Course 3

Introduction to Machine Learning

Christophe Eloy

Multivariate linear regression

- Hypothesis: $h_\theta(x) = \boldsymbol{\theta}^T \boldsymbol{x}$
- Parameters: $\boldsymbol{\theta} = [\theta_0 \cdots \ \theta_n]^T$
- Cost function (least squares): $J(\boldsymbol{\theta}) = \frac{1}{2N} \sum_{i=1}^{N} \left(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)} \right)^2$
- Goal: find $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
- Gradient descent: $\boldsymbol{\theta} := \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
- Normal equation: $\boldsymbol{\theta} = \left(X^T X \right)^{-1} X^T \boldsymbol{y}$

Outline

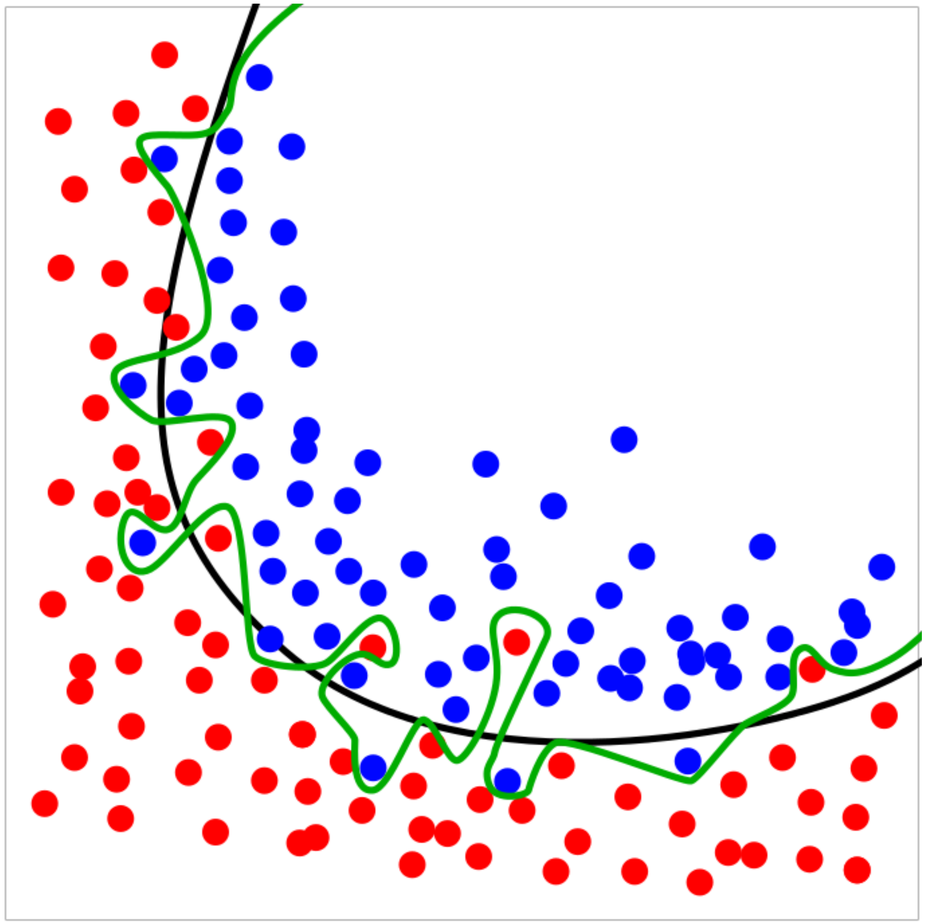
- Logistic regression
- Regularization
- Support vector machines (SVMs)
- Naive Bayes
- Decision trees
- k-means

Logistic regression

- Hypothesis: $h_\theta(x) = \dfrac{1}{1 + \mathrm{e}^{-\boldsymbol{\theta}^T \boldsymbol{x}}}$
- Parameters: $\boldsymbol{\theta} = [\theta_0 \cdots \ \theta_n]^T$
- Cost function: $J(\boldsymbol{\theta}) = -\dfrac{1}{N} \sum_{i=1}^{N} \left[y^{(i)} \log \left(h_\theta(\boldsymbol{x}^{(i)}) \right) + \left(1 - y^{(i)} \right) \log \left(1 - h_\theta(\boldsymbol{x}^{(i)}) \right) \right]$
- Gradient descent: $\theta_j := \theta_j - \alpha \dfrac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$

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

Problem of overfitting



- To avoid overfitting, one method is regularization

Regularization

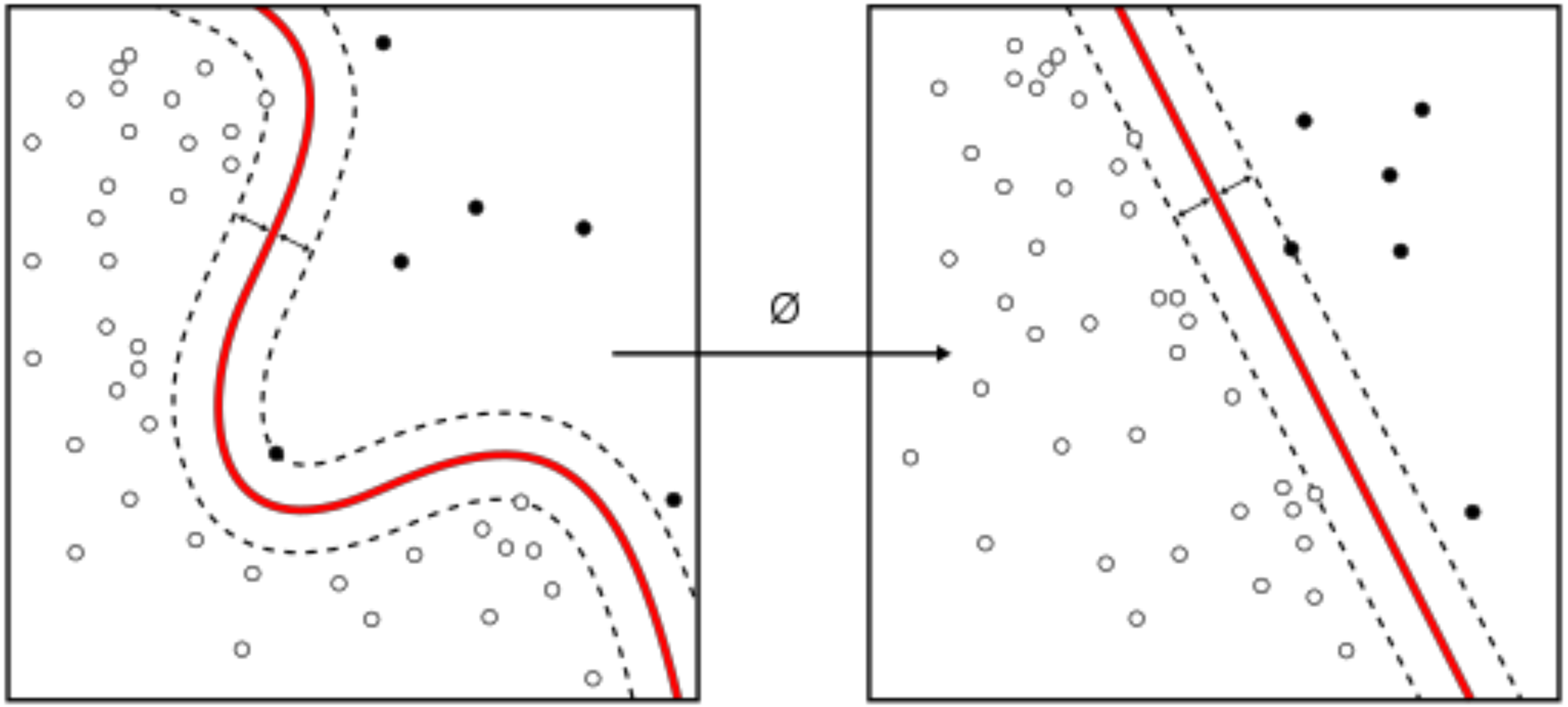
- New cost function: $J(\boldsymbol{\theta}) = \frac{1}{2N}\left[\sum_{i=1}^{N}\left(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$
- Calculation of the gradient

