Machine Learning

Classification

Fernando Rodríguez Sánchez

Computational Intelligence Group

Universidad Politécnica de Madrid

27/01/2020

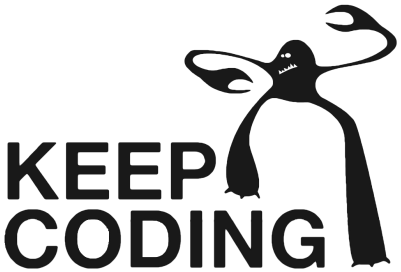


Table of contents

Table of contents

Supervised learning

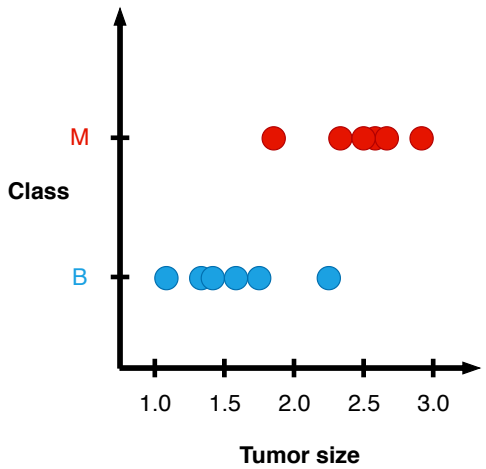
	X_1	$\dots$	X_n	Y
$(\mathbf{x}^{(1)}, y^{(1)})$	$x_1^{(1)}$	$\dots$	$x_n^{(1)}$	$y^{(1)}$
$(\mathbf{x}^{(2)}, y^{(2)})$	$x_1^{(2)}$	$\dots$	$x_n^{(2)}$	$y^{(2)}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$(\mathbf{x}^{(m)}, y^{(m)})$	$x_1^{(m)}$	$\dots$	$x_n^{(m)}$	$y^{(m)}$

Classification

- X_i is discrete/continuous
- Y is discrete (the **class**)

Introduction

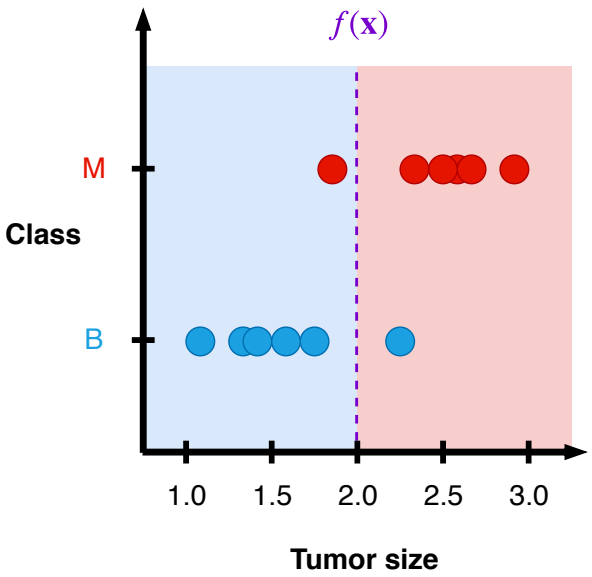
- Given $(\mathbf{x}^{(1)}, y^{(1)})$ learn a function $f(\mathbf{x})$ to predict y given $\mathbf{x}$
- y is discrete



One-dimensional

Introduction

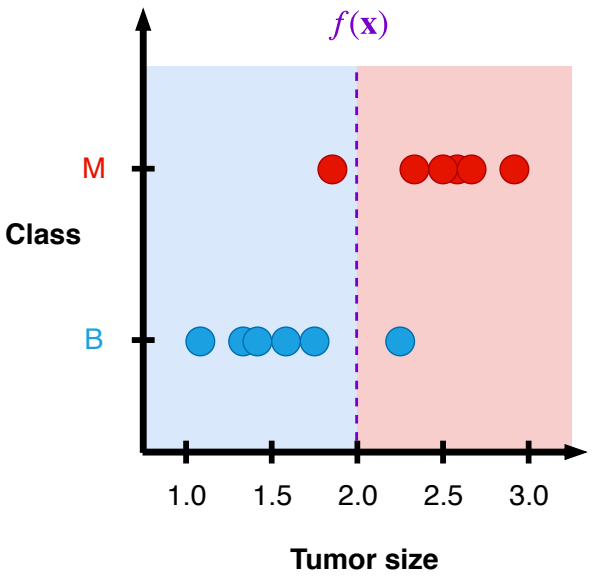
- Given $(\mathbf{x}^{(1)}, y^{(1)})$ learn a function $f(\mathbf{x})$ to predict y given $\mathbf{x}$
- y is discrete



