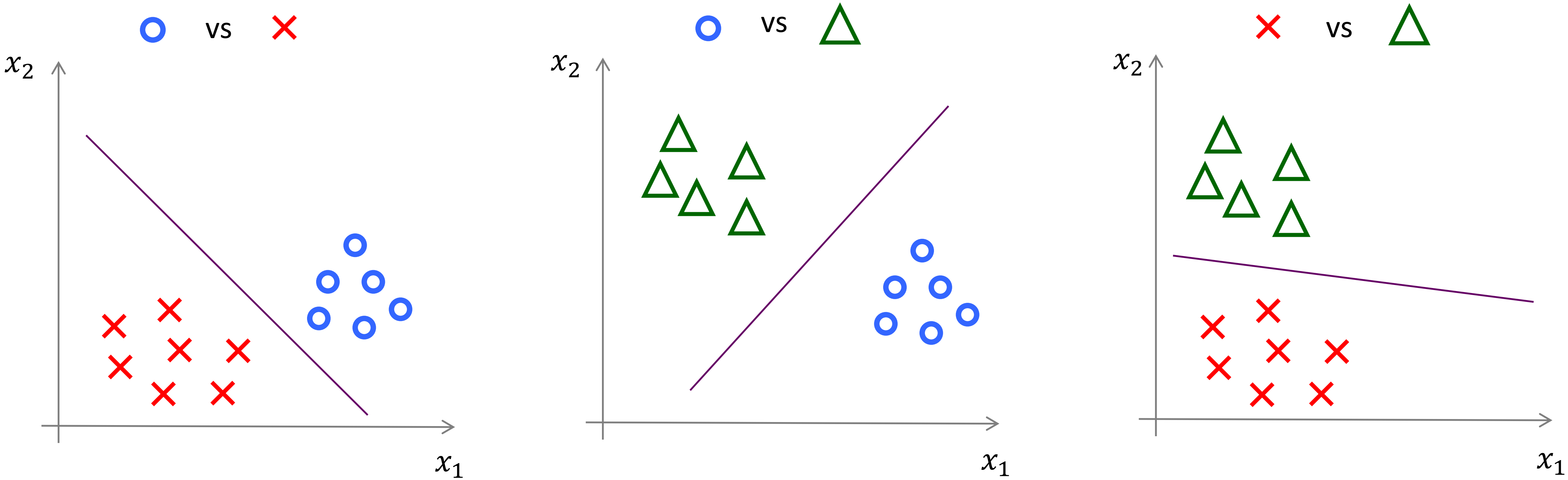
## Procedure

(1) Convert a $M$-class problem onto $(M-1)M/2$ binary classification problems.

(2) Learn a binary classifier – e.g. a logistic regressor – for each binary classification problem delivering a confidence score.

(3) Apply the $M$ binary classifiers on new data to obtain $M$ confidence scores.

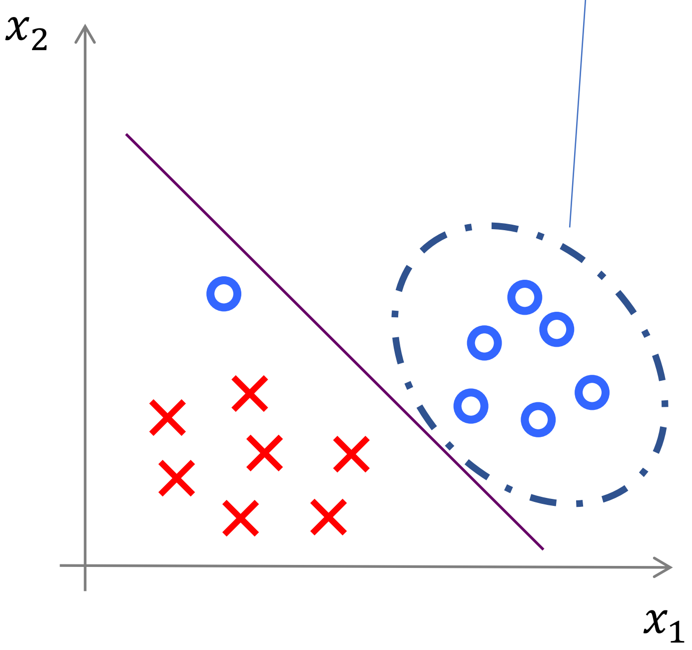
(4) Report the class that has been “voted” the highest by the different classifiers.



# Precision vs Recall

$$\text{Precision} = \frac{\text{true positive}}{\text{No. of predicted positive}}$$

$$\text{Recall} = \frac{\text{true positive}}{\text{No. of actual positive}}$$




- We want to predict y=1 only in very confident sets

Predict 1 if $h(x) \geq \cancel{0.5}$ 0.7
Predict 1 if $h(x) < \cancel{0.5}$ 0.7

High-precision and low recall classifier

- We want to predict y=1 avoiding false negative

Predict 1 if $h(x) \geq \cancel{0.5}$ 0.3
Predict 1 if $h(x) < \cancel{0.5}$ 0.3

Low-precision and high recall classifier

# Confusion Matrix

| Predicted Values | Actual Values: Positive (1) | Actual Values: Negative (0) |
|---|---|---|
| Positive (1) | TP | FP |
| Negative (0) | FN | TN |