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{N}\sum_{i=1}^{N}\left(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)}\right)x_0^{(i)}$$

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{N}\sum_{i=1}^{N}\left(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)}\right)x_j^{(i)} + \frac{\lambda}{N}\theta_j$$

- Normal equation $\boldsymbol{\theta} = \left(X^TX + \lambda\begin{bmatrix}0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1\end{bmatrix}\right)^{-1} X^T\boldsymbol{y}$

Support vector machines



- We want “fat” margins

Support vector machines

- Support vectors (touching the margin) such that $y^{(i)}\left(\mathbf{a}^T\mathbf{x}_n + b\right) = 1$
- Size of each margin is $d = 1/\|\mathbf{a}\|$
- Optimization problem: Find $\min\left(\|\mathbf{a}\|^2\right)$ subject to $y^{(i)}\left(\mathbf{a}^T\mathbf{x} + b\right) \geq 1$
- Soft margin: Find

$$\min\left(\lambda\|\mathbf{a}\|^2 + \frac{1}{n}\sum_{i=1}^{N}\max\left(0,1 - y^{(i)}\left(\mathbf{a}^T\mathbf{x} + b\right)\right)\right)$$

- Non-linear kernels: **x** can be replaced by **f**(**x**)

Naive Bayes

- Bayes theorem

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

- “Naive” assumption of independence: $p(C_k, \mathbf{x}) = p(x_1 \mid C_k) \cdots p(x_n \mid C_k)\, p(C_k)$

- Conditional probability

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)}{p(\mathbf{x})} \prod_{i=1}^{N} p(x_i \mid C_k)$$

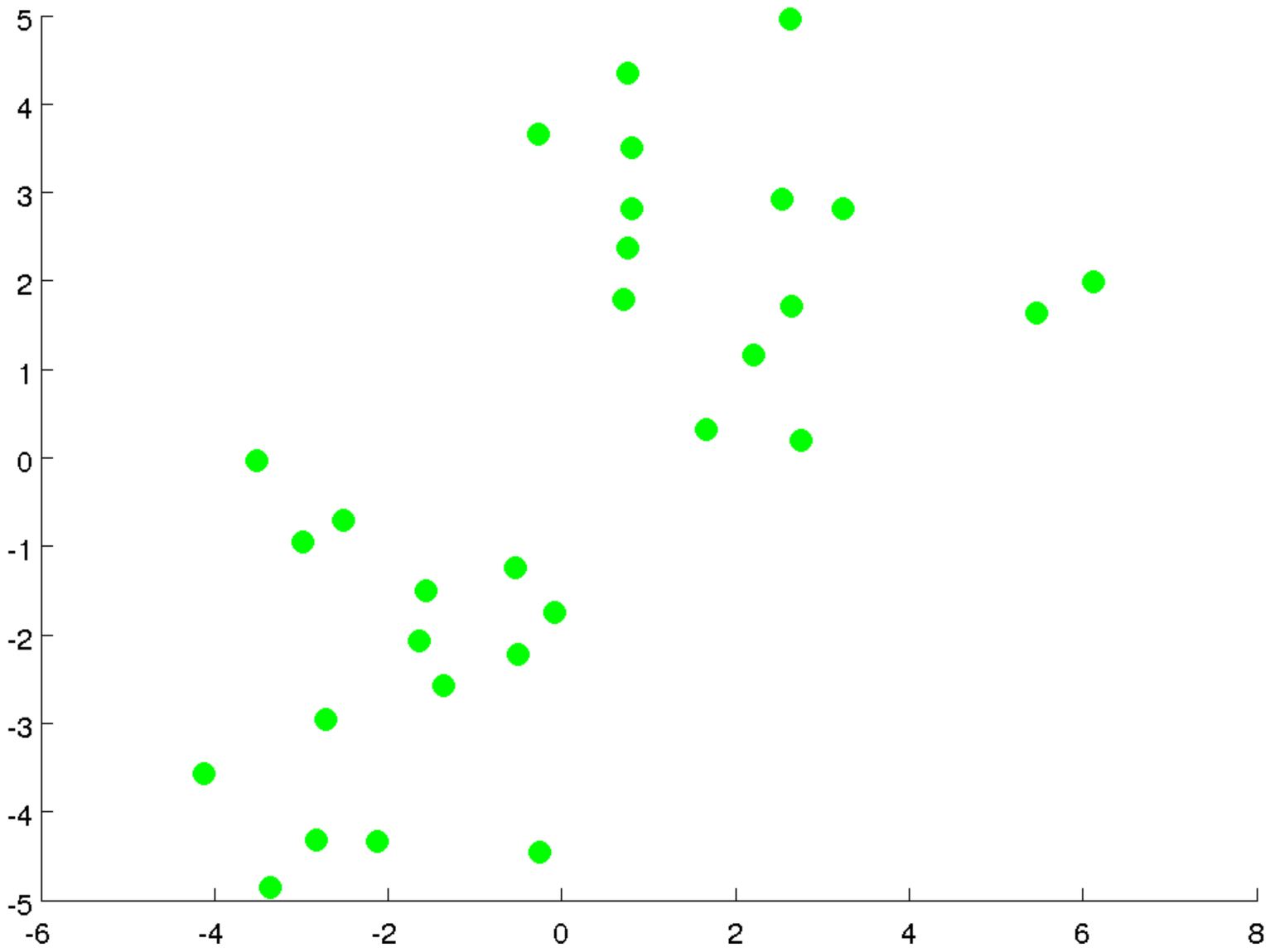
- Naive Bayes classifier: $y = \operatorname{argmax}_k \; p(C_k) \prod_{i=1}^{N} p(x_i \mid C_k)$