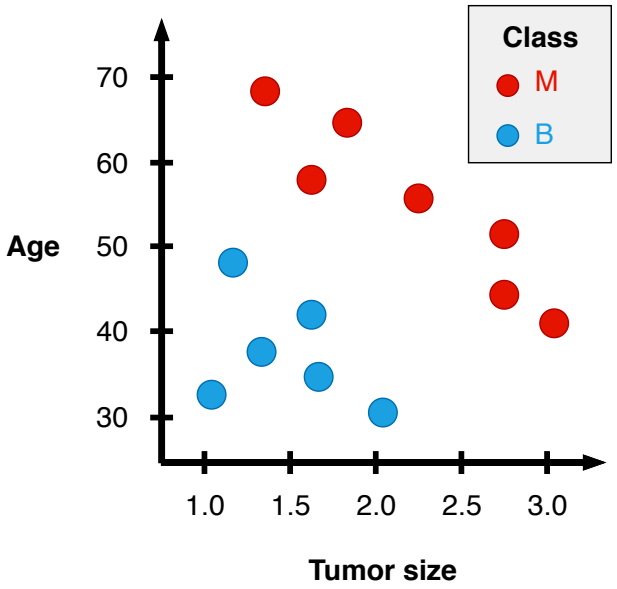
One-dimensional

Introduction

- Given $(\mathbf{x}^{(1)}, y^{(1)})$ learn a function $f(\mathbf{x})$ to predict y given $\mathbf{x}$
- y is discrete



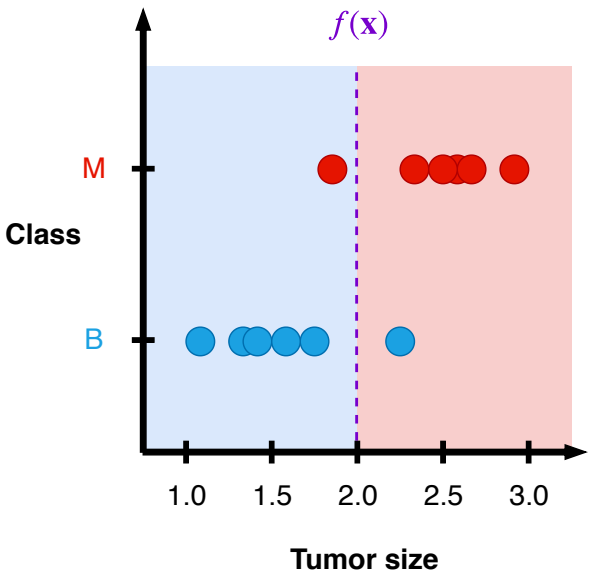
One-dimensional



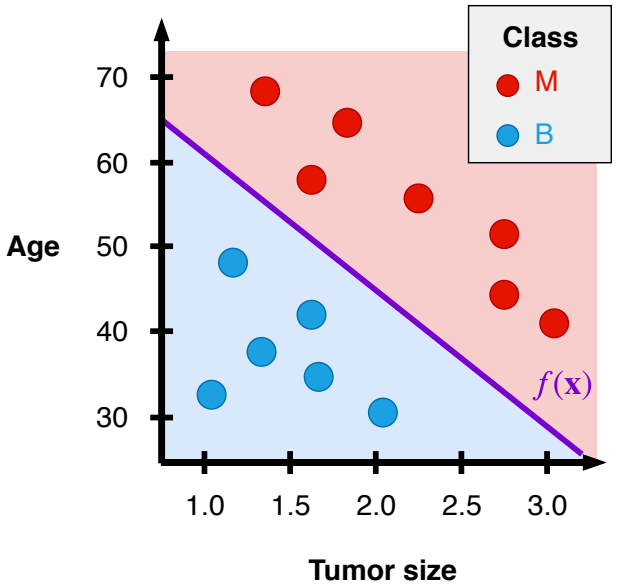
Multi-dimensional

Introduction

- Given $(\mathbf{x}^{(1)}, y^{(1)})$ learn a function $f(\mathbf{x})$ to predict y given $\mathbf{x}$
- y is discrete