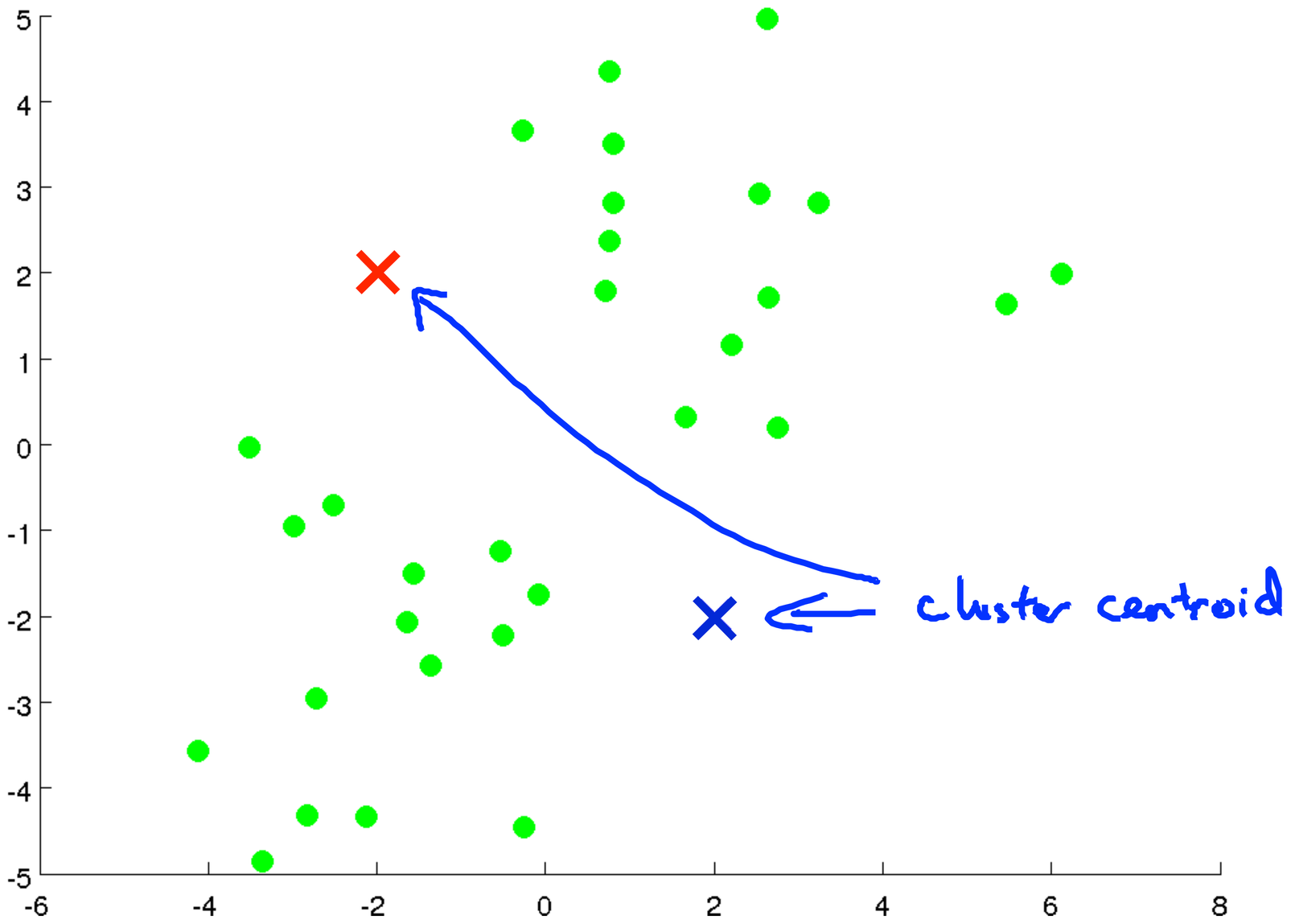
k-means clustering

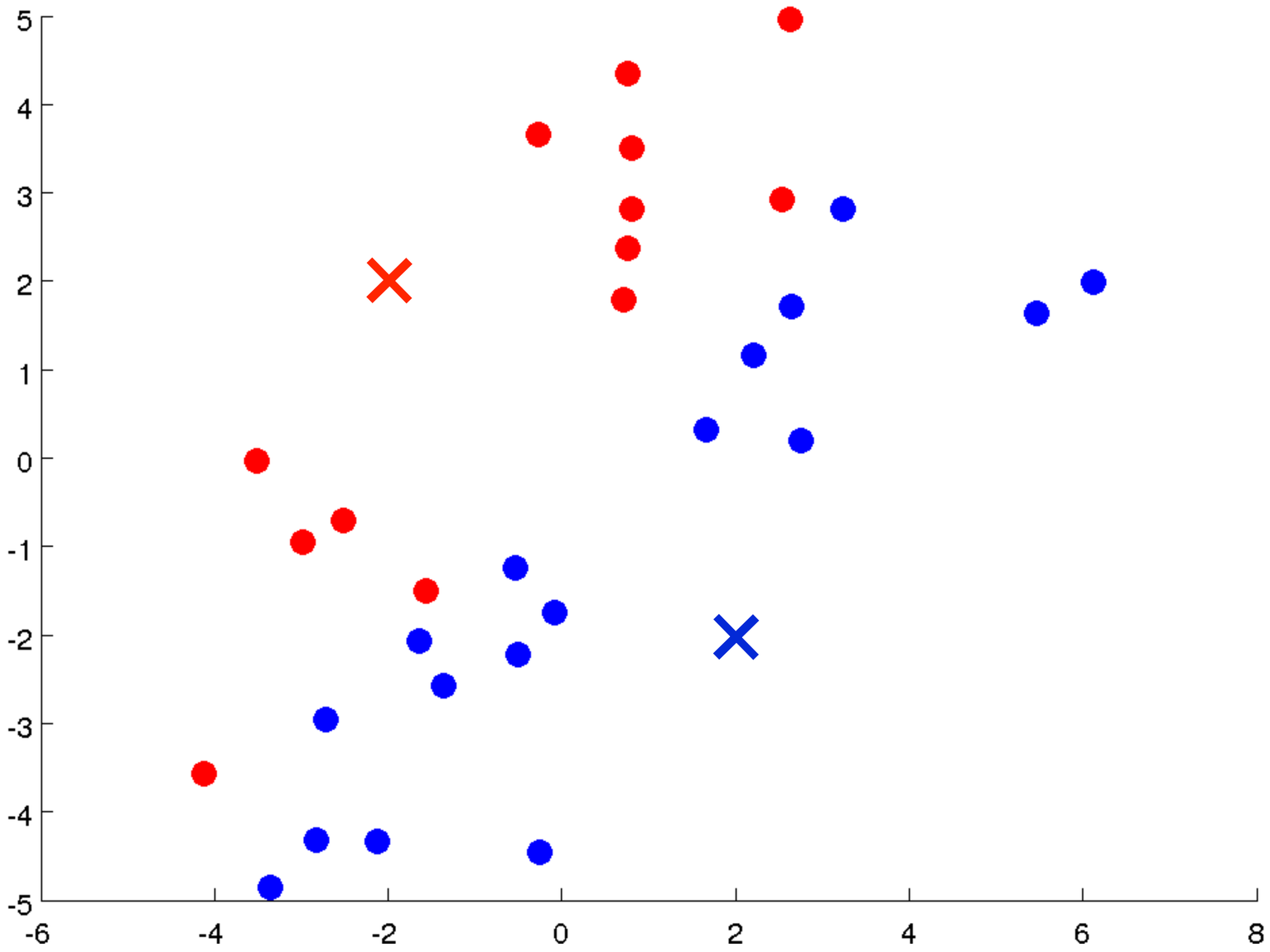
- The goal is to partition the space into *K* Voronoi cells (clusters)
- Objective function: $J = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_i\|^2$
- Where $\boldsymbol{\mu}_i$ is the mean of points in cluster S_i
- Iterative algorithm:
  - Randomly choose *K* centroids
  - Compute for each **x** the cluster to which it belongs
  - Calculate the average of **x**'s in each cluster and update centroid position

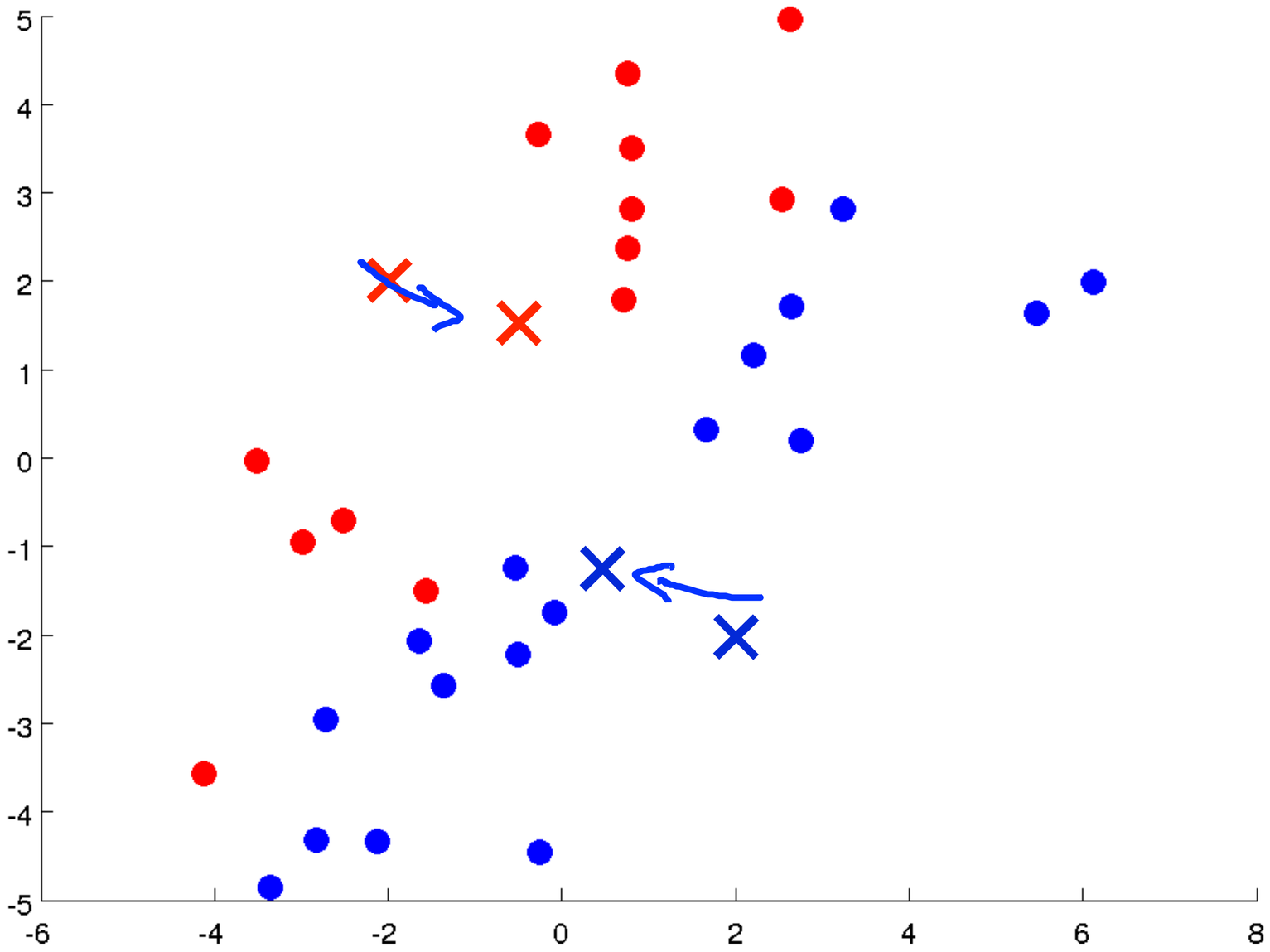
Example of *k*-means

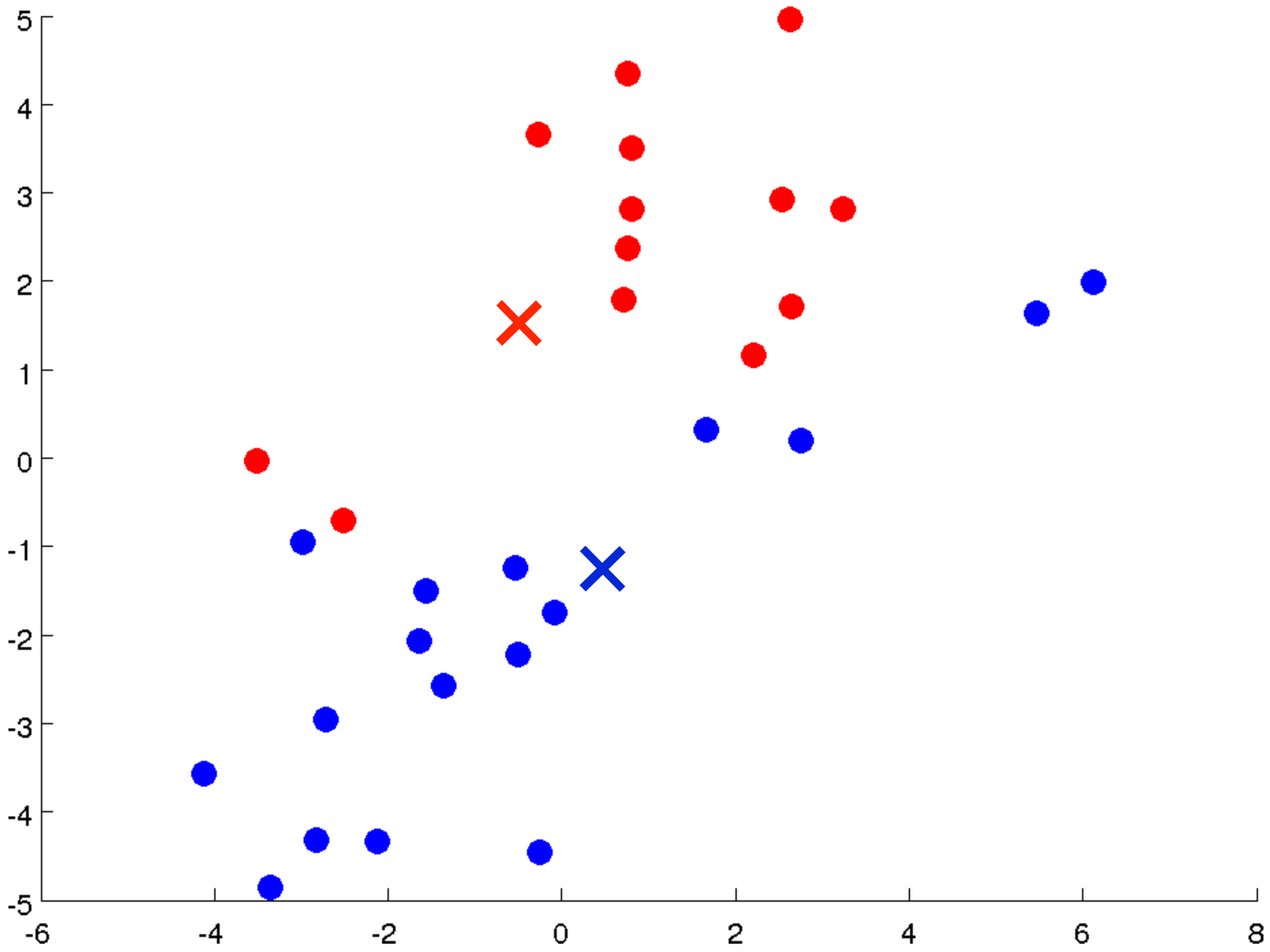
From MOOC Machine Learning, Andrew Ng (Stanford)