One-dimensional



Multi-dimensional

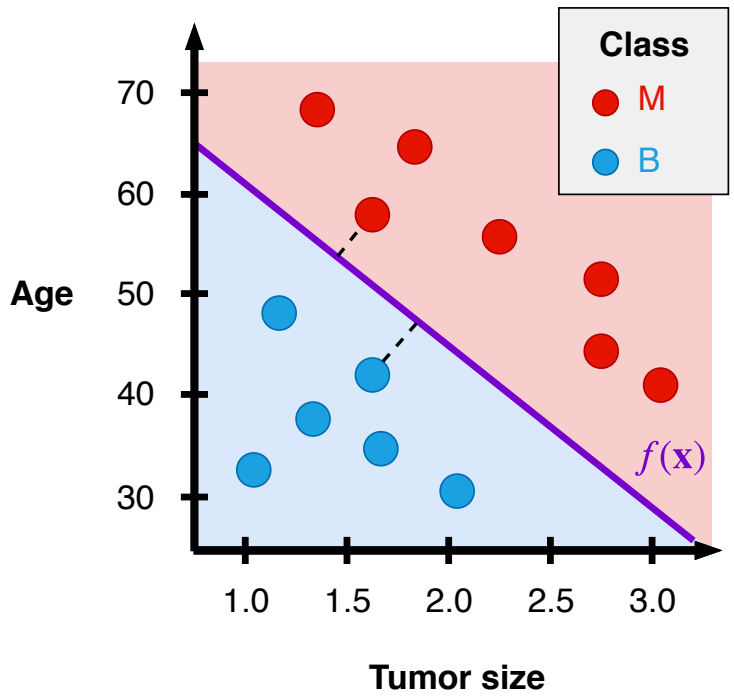
Table of contents

Support Vector Machines

Support Vector Machines try to find the linear function $f(x)$ that best separate **two** classes

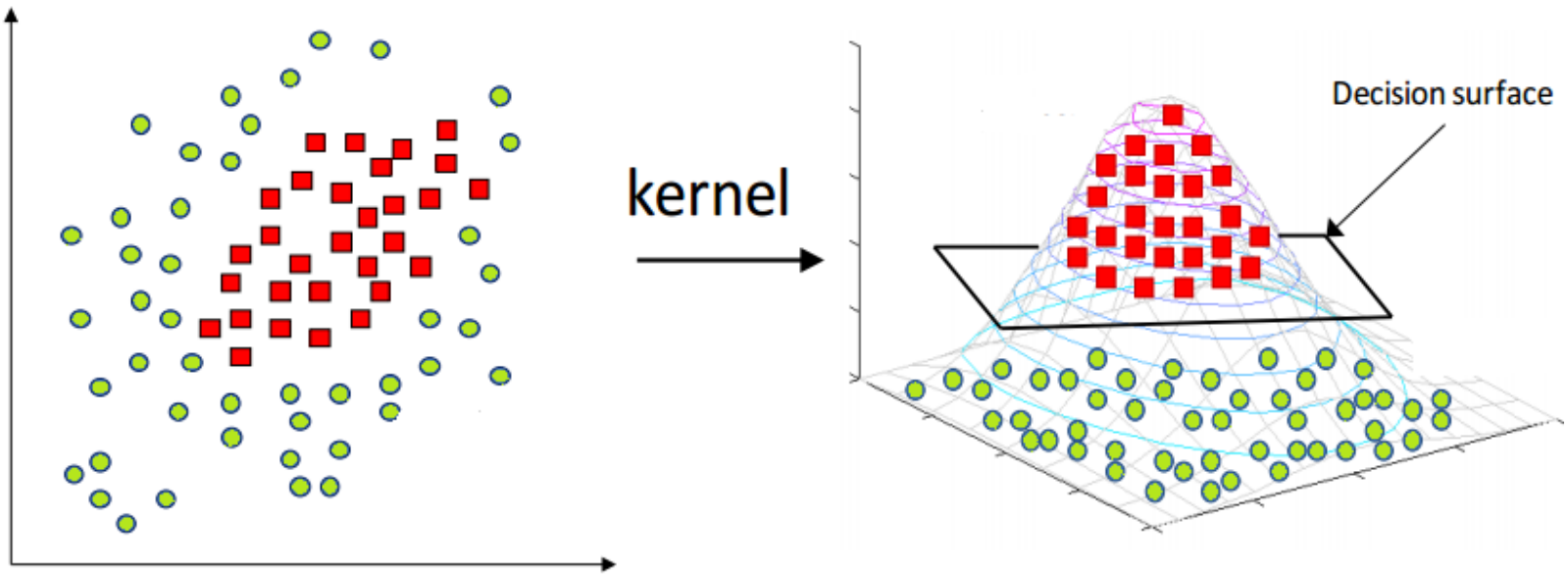
Tries to make the separation **as wide as possible**