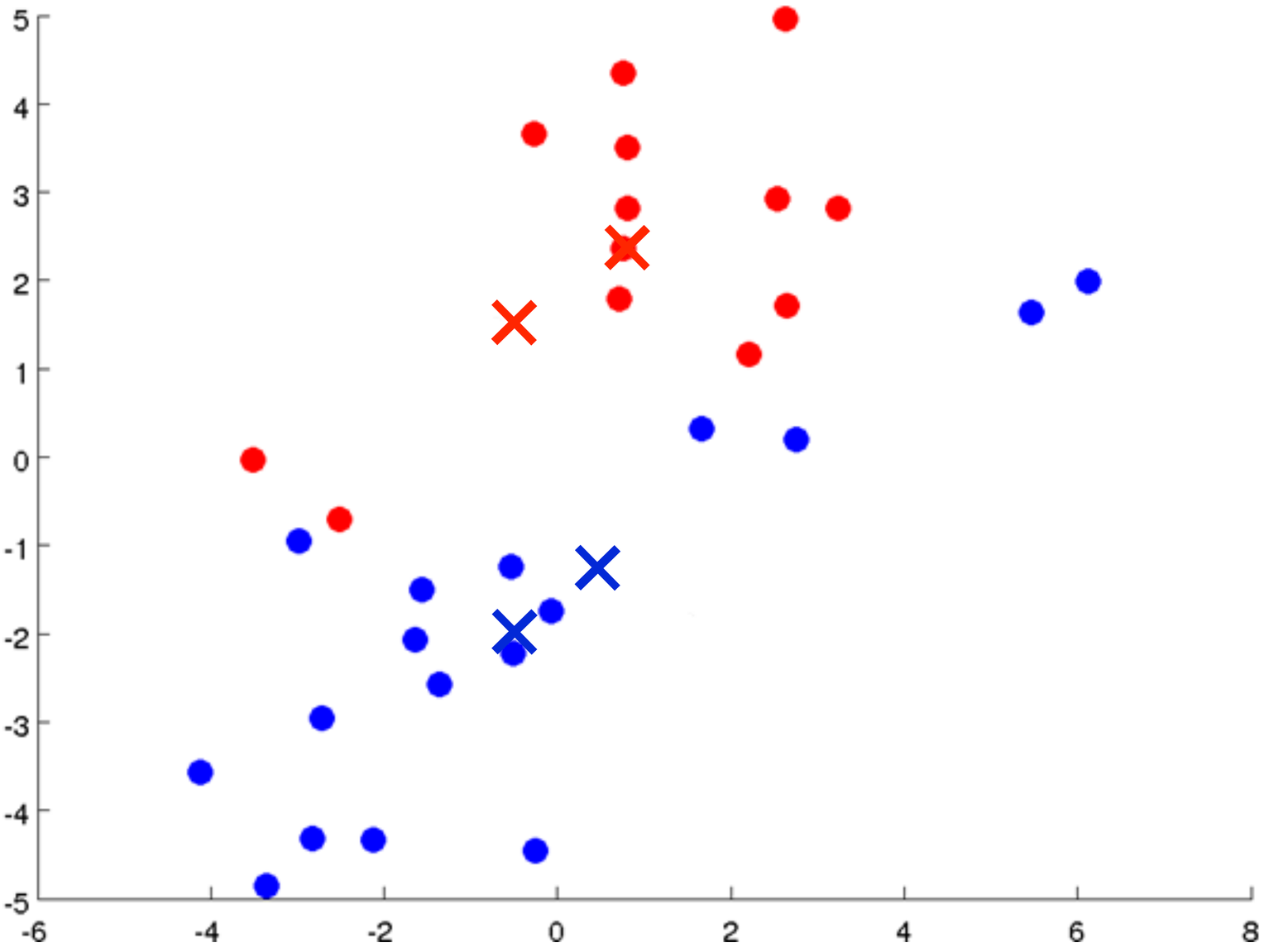

5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8

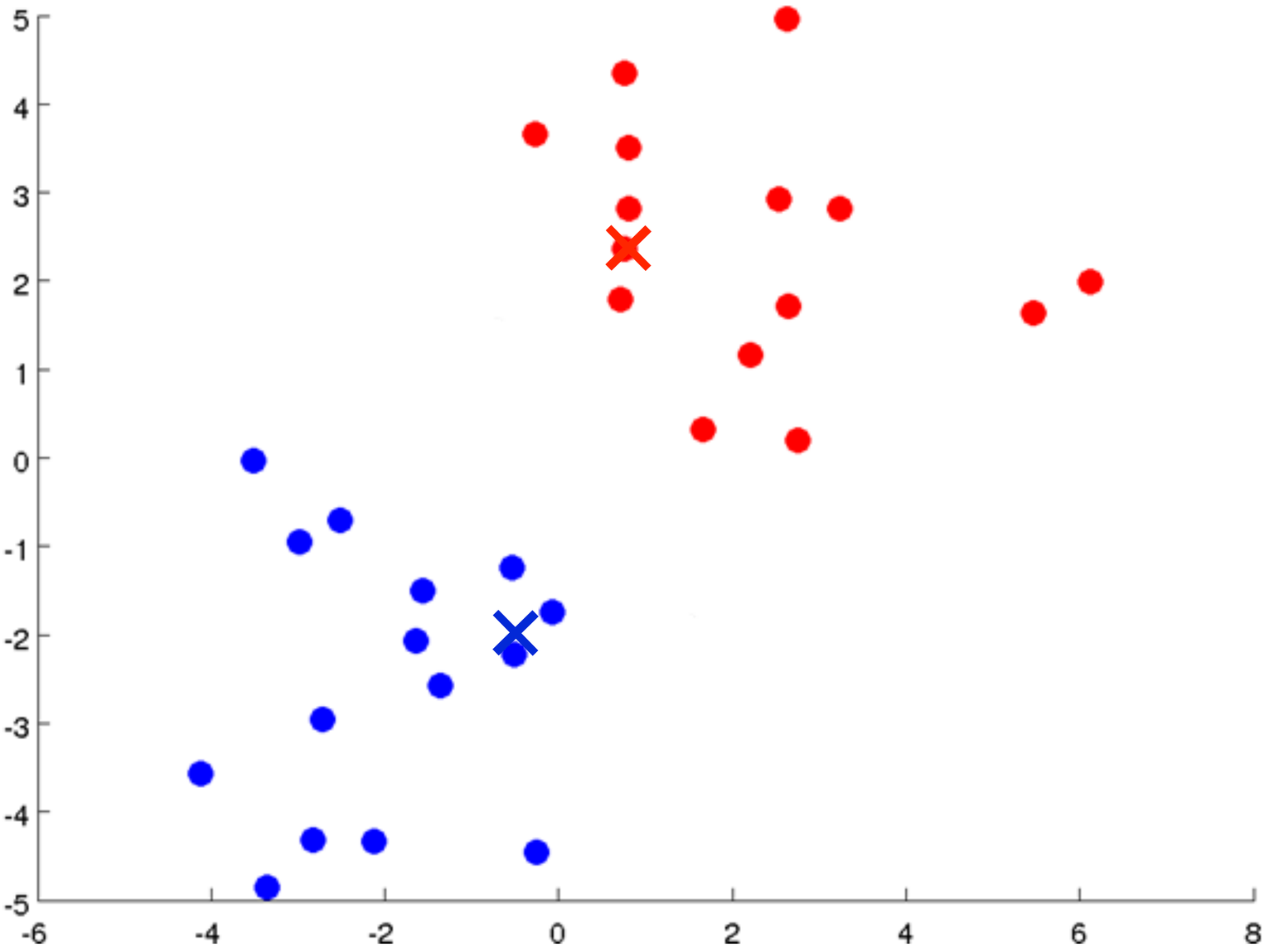

5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8
cluster centroid

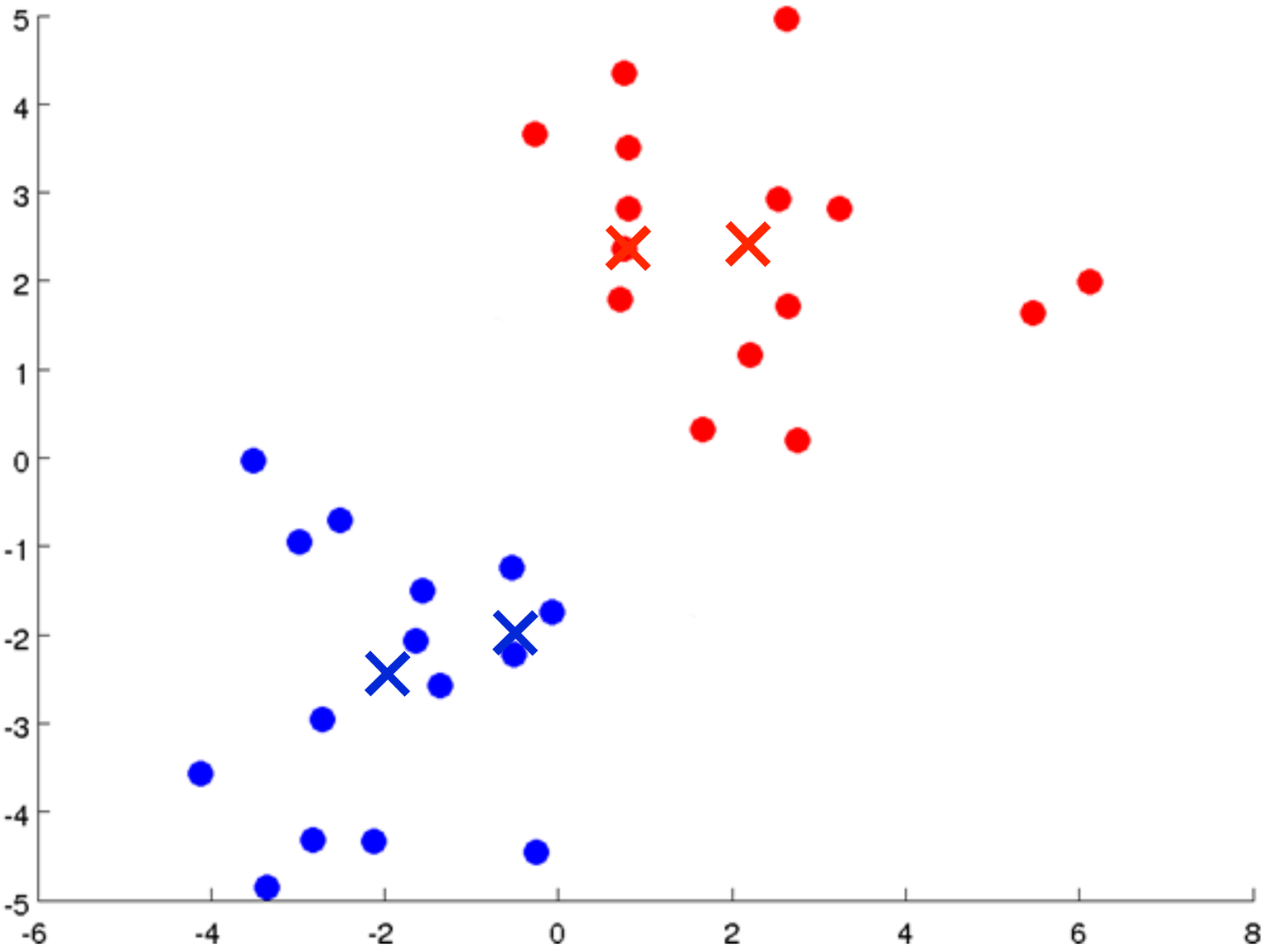

5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8

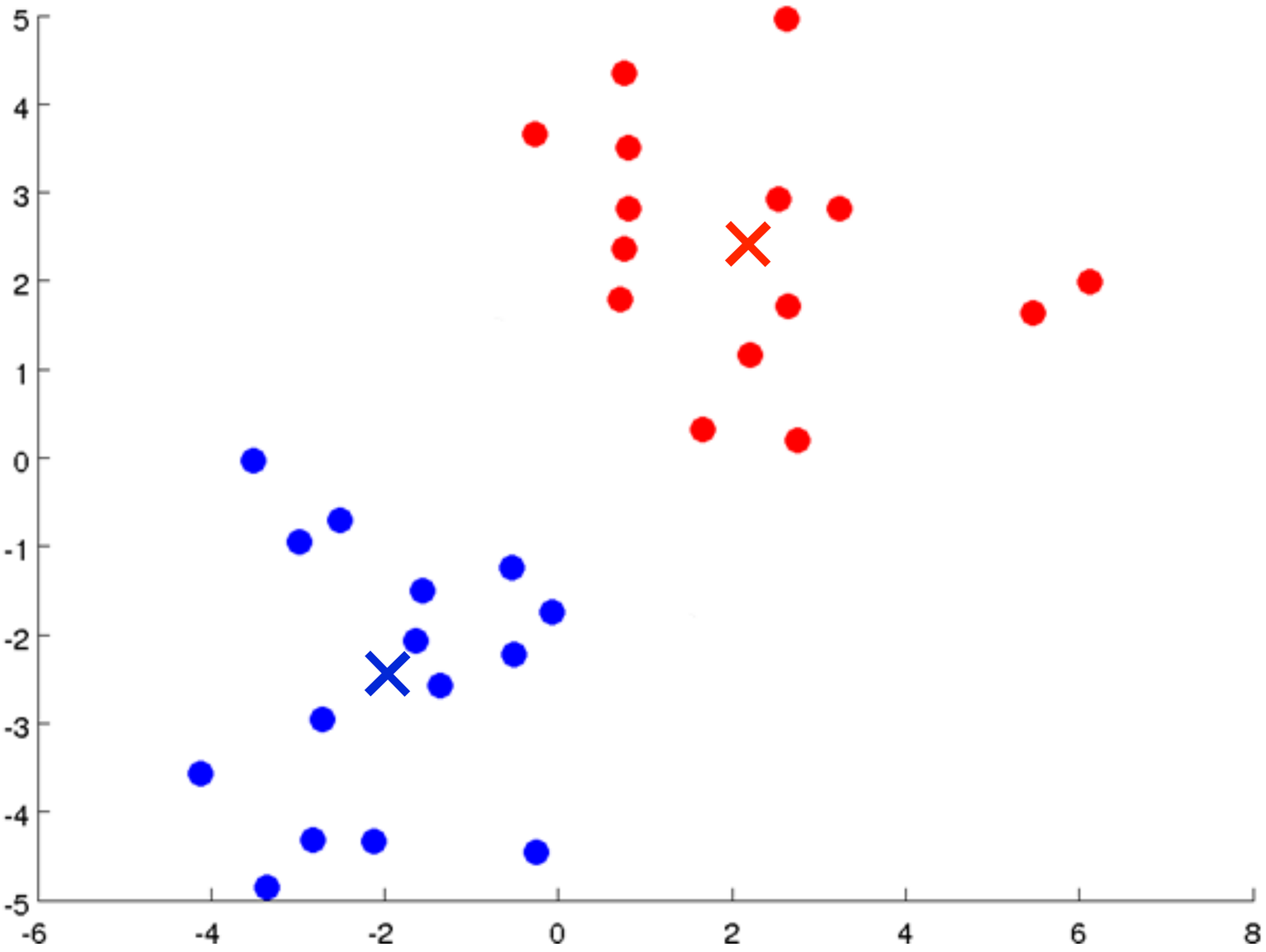