Support vectors $\rightarrow$ closest points to the line



Kernel trick

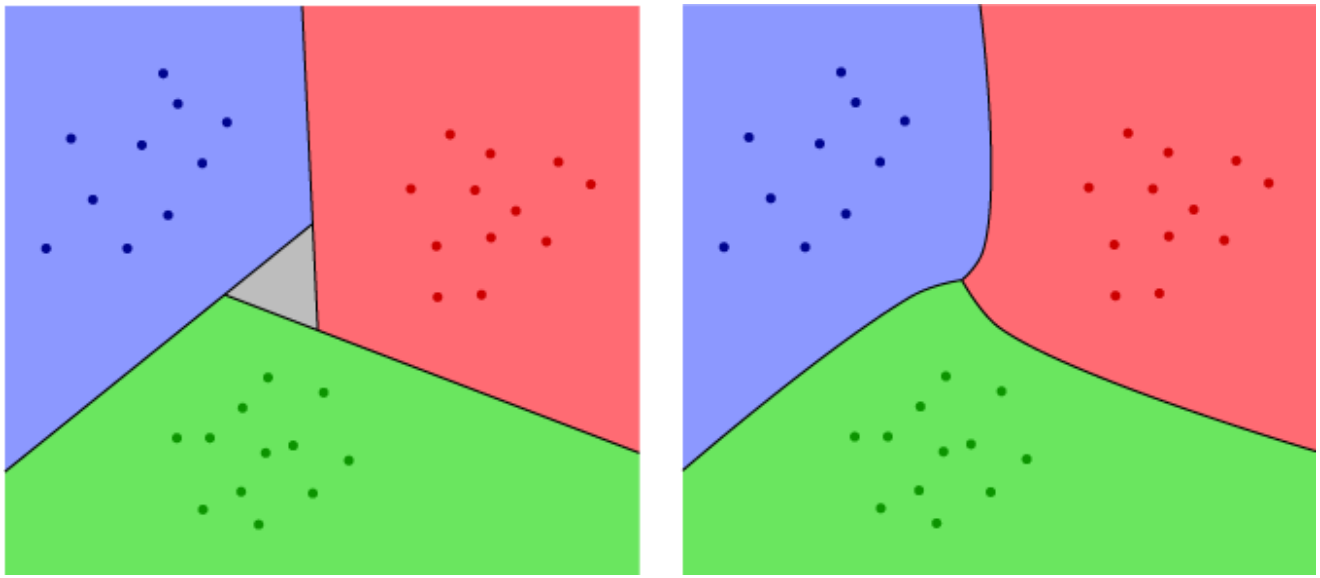
What happens when classes are not **linearly** separable?



The training points are mapped to a 3-dimensional space where a separating hyperplane can be easily found

$$(A, B) \rightarrow (A, B, A^2 + B^2)$$

Multi-class classification



Multi-class classification via **All vs. All**

What happens on **ties** (grey area)?