5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8

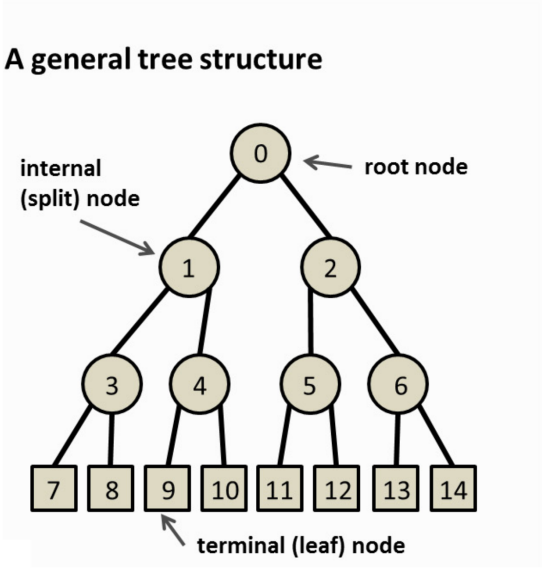

5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8

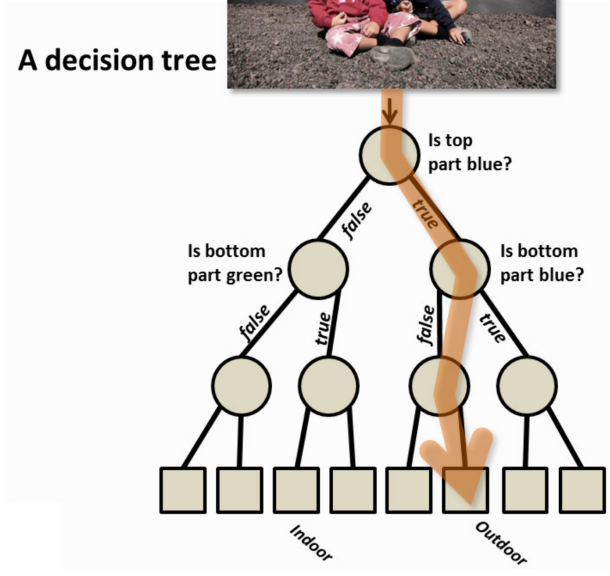

5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8

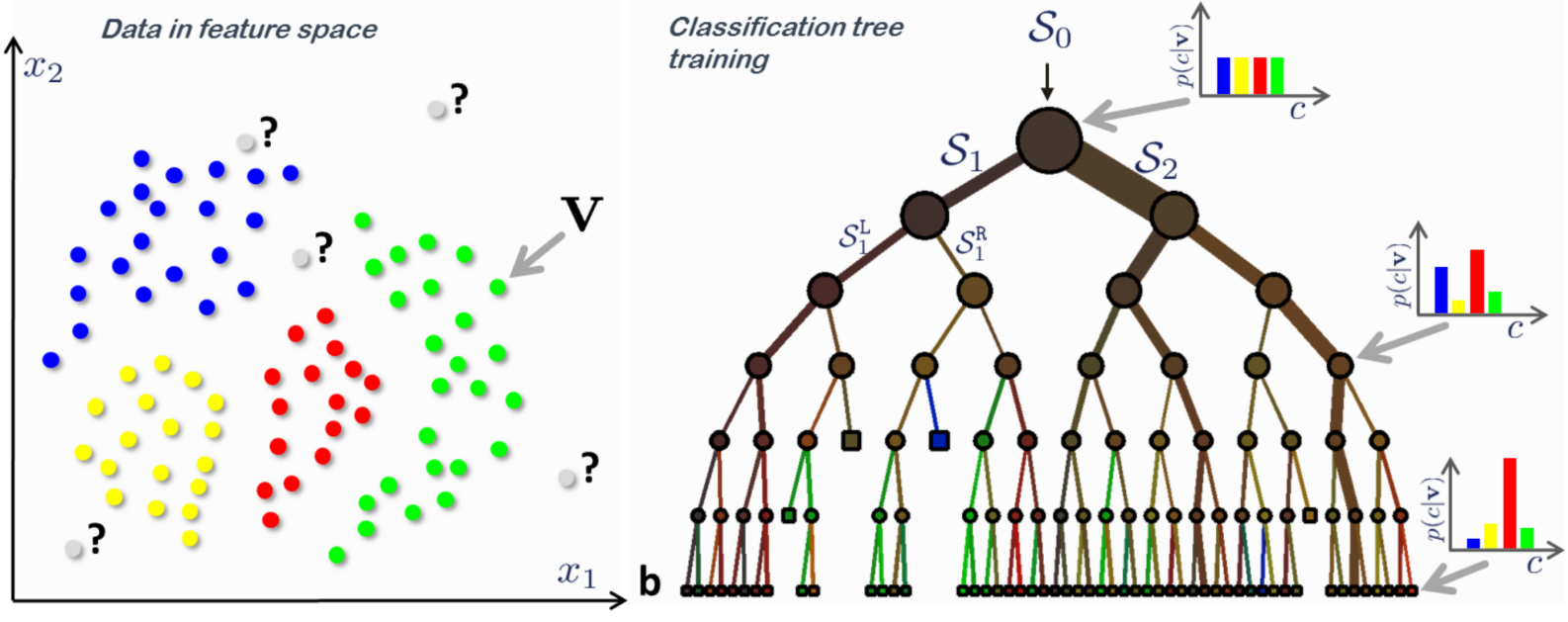
Decision Trees

Image classification example





Another classification tree



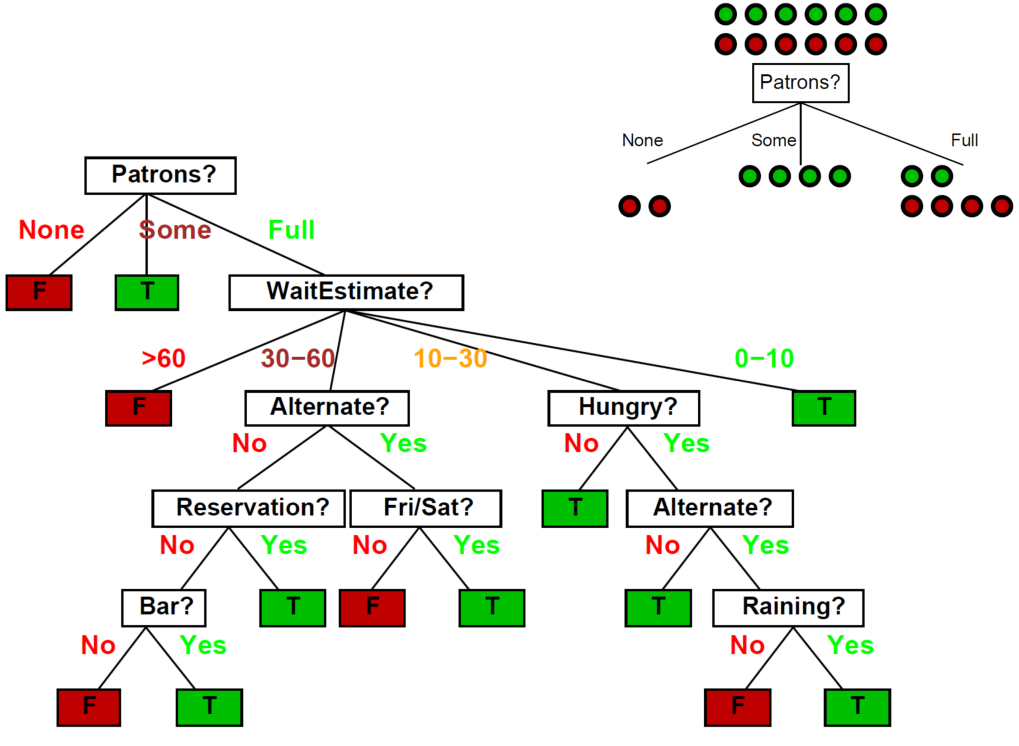
How do we build a tree

Building a node from data

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

How do we build a tree

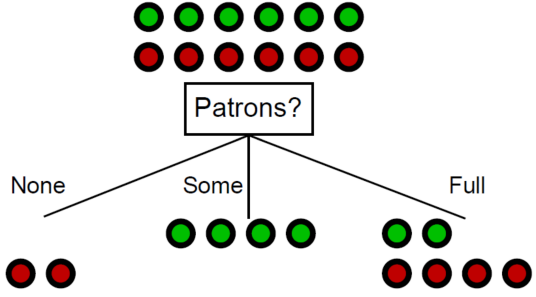
A learned tree

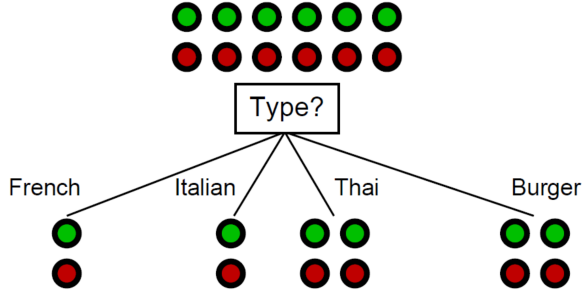


How do we build a tree

Which split is better?

Ideally we want to separate negative and positive examples





How do we build a tree

- Shannon Entropy

$$H = -\sum_i p_i log_2(p_i)$$

- Expected Entropy (for a feature F with K values)

$$EH(F) = -\sum_{i=1}^{K} \frac{n_i}{N} H_i$$

- Information Gain $I(F)$

$$I(F) = H(Data) - EH(F)$$

How do we build a tree

The patron vs type example