- Depends on implementation
- *Scikit-learn* assigns a class probability via K-fold cross validation

Strengths and weaknesses

Strengths

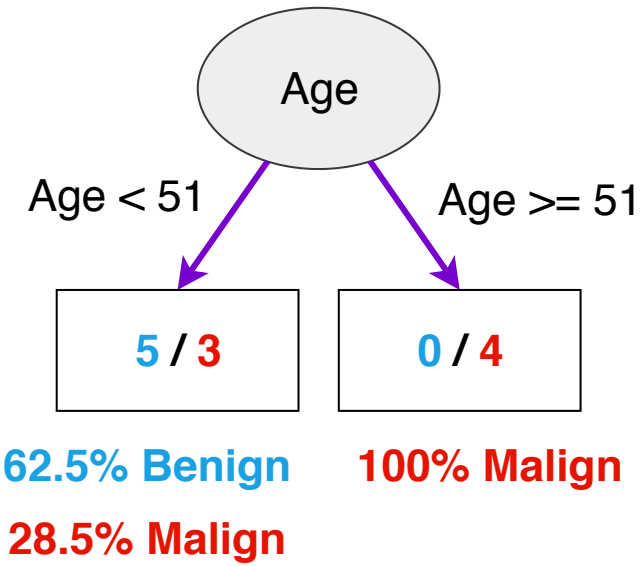
- Memory efficient (only need to store the support vectors)
- Can represent many decision boundaries via kernels
- Effective in high dimensional spaces

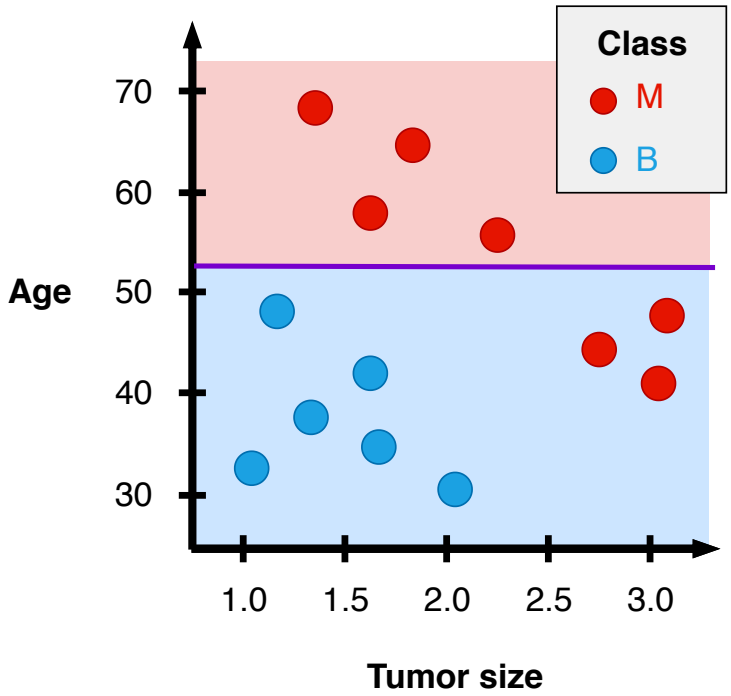
Weaknesses

- Performance is sometimes kernel-dependent
- Don't scale well to large datasets

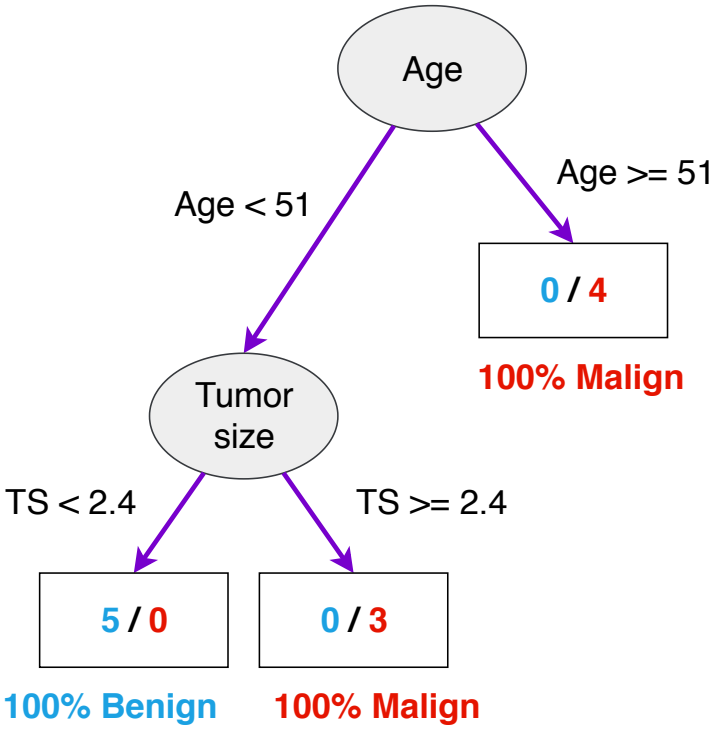
Table of contents

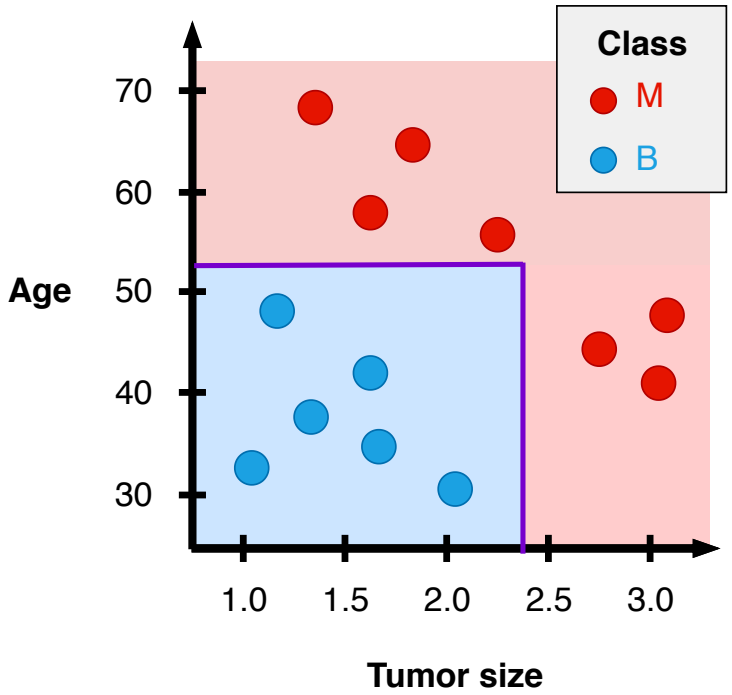
Decision trees





Decision trees





Overfitting?

Strengths and weaknesses

Strengths

- Easy to understand
- Easy to generate rules
- **Very good when done in ensembles**

Weaknesses

- Individual trees are prone to **overfitting**
- Pruning is usually necessary (when/how to **prune**?)
- Does not easily handle nonnumeric data

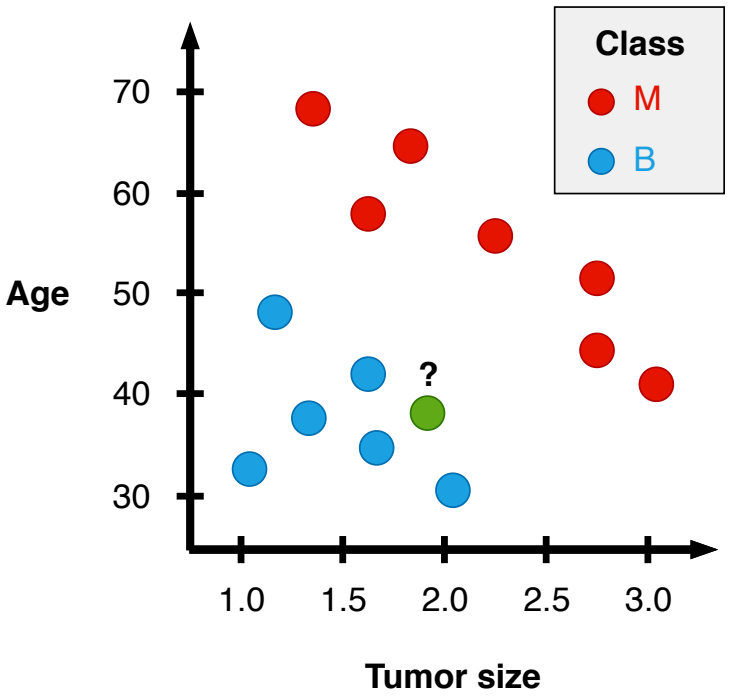
Table of contents

K-nearest neighbours

Non-parametric model (store all instances)

Procedure to classify a new x:

- Measure distance to all the other instances
- Select k closest ones
- Assigns the most frequent class of those k instances

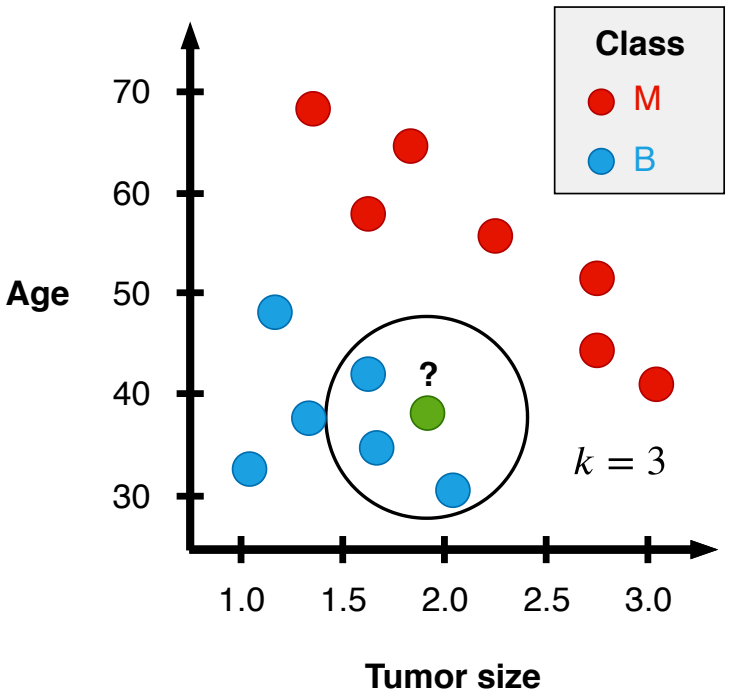


K-nearest neighbours

Non-parametric model (just store all instances)

Procedure to classify a new x:

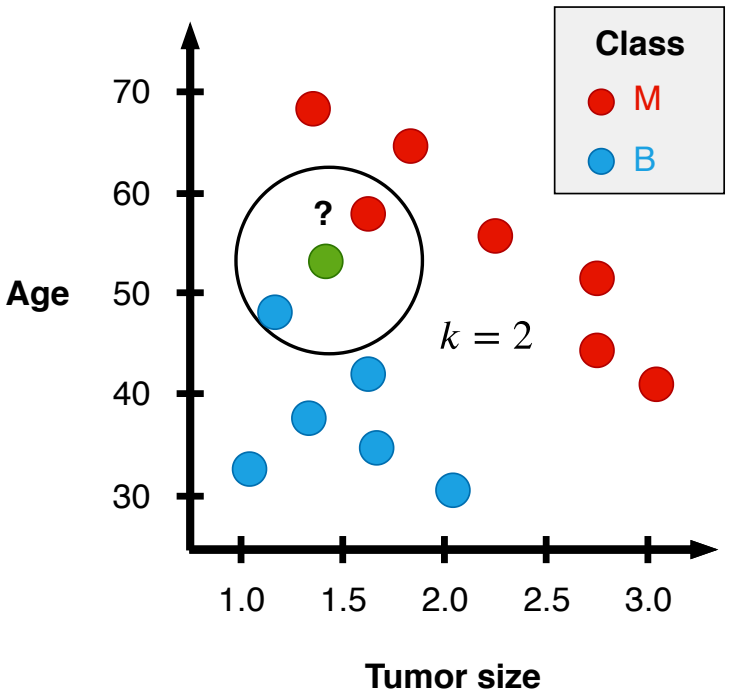
- Measure distance to all the other instance
- Select k closest ones
- Assigns the most frequent class of those k instances



K-nearest neighbours

What happens if there is a **tie**?

- Depends on implementation
- *Scikit-learn* chooses the first ordered instance of the k and assigns its class to x



Strengths and weaknesses

Strengths

- Easy to understand
- Can represent any function with enough data

Weaknesses

- Memory intensive
- Problems on high dimensional data (distances)

Machine Learning

Classification

Fernando Rodríguez Sánchez

Computational Intelligence Group

Universidad Politécnica de Madrid

27/01/